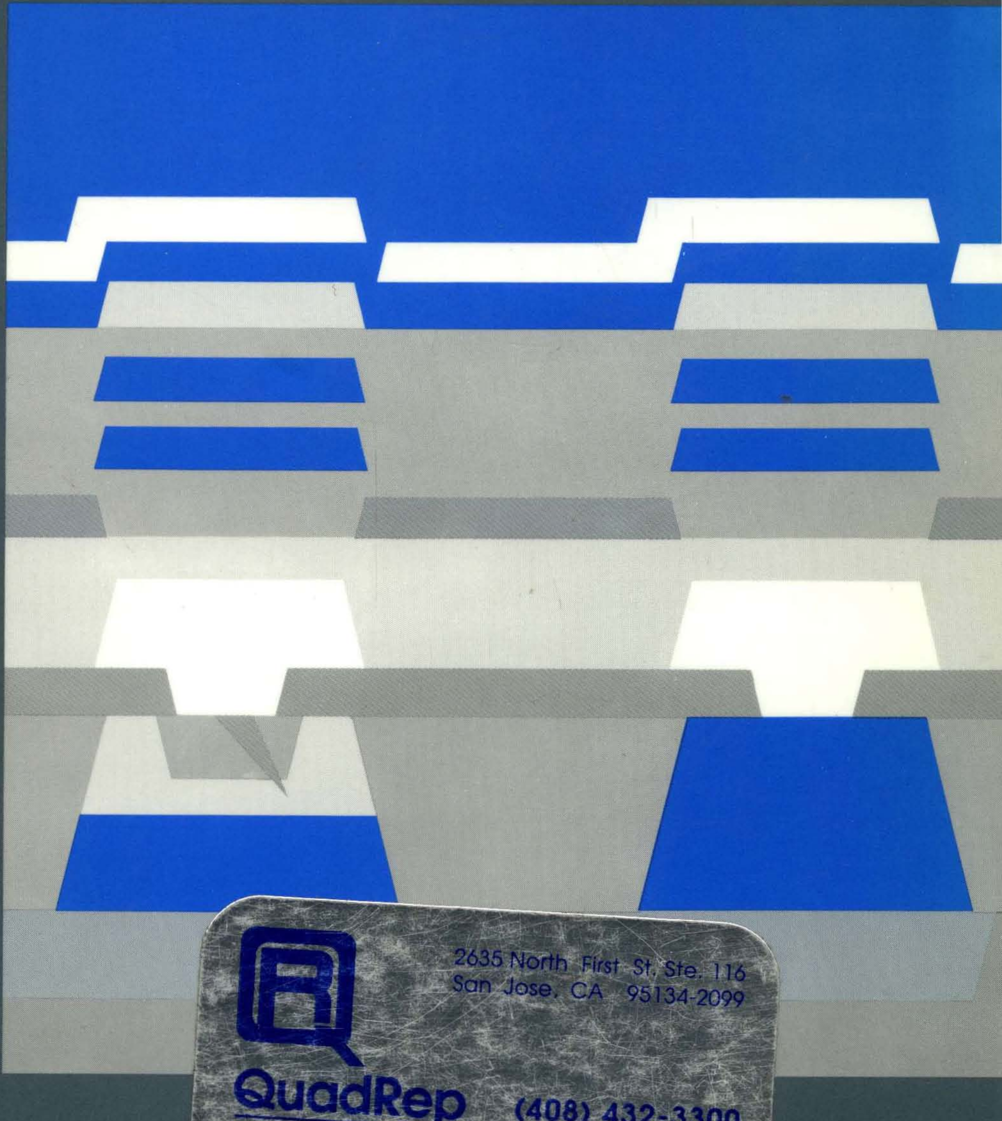
HITACHI®

8/16-BIT MICROPROCESSOR DATA BOOK



8/16-BIT MICROPROCESSOR DATA BOOK

HITACHI®

M21T132

When using this document, keep the following in mind:

1. This document may, wholly or partially, be subject to change without notice.
2. All rights are reserved: No one is permitted to reproduce or duplicate, in any form, the whole or part of this document without Hitachi's permission.
3. Hitachi will not be held responsible for any damage to the user that may result from accidents or any other reasons during operation of the user's unit according to this document.
4. Circuitry and other examples described herein are meant merely to indicate the characteristics and performance of Hitachi's semiconductor products. Hitachi assumes no responsibility for any intellectual property claims or other problems that may result from applications based on the examples described herein.
5. No license is granted by implication or otherwise under any patents or other rights of any third party or Hitachi, Ltd.
6. MEDICAL APPLICATIONS: Hitachi's products are not authorized for use in MEDICAL APPLICATIONS without the written consent of the appropriate officer of Hitachi's sales company. Such use includes, but is not limited to, use in life support systems. Buyers of Hitachi's products are requested to notify the relevant Hitachi sales offices when planning to use the products in MEDICAL APPLICATIONS.

September 1991 Printed in U.S.A.

INDEX
8/16-Bit Microprocessor Data Book

DATA SHEETS

CONTENTS

SECTION 1

■ GENERAL INFORMATION

SECTION 2

■ DATA SHEETS

SECTION 3

SECTION 4

Section One

General Information

- Quick Reference Guide
- Introduction of Packages
- Sockets
- Device Packing
- Reliability and Quality Assurance
- Reliability Test Data of Microcomputer
- Program Development and Support System
- Device Availability

QUICK REFERENCE GUIDE

■ NMOS 8-BIT MICROPROCESSOR

Type No.	HD6802	HD6802W	HD6803 HD6803-1	HD6809 HD68A09 HD68B09	HD6809E HD68A09E HD68B09E
Clock Frequency (MHz)	1.0	1.0	1.0 (HD6803) 1.25(HD6803-1)	1.0 (HD6809) 1.5 (HD68A09) 2.0 (HD68B09)	1.0 (HD6809E) 1.5 (HD68A09E) 2.0 (HD68B09E)
Supply Voltage (V)	5.0	5.0	5.0	5.0	5.0
Operating Temperature* (°C)	−20~+75	−20~+75	0~+70	−20~+75	−20~+75
RAM (byte)	128	256	128	–	–
Oscillator	Yes	Yes	Yes	Yes	–
Package	DP-40	DP-40	DP-40	DP-40	DP-40
Features	• Internal oscillator and RAM added to the HD6800 • 32 byte RAM Battery backed up possible		• Upward instruction compatibility with the HD6800 • On-chip SCI and timer	• The highest version of the HMCS6800 family • Powerful addressing modes • Easy relocatable/reentrant programming	• Full software compatibility with the HD6809 • Bus employment on time sharing basis • External clock
Compatibility	MC6802	–	MC6803 MC6803-1	MC6809 MC68A09 MC68B09	MC6809E MC68A09E MC68B09E

* Wide Temperature Range (−40~+85°C) version is available.

Quick Reference Guide

■ **CMOS 8-BIT MICROPROCESSOR**

Type No.	HD6303R HD63A03R HD63B03R	HD6303X HD63A03X HD63B03X	HD6303Y HD63A03Y HD63B03Y HD63C03Y
Clock Frequency (MHz)	1.0 (HD6303R) 1.5 (HD63A03R) 2.0 (HD63B03R)	1.0 (HD6303X) 1.5 (HD63A03X) 2.0 (HD63B03X)	1.0 (HD6303Y) 1.5 (HD63A03Y) 2.0 (HD63B03Y) 3.0 (HD63C03Y)
Supply Voltage (V)	5.0	5.0	5.0
Operating Temperature* (°C)	0 ~ +70	0 ~ +70	0 ~ +70
RAM (byte)	128	192	256
External Memory Expansion (byte)	65k	65k	65k
Package	DP-40, FP-54 CG-40, CP-52, CP-44	DP-64S, FP-80, CP-68	DP-64S, FP-64, CP-68
Features	● On-chip timer and synchronous/asynchronous SCI ● Upward instruction compatibility with the HD6800 ● Low power consumption modes (sleep and standby)		

Type No.	HD6305X2 HD63A05X2 HD63B05X2	HD6305Y2 HD63A05Y2 HD63B05Y2	HD63B09/E HD63C09/E	HD64180R/Z
Clock Frequency (MHz)	1.0 (HD6305X2) 1.5 (HD63A05X2) 2.0 (HD63B05X2)	1.0 (HD6305Y2) 1.5 (HD63A05Y2) 2.0 (HD63B05Y2)	2.0 (HD63B09/E) 3.0 (HD63C09/E)	6.0 (HD64180R/Z-6) 8.0 (HD64180R/Z-8) 10.0 (HD64180R/Z-10)
Supply Voltage (V)	5.0	5.0	5.0	5.0
Operating Temperature* (°C)	0 ~ +70	0 ~ +70	-20 ~ +75	-20 ~ +75
RAM (byte)	128	256	—	—
External Memory Expansion (byte)	16k	16k	65k	512k/1M
Package	DP-64S, FP-64	DP-64S, FP-64	DP-40 CP-44	DP-64S CP-68 FP-80
Features	• On-chip timer and synchronous SCI • Powerful bit manipulation instruction • Low power consumption modes (wait, stop and standby)		• Software compatibility with the HD6809/E • Easy relocatable/ reentrant programming • Flexible system expansion capabilities • Powerful addressing mode	• On-chip MMU, DMAC, synchronous/ asynchronous SCI and timer • Software compatibility with Z80/8080 • R version—68/63, 80xx interface • Z version—Z80 interface

Type No.	HD64180S	HD641180X	HD643180X	HD647180X
Clock Frequency (MHz)	8.0 (HD64180SLP-8) 10 0 (HD64180SLP-10)	4.0 (HD641180X-4) 6 0 (HD641180X-6) 8.0 (HD641180X-8L)	4 0 (HD643180X-4) 6 0 (HD643180X-6) 8 0 (HD643180X-8L)	4.0 (HD647180X-4) 6.0 (HD647180X-6) 8.0 (HD647180X-8L)
Supply Voltage (V)	5.0	5.0	5.0	5.0
Operating Temperature* (°C)	-20 ~ +75	-20 ~ +75	-20 ~ +75	-20 ~ +75
RAM (byte)	—	512	512	512
External Memory Expansion (byte)	1M	1M	1M	1M
Package	CP-84	CP-90S, FP-80B, CP-84	DP-90S, FP-80G, CP-84	DP-90S, FP-80B, CP-84, CG-84
Features	• Software compatibility with HD64180R/Z • 2-Channel Serial Interface	• Software compatibility with HD64180R/Z • No internal ROM (Romless)	• Software compatibility with HD64180R/Z • 16K byte Mask ROM	• Software compatibility with HD64180R/Z • 16K byte PROM

***Wide Temperature Range (—40~+85°C) version is available.**
CP/M® is the registered trade mark of Digital Research Inc.

■ NMOS 16-BIT MICROPROCESSOR

Type No.	HD68000-8 HD68000-10 HD68000-12	HD68000Y8 HD68000Y10 HD68000Y12	HD68000P8	HD68000PS8	HD68000CP8

Clock Frequency (MHz)	8.0(HD68000-8)
	10.0(HD68000-10)
	12.5(HD68000-12)
		8.0(HD68000Y-8)
		10.0(HD68000Y-10)
		12.5(HD68000Y-12)
			8.0(HD68000P-8)	8.0(HD68000PS-8)	8.0(HD68000CP-8)

Supply Voltage (V)	5.0				
Operating Temperature (°C)	0 ~ +70				
Power Dissipation (W)	1.5 (f = 6MHz, 8MHz, 10MHz), 1.75 (f = 12.5 MHz)		0.9 (f = 8MHz)		
Package	DC-64	PGA-68	DP-64	DP-64S	CP-68
Feature	High performance MPU featuring 32-bit data processing function				
Compatibility	MC68000L6 MC68000L8 MC68000L10 MC68000L12	MC68000R6 MC68000R8 MC68000R10 MC68000R12	MC68000P6 MC68000P8		MC68000FN6 MC68000FN8

■ CMOS 16-BIT MICROPROCESSOR

Type No.	HD68HC000-8 HD68HC000-10 HD68HC000-12	HD68HC000Y-8 HD68HC000Y-10 HD68HC000Y-12	HD68HC000P-8 HD68HC000P10 HD68HC000P-12	HD68HC000PS-8 HD68HC000PS-10 HD68HC000PS-12	HD68HC000CP-8 HD68HC000CP-10 HD68HC000CP-12

Clock Frequency (MHz)	8.0(HD68HC000-8)
	10.0(HD68HC000-10)
	12.5(HD68HC000-12)
		8.0(HD68HC000Y-8)
		10.0(HD68HC000Y10)
		12.5(HD68HC000Y12)
			8.0(HD68HC000P-8)
			10.0(HD68HC000P-10)
			12.5(HD68HC000P-12)
				8.0(HD68HC000PS-8)
				10.0(HD68HC000PS-10)
				12.5(HD68HC000PS-12)
					8.0(HD68HC000CP-8)
					10.0(HD68HC000CP-10)
					12.5(HD68HC000CP-12)

Supply Voltage (V)	5.0				
Operating Temperature (°C)	0 ~ +70				
Current Dissipation (mA)	25 (f = 8 MHz) 30 (f = 10 MHz) 35 (f = 12.5 MHz)				
Package	DC-64	PGA-68	DP-64	DP-64S	CP-68
Feature	High performance MPU featuring 32-bit data processing function				
Compatibility	MC68HC000L8 MC68HC000L10 MC68HC000L12	MC68HC000R8 MC68HC000R10 MC68HC000R12	MC68HC000G8 MC68HC000G10 MC68HC000G12		MC68HC000FN8 MC68HC000FN10 MC68HC000FN12

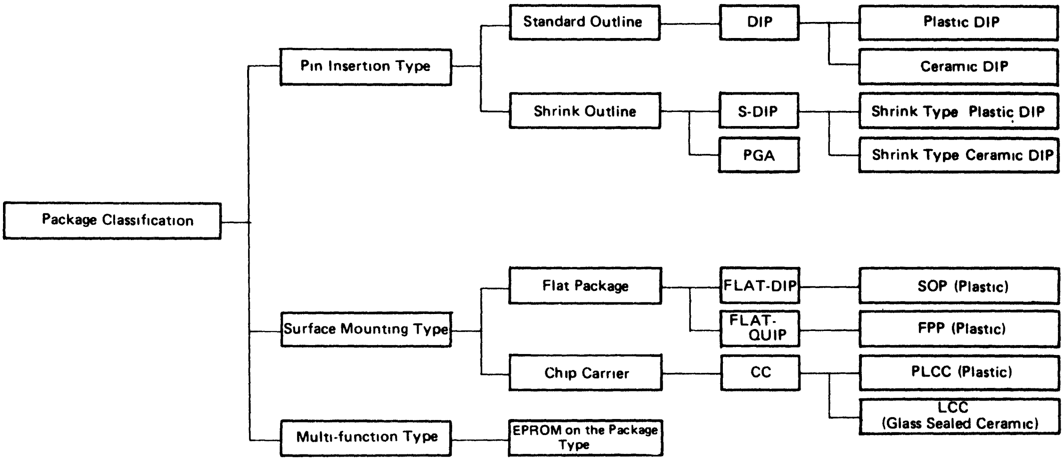
INTRODUCTION OF PACKAGES

Hitachi microcomputer devices include various types of package which meet a lot of requirements such as ever smaller, thinner and more versatile electric appliances. When selecting a package suitable for the customers' use, please refer to the following for Hitachi microcomputer packages.

1. Package Classification

There are pin insertion types, surface mounting types and multi-function types, applicable to each kind of mounting method. Also, plastic and ceramic materials are offered according to use.

Fig. 1 shows the package classification according to the mounting types on the Printed Circuit Board (PCB) and the materials.



DIP; DUAL IN LINE PACKAGE
S-DIP; SHRINK DUAL IN LINE PACKAGE
PGA PIN GRID ARRAY
FLAT-DIP; FLAT DUAL IN LINE PACKAGE
FLAT-QUIP, FLAT QUAD IN LINE PACKAGE
CC CHIP CARRIER
SOP; SMALL OUTLINE PACKAGE
FPP; FLAT PLASTIC PACKAGE
PLCC; PLASTIC LEADED CHIP CARRIER
LCC ; LEADLESS CHIP CARRIER

Fig. 1 Package Classification according to the Mounting Type on the Printed Circuit Board and the Materials.

2. Type No. and Package Code Indication

Type No. of Hitachi microprocessor is followed by package material and outline specifications, as shown below. The package type used for each device is identified by code as follows, illustrated in the data sheet of each device.

When ordering, please write the package code beside the type number.

Type No. Indication

HD××××P

(Note) The HD68000 with shrink type plastic DIP (DP-64S) has a different type No. from other devices.

Type No., HD68000PS8

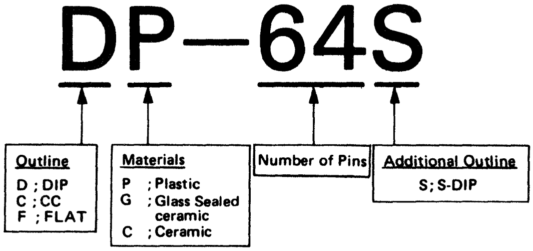
Package designation

Package Classification

No indication	:	Ceramic DIP
P	;	Plastic DIP
F	;	FPP
CP	;	PLCC
CG	;	LCC
Y	;	PGA (16-bit microcomputer device)

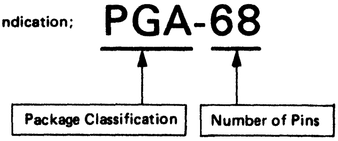
Introduction of Packages

Package Code Indication



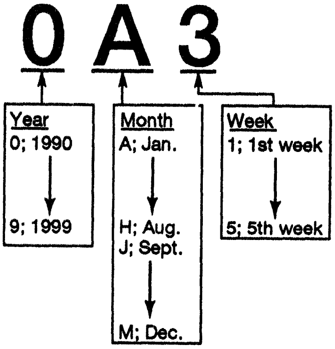
(Note) PGA packages of 16-bit microcomputer devices have a different indication.

Package Code Indication;



Date Code Indication

Assembly lot date code.



3. Package Dimensional Outline

Hitachi microprocessor employs the packages shown in Table 1 according to the mounting method on the PCB.

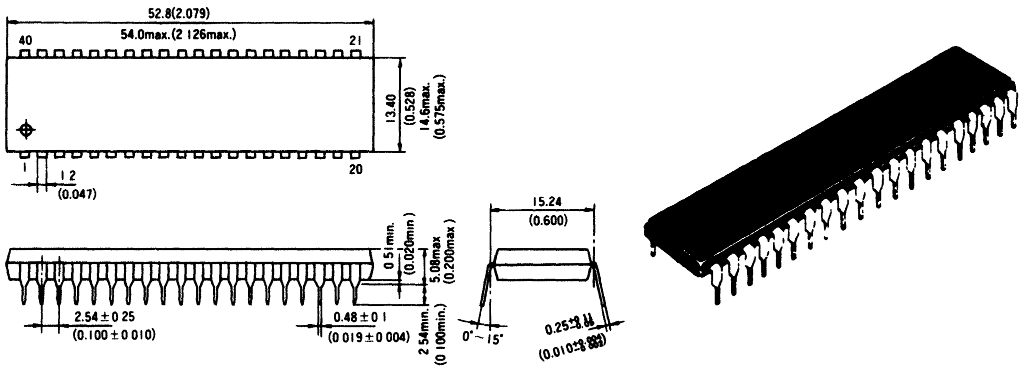
Table 1 Package List

Method of Mounting	Package Classification		Package Material	Package Code
Pin Insertion Type	Standard Outline (DIP)		Plastic	DP-40 DP-64
			Ceramic	DC-64
	Shrink Outline	S-DIP	Plastic	DP-64S DP-90S
		PGA	Glass Sealed Ceramic	PGA-68
Surface Mounting Type	Flat Package	FLAT-QUIP (FPP)	Plastic	FP-54 FP-64 FP-80 FP80B
	Chip-Carrier	PLCC	Plastic	CP-44 CP-52 CP-68 CP-84
		LCC	Glass Sealed Ceramic	CG-40 CG-84

Plastic DIP

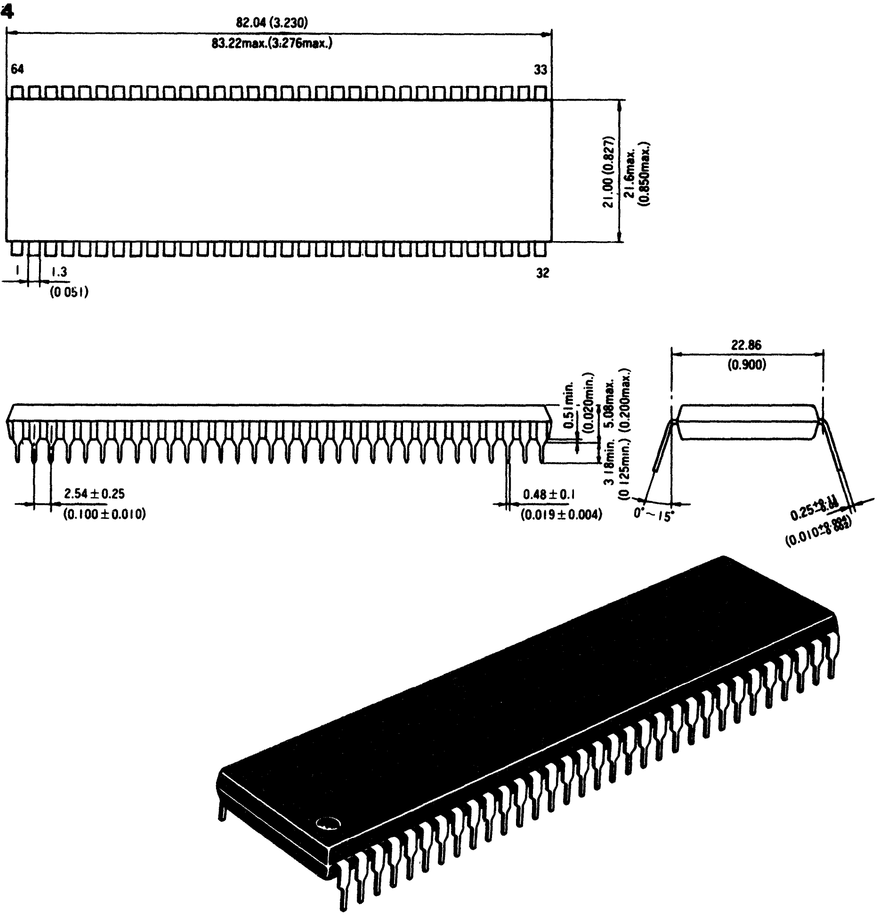
Unit : mm(inch)

- **DP-40**



Unit : mm(inch)

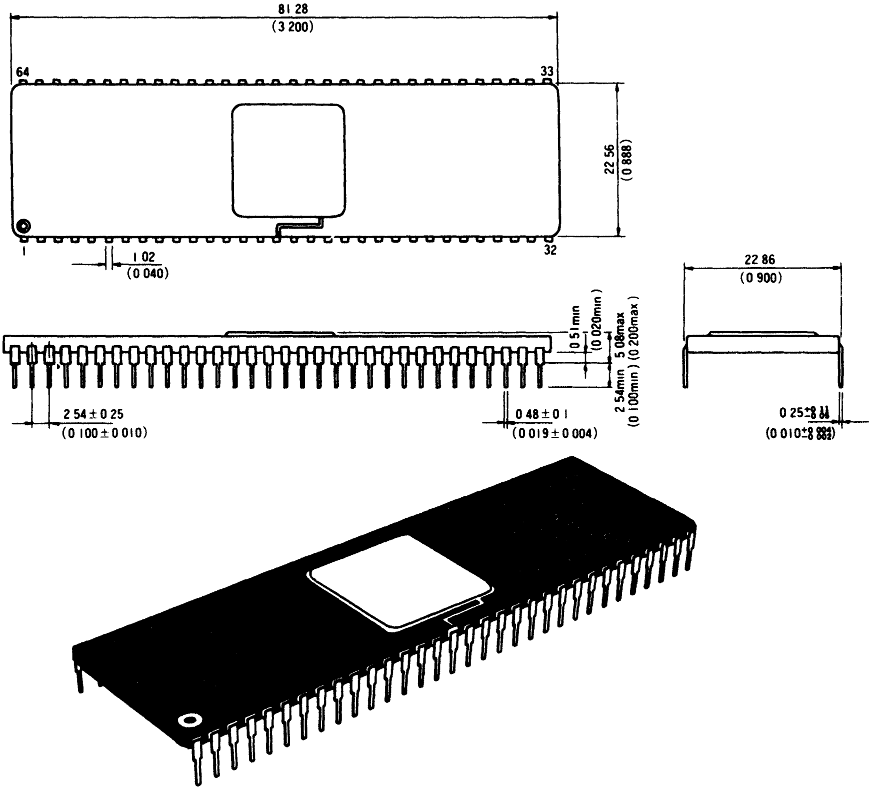
● DP-64



Ceramic DIP

Unit : mm(inch)

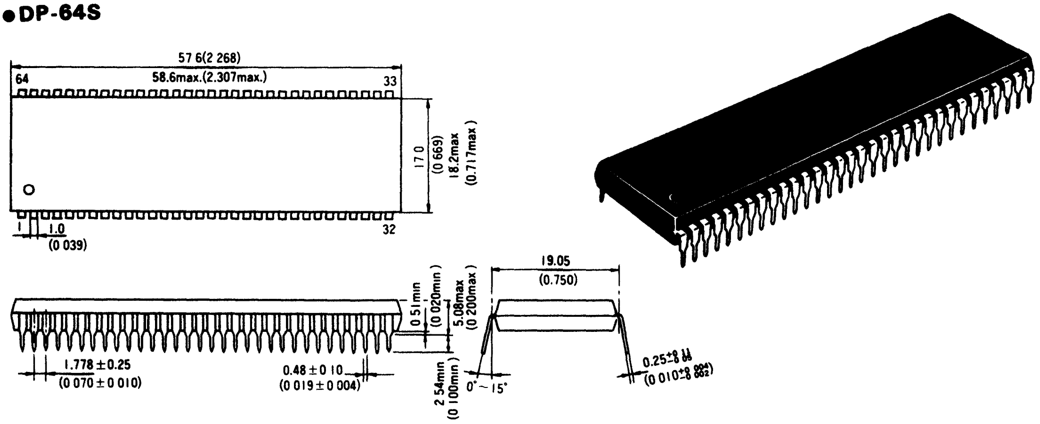
●DC-64



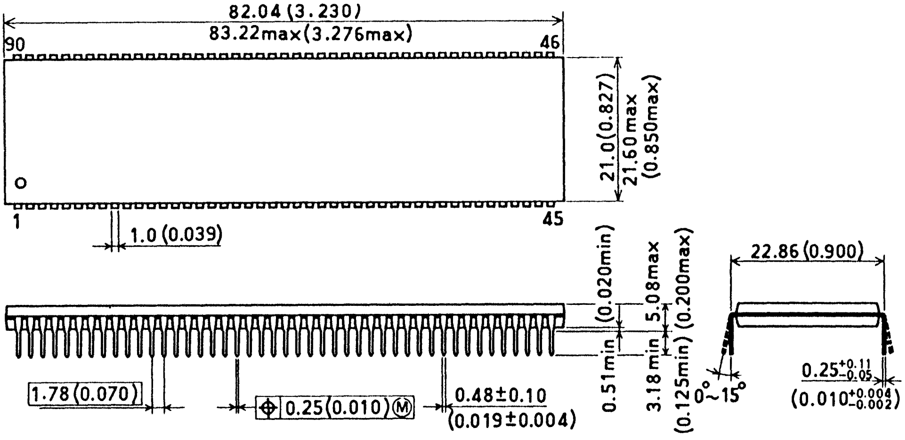
Shrink Type Plastic DIP

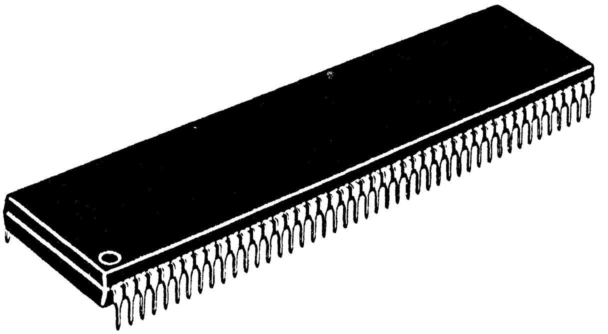
Unit: mm(inch)

●DP-64S



- **DP-90S**

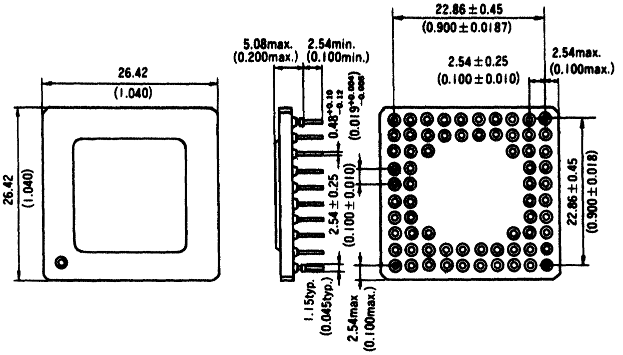


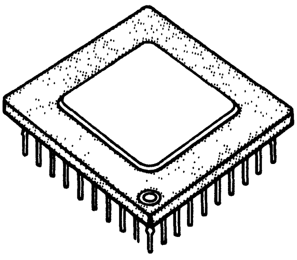


Pin Grid Array

Unit: mm(inch)

- **PGA-68**



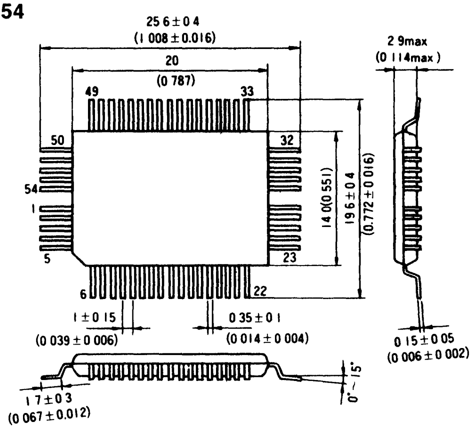


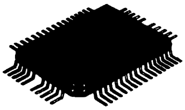
HITACHI

Flat Package

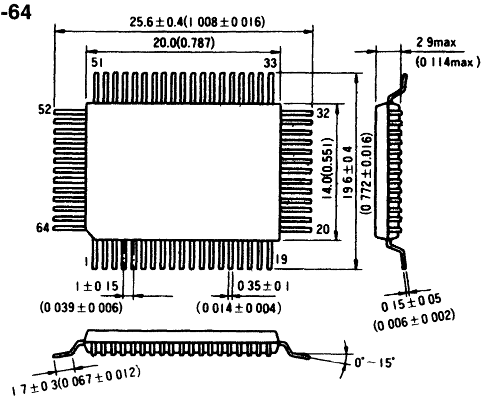
Unit: mm(inch)

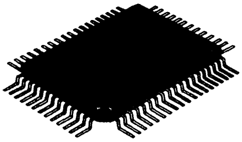
- **FP-54**



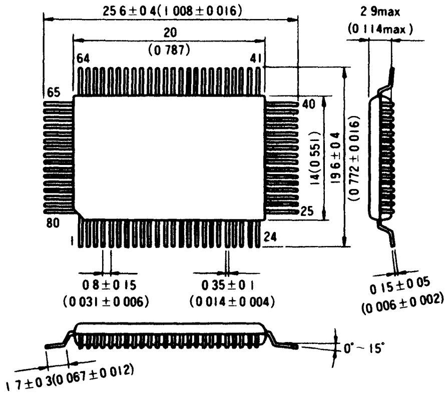


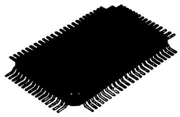
- **FP-64**



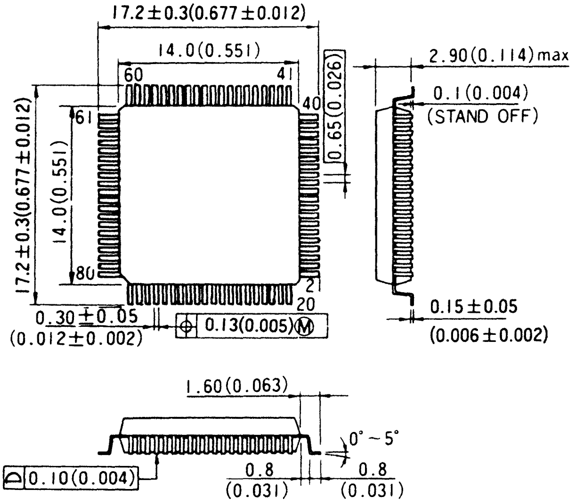


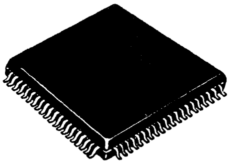
- **FP-80**



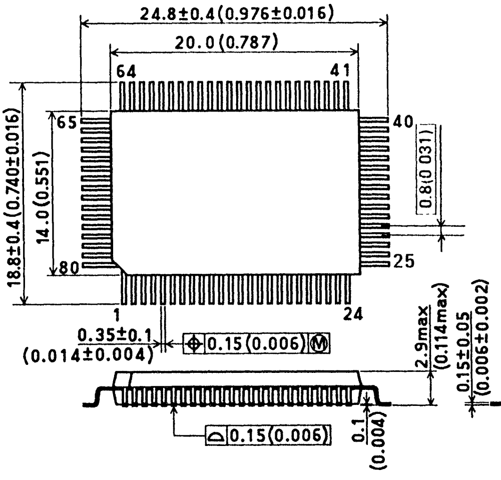


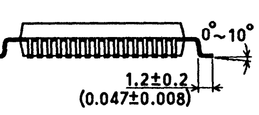
- **FP-80A**

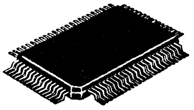

17.2±0.3(0.677±0.012)
14.0(0.551)
60
41
40
61
80
21
20
0.65(0.026)
17.2±0.3(0.677±0.012)
14.0(0.551)
0.30±0.05
(0.012±0.002)
⊕ 0.13(0.005)Ⓜ
2.90(0.114)max
0.1(0.004)
(STAND OFF)
0.15±0.05
(0.006±0.002)
1.60(0.063)
0°～5°
⏥ 0.10(0.004)
0.8
(0.031)
0.8
(0.031)



- **FP-80B**


24.8±0.4(0.976±0.016)
20.0(0.787)
64
41
40
65
80
25
1
24
18.8±0.4(0.740±0.016)
14.0(0.551)
0.8(0.031)
0.35±0.1
(0.014±0.004)
⊕ 0.15(0.006) Ⓜ
2.9max
(0.114max)
0.15±0.05
(0.006±0.002)
⏥ 0.15(0.006)
0.1
(0.004)

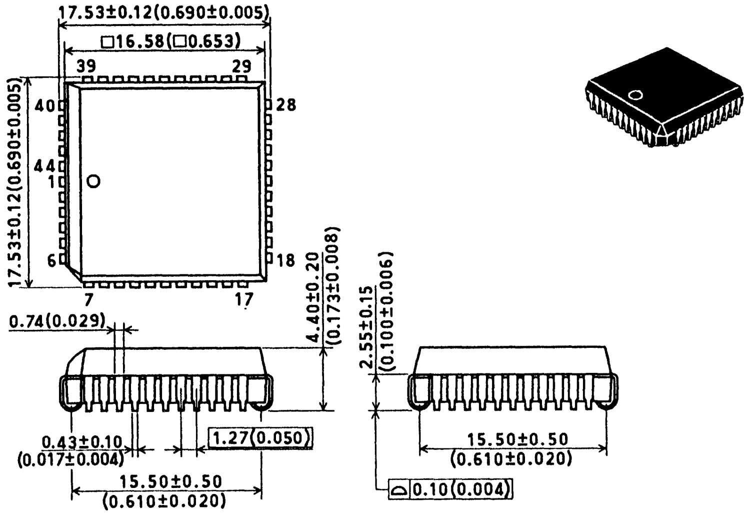

0°～10°
1.2±0.2
(0.047±0.008)



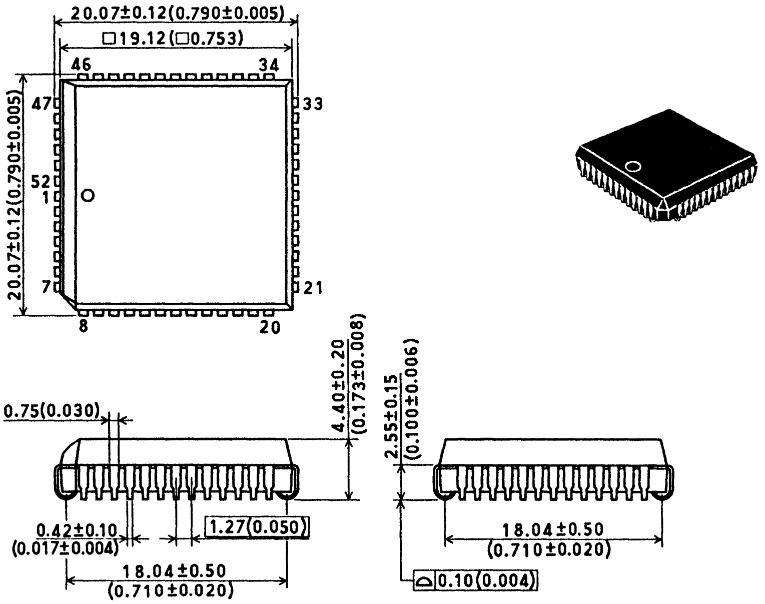
Plastic Leaded Chip Carrier

Unit: mm(inch)

- **CP-44**

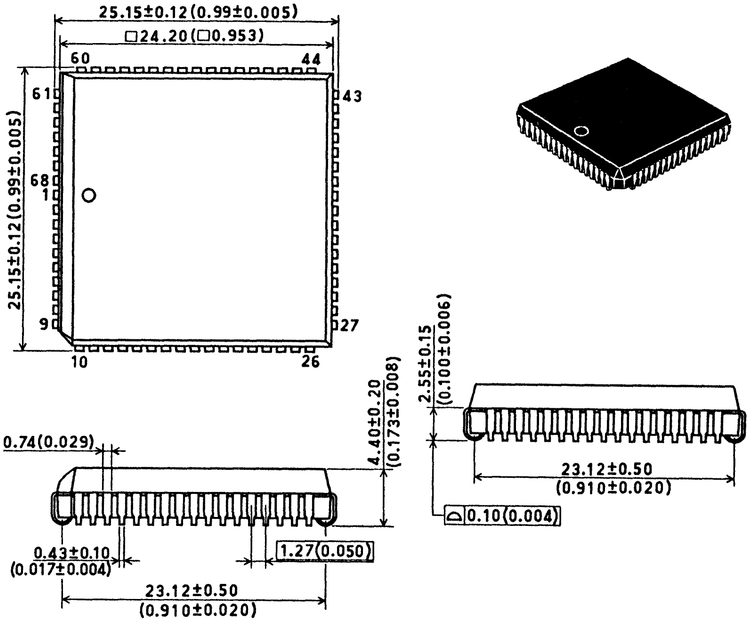


- **CP-52**

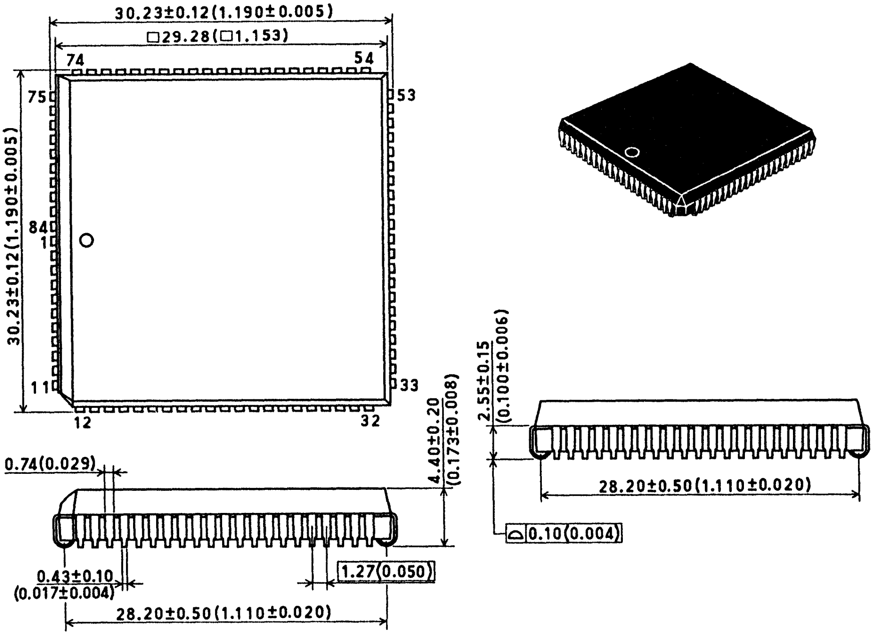


Unit: mm(inch)

- **CP-68**



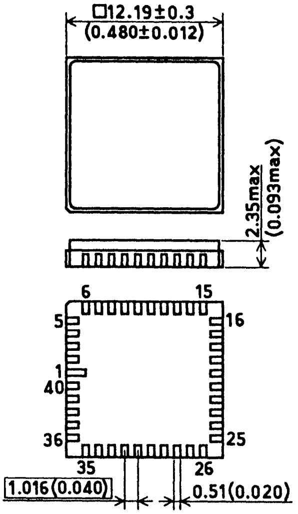
- **CP-84**

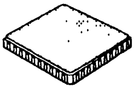


Leadless Chip Carrier

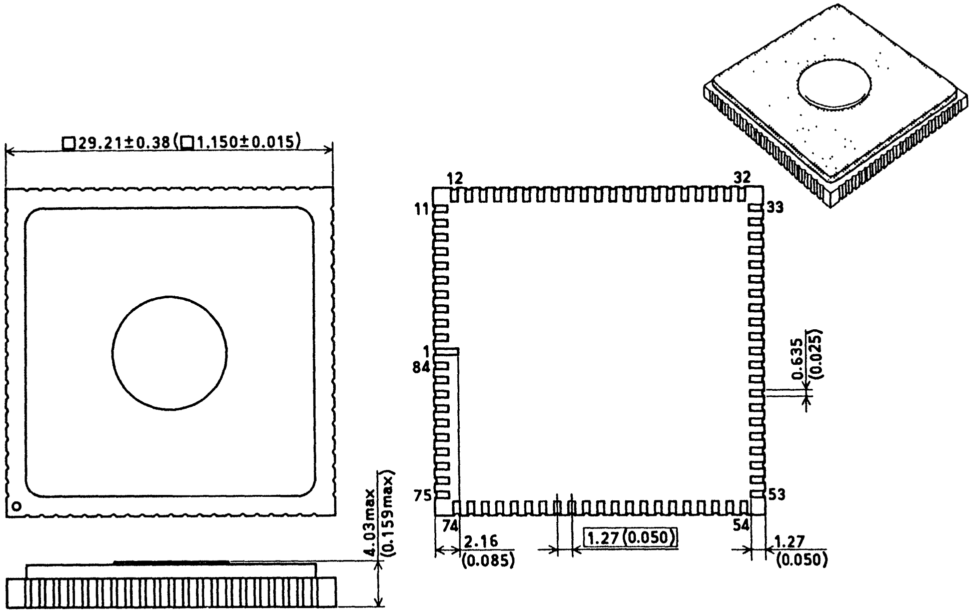
Unit: mm(inch)

- **CG-40**





- **CG-84**



4. Mounting Method on Board

Lead pins of the package have surface treatment, such as solder coating or solder plating, to make them easy to mount on the PCB. The lead pins are connected to the package by eutectic solder. The following explains the common connecting method of leads and precautions.

4.1 Mounting Method of Pin Insertion Type Package

Insert lead pins of the package into through-holes (usually about ϕ0.8mm) on the PCB. Soak the lead part of the package in a wave solder tub.

Lead pins of the package are held by the through-holes. Therefore, it is easy to handle the package through the process up to soldering, and easy to automate the soldering process. When soldering the lead part of the package in the wave solder tub , be careful not to get the solder on the package, because the wave solder will damage it.

4.2 Mounting Method of Surface Mounting Type Package

Apply the specified quantity of solder paste to the pattern on any printed board by the screen printing method, and put a package on it. The package is now temporarily fixed to the printed board by the surface tension of the paste. The solder paste melts when heated in a reflowing furnace, and the leads of the package and the pattern of the printed board are fixed together by the surface tension of the melted solder and the self alignment.

The size of the pattern where the leads are attached, partly depending on paste material or furnace adjustment, should be 1.1 to 1.3 times the leads' width.

The temperature of the reflowing furnace depends on package material and also package types. Fig. 2 lists the adjustment of the reflowing furnace for FPP. Pre-heat the furnace to 150°C. The surface temperature of the resin should be kept at 235°C max. for 10 minutes or less.

(1) The temperature of the leads should be kept at 260°C for 10 minutes or less.
(2) The temperature of the resin should be kept at 235°C for 10 minutes or less.
(3) Below is shown the temperature profile when soldering a package by the reflowing method.

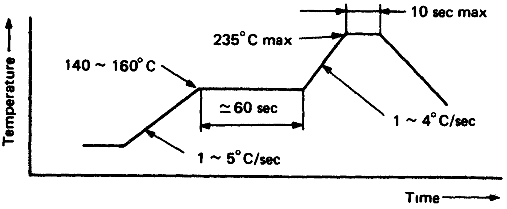


Figure 2 Reflowing Furnace Adjustment for FPP

Ensure good heater or temperature controls because the material of a plastic package is black epoxy-resin which damages easily. When an infrared heater is used, if the temperature is higher than the glass transition point of epoxy-resin (about 150°C), for a long time, the package may be damaged and the reliability lowered. Equalize the temperature inside and outside the packages by lessening the heat of the upper surface of the packages.

Leads of FPP may be easily bent under shipment or during handling and cannot be soldered onto the printed board. If they are, heat the bent leads again with a soldering iron to reshape them.

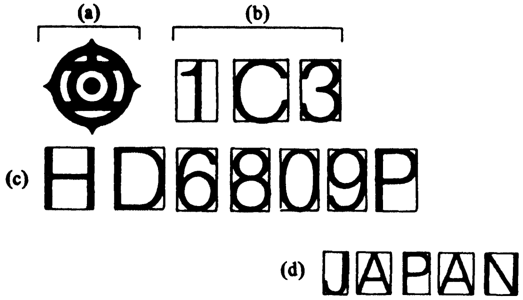
Use a rosin flux when soldering. Don't use a chloric flux because the chlorine in the flux tends to remain on the leads and lower the reliability of the product.

Even if you use a rosin flux, remaining flux can cause the leads to deteriorate. Wash away flux from packages with alcohol, chlorothene or freon. But don't leave these solvents on the packages for a long time because the marking may disappear.

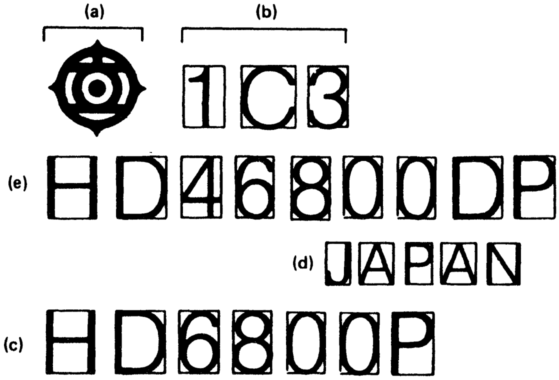
5. Marking

Hitachi trademark, product type No., etc. are printed on packages. Case I and Case II give examples of marks and Nos. Case I applies to products which have only a standard type No. Case II applies to products which have an old type No. and a standard type No.

Case I; Includes a standard type No.



Case II; Includes an old type No. and a standard type No.



Meaning of Each Mark

(a)	Hitachi Trademark
(b)	Lot Code
(c)	Standard Type No.
(d)	Japan Mark
(e)	Old Type No.

SOCKETS FOR EVALUATING SURFACE AND THROUGH-HOLE MOUNT PACKAGES

1. SOCKET LIST

Table 1 lists the sockets available on the market for evaluating the characteristics of surface and through-hole package devices. For details, please inquire directly to the socket manufacturer.

Table 1 IC Socket List

Package Type	Package Code	Socket Code	Manufacturer Name
QFP	FP-54	IC51-0544-517-2	Yamaichi Denki
	FP-64	IC51-0644-472-2 FPQ-64-1.0-08A	Yamaichi Denki Enplas
	FP-64A	IC51-0644-692-3 FPQ-64-0.8-01A	Yamaichi Denki Enplas
	FP-80	IC51-0804-394-2 FPQ-80-0.8-11A	Yamaichi Denki Enplas
	FP-80B	IC51-0804-819-1 FPQ-80-0.8-11A FPQ-80-0.8-13A	Yamaichi Denki Kogyo Enplas Enplas
PLCC	CP-44	IC51-0444-400	Yamaichi Denki Kogyo K.K.
	CP-52	IC51-0524-411	Yamaichi Denki Kogyo K.K.
	CP-68	IC51-0684-390 PLCC-68-1.27-02	Yamaichi Denki Kogyo K.K. Enplas
	CP-84	PC1-084050-003 IC51-0844-401-1	Nepenthe Yamaichi Denki Kogyo
DIP (Plastic)	DP-40	IC37NR-4006-G4	Yamaichi Denki
	DP-64	IC8620-6409-G4 IC86-6409	Yamaichi Denki Kogyo
DIP (Plastic, Shrink)	DP-64S	IC7620-64075-G4 IC38-64075-G4	Yamaichi Denki Yamaichi Denki
	DP-90S	IC121-9009-G4	Yamaichi Denki Kogyo
Dip (Ceramic)	DC-64	IC8620-6409-G4	Yamaichi Denki Kogyo
PGA (Glass Sealed Ceramic)	PC-68	PPS68-AG2D	AUGAT
LCC (Glass Sealed Ceramic)	CG-40	240-5084-00-1102	TEXTOOL
	CG-84	PC1-084050-003 SHIM-0844-401-047 IC51-0844-401-1	Nepenthe Nepenthe Yamaichi Denki Kogyo

DEVICE PACKING

1. SHIPPING CONTAINERS AND HANDLING

1.1 SHIPPING CONTAINER FORMS

Figures 1 and 2 illustrate the shipping container forms for ordinary IC devices. Within the outer corrugated cardboard carton there are one or more inner cartons. These inner carton contain magazines, trays, or tape reels, which the IC devices are shipped in.

Plastic surface mount packages containing large chips can crack if they absorb moisture and are mounted by reflow soldering. These surface mount devices are packed with a moisture-proof material to prevent the packages from absorbing moisture during shipping and storage.

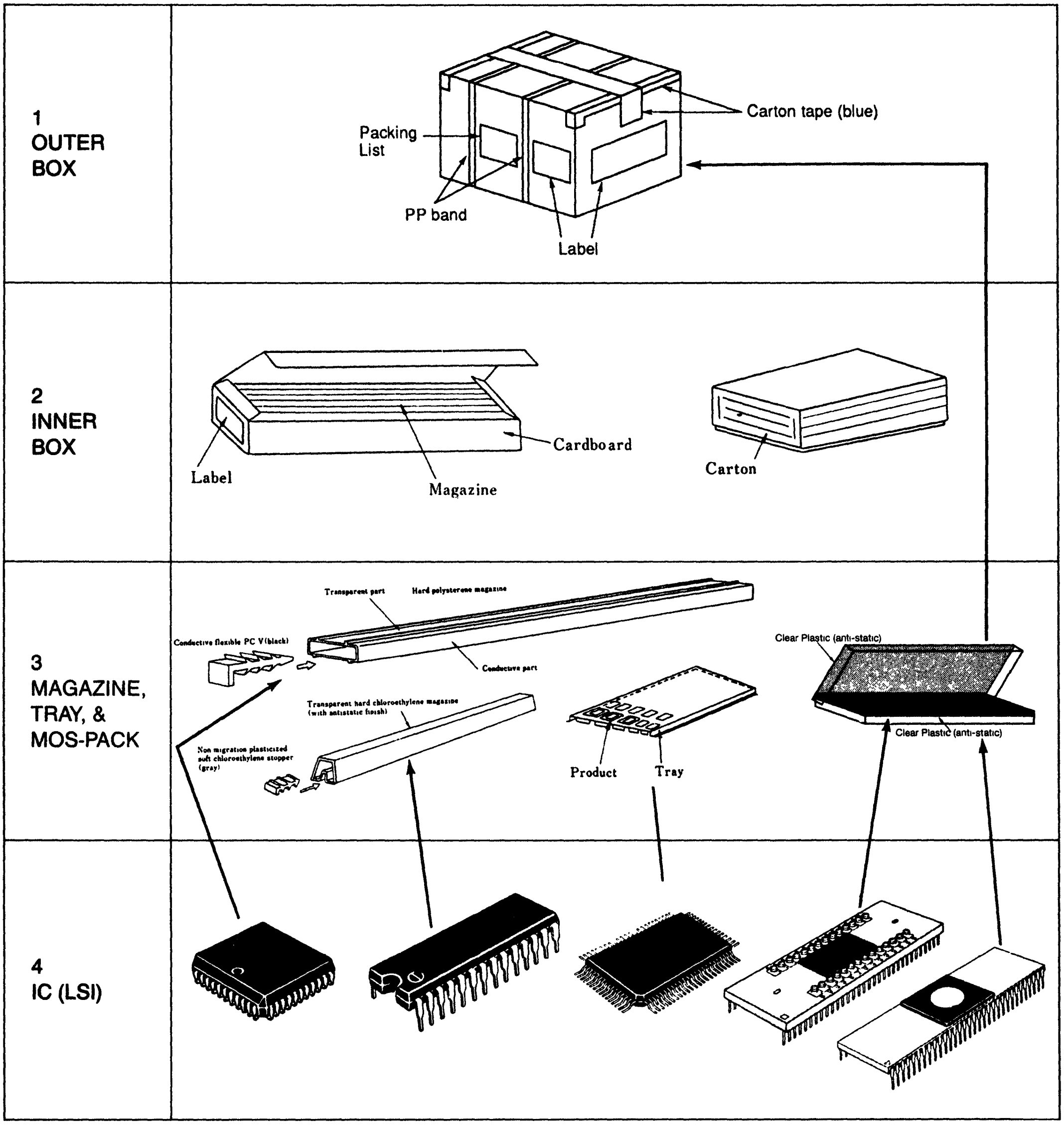

1
OUTER
BOX
Packing List
Carton tape (blue)
PP band
Label
2
INNER
BOX
Cardboard
Label
Magazine
Carton
3
MAGAZINE,
TRAY, &
MOS-PACK
Transparent part
Hard polysterene magazine
Conductive flexible PC V(black)
Conductive part
Transparent hard chloroethylene magazine (with antistatic finish)
Non migration plasticized soft chloroethylene stopper (gray)
Product
Tray
Clear Plastic (anti-static)
Clear Plastic (anti-static)
4
IC (LSI)

Figure 1. Shipping Containers

1.2 NOTES ON HANDLING

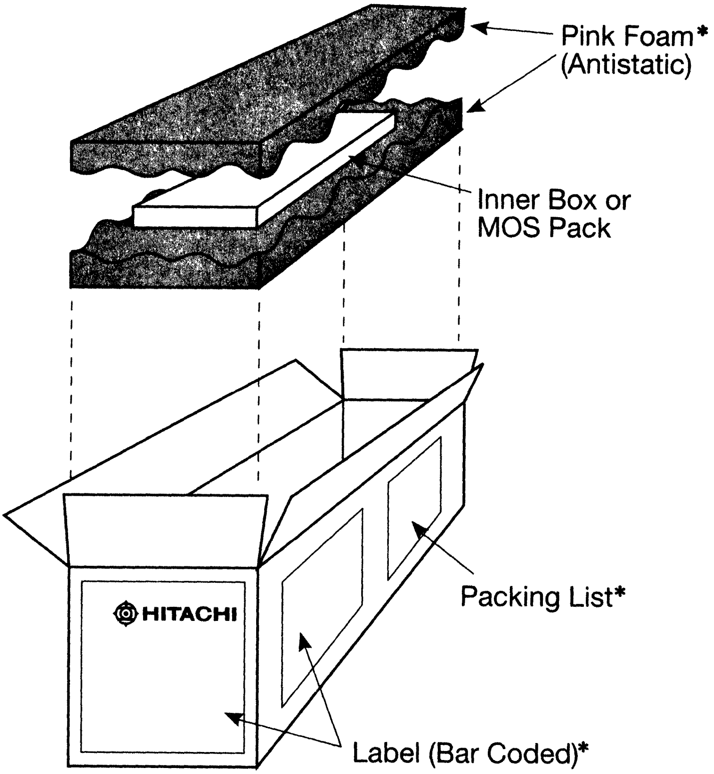
(1) Handles the outer cardboard carton with care. Sudden drops or shocks can cause damage to the enclosed products. Be sure not to overstack the cartons.

(2) Prevent water leakage. Do not leave shipping containers outside or store them in high-temperature, high-humidity areas.

(3) Handle the inner cartons with care. Dropping a box may dislodge a magazine stopper, allowing devices to slide out in which care their leads may be deformed. Dropping may also cause damage to ceramic packages and cause leaks to air-tight seals. The surface of transparent vinyl-chloride magazines are treated with an antistatic coating to prevent static charge. Be aware of the following notes concerning this coating:

- Water leakage will cause the anti-static material to peel off and lose it effectiveness.
- The anti-static material may become sticky in high-temperature, high-humidity environments.
- The anti-static material may warp over time; avoid storage beyond six months. Do not reuse the material.
- Note that the surface resistance of transparent magazines is less than 1×10^{10} Ohms, and the surface resistance of black magazines is less than 1×10^{6} Ohms.
- Store vinyl-chloride trays between -25°C and +40°C. Both the shape and color may change in an environment above 55°C.

1.3 PARTIAL SHIPMENTS PACKING



DIMENSION IN MM W x H x L	DIMENSION IN INCHES W x H x L
250 x 250 x 500	10 x 10 x 20
250 x 225 x 550	10 x 9 x 22
175 x 150 x 550	7 x 6 x 22

* Materials or placement may vary.

Figure 2. Partial Shipments

2. MOISTURE-PROOF (DRY PACK) PACKING AND HANDLING

If a surface mount package is mounted with solder reflow after it has absorbed moisture, then package cracks may occur. In order to prevent moisture absorption during shipping or storage, the packages are encased ın vacuum packed moisture-proof (dry pack) packing material as shown in figure 3 The following sections describe how to handle this material.

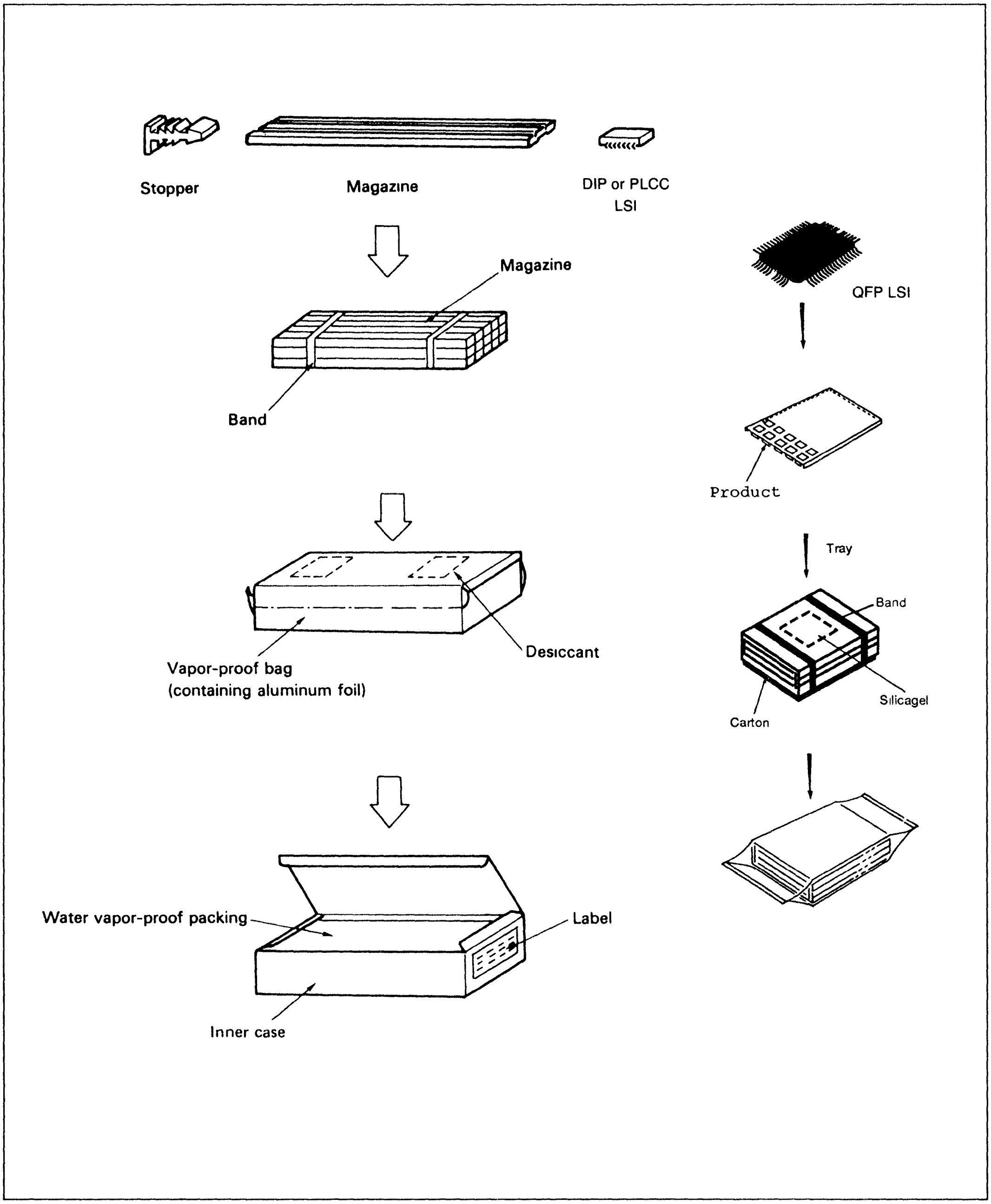


Fıgure 3. Vacuum Packed Moısture-Proof (Dry Pack) Packıng

2.1 STORAGE METHOD:

Storing packed ICs under inappropriate conditions can cause deterioration in solderability and performance. Hitachi recommends that products in vacuum packed moisture-proof (dry pack) packing material be stored in tray boxed If this is not possible, packages should be stored under the following conditions:

- Temperature· 5 to 30°C
- Humidity less than 60% RH

Parts stored in unopened vacuum packed moisture-proof (dry pack) condition may remain solderable for three (3) to five (5) years.

2.2 HANDLING AFTER OPENING:

In order to prevent re-absorption after opening the moisture-proof material, store under the conditions listed above and reflow mount the packages within one week If the packages must be placed into storage again after opening, then seal in a new (non-moisture contaminated) silica gel (confirm with blue-colored indicator) and store under the conditions listed above Try to reseal in vacuum packed moisture-proof (dry pack) packing material

2.3 BAKING BEFORE SOLDER REFLOW:

Baking is necessary if the indicator of the silica gel does not appear blue-colored throughout; more than one week has elapsed since opening (even stored under the conditions listed above); or the affixed label indicates baking is required.

2.4 RECOMMENDED BAKING CONDITIONS:

Baking should be performed under the following conditions:

- Temperature. 125°C
- Duration: 16 to 24 hours

The magazines, trays, and tape reels normally used for shipment are not heat-proof, therefore containers cannot be baked as shipped. Devices must first be transferred into a heat-proof container Heat-proof magazines and trays are currently under development.

Tray labelled as heat-proof can be used, however do not bake with the moisture-proof bag Bake on a level plane to prevent sliding.

3. PACKING SPECIFICATIONS FOR VARIOUS PACKAGES

3.1 PACKING SPECIFICATIONS for DIP Packages

Package Code (corresponding diagram)	Illustration in figure 4(b)	Quantity			Inner Box* Dimensions W x H x L in mm. and (inches)
		ICs/Magazine	IC/MOS Pack	Magazines/Inner Box	
DP-40	(A or B)	9	—	20	112.5 x 59.4 x 500 (4½ x 2⅜ x 20)
DP-64					
(DP-64S)	(C)	8	—	12	75 x 59.4 x 500 (3 x 2⅜ x 20)
DP-90S					
DC-64	(D)	—	10	N/A	80 x 16 x 240 (3⅛ x ⅝ x 9⅝)

Figure 4(a). Packing Specifications for DIP Packages *(see Fig 1, this section)

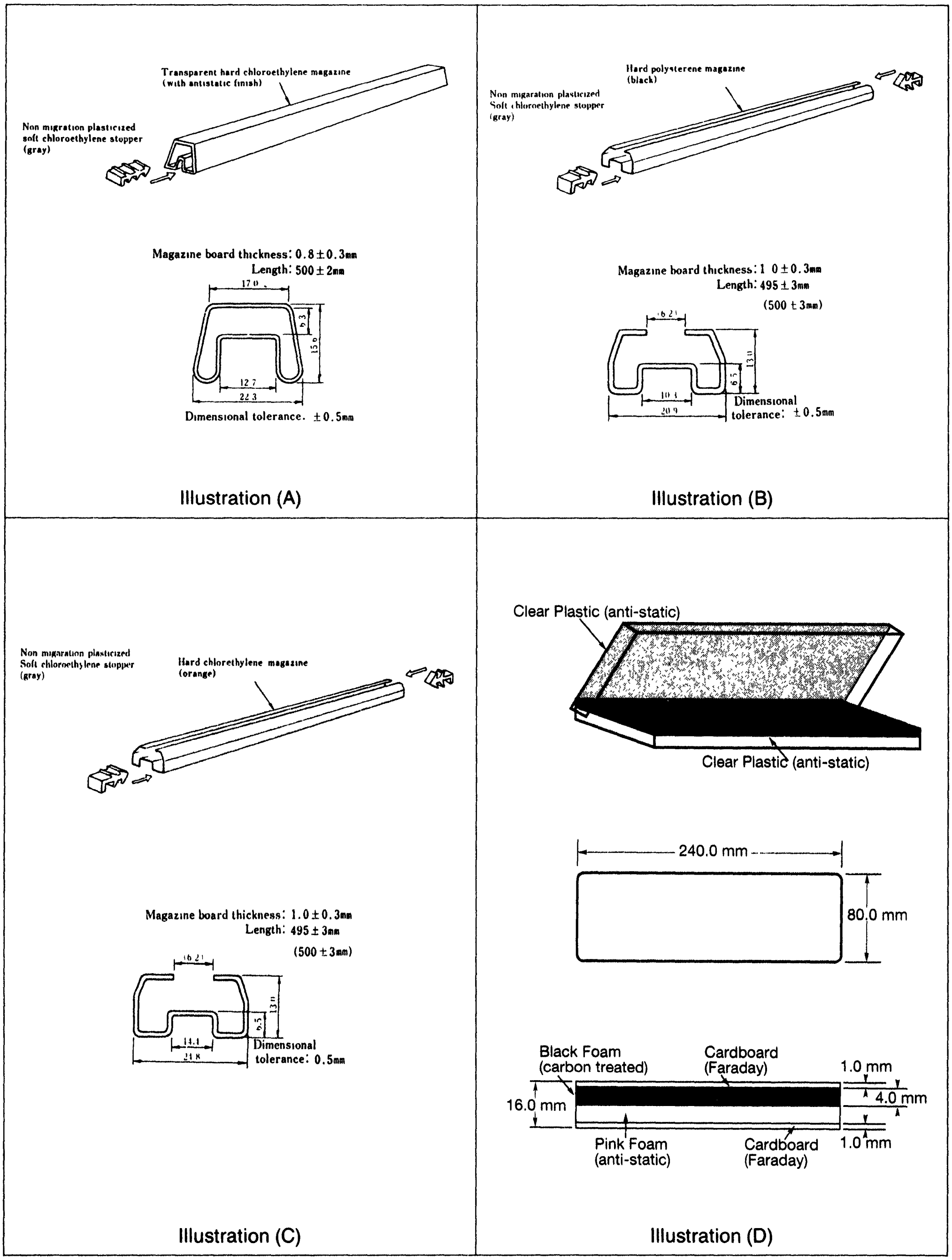

Transparent hard chloroethylene magazine (with antistatic finish)
Non migration plasticized soft chloroethylene stopper (gray)
Magazine board thickness: 0.8 ± 0.3mm
Length: 500 ± 2mm
17 0
6 3
15 6
12 7
22 3
Dimensional tolerance. ± 0.5mm
Hard polysterene magazine (black)
Non migaration plasticized Soft chloroethylene stopper (gray)
Magazine board thickness: 1 0 ± 0.3mm
Length: 495 ± 3mm
(500 ± 3mm)
(6 2)
13 0
6 5
10 3
20 9
Dimensional tolerance: ± 0.5mm
Non migaration plasticized Soft chloroethylene stopper (gray)
Hard chlorethylene magazine (orange)
Magazine board thickness: 1.0 ± 0.3mm
Length: 495 ± 3mm
(500 ± 3mm)
(6 2)
13 0
6.5
14.1
21 8
Dimensional tolerance: 0.5mm
Clear Plastic (anti-static)
Clear Plastic (anti-static)
240.0 mm
80.0 mm
Black Foam (carbon treated)
Cardboard (Faraday)
1.0 mm
4.0 mm
16.0 mm
Pink Foam (anti-static)
Cardboard (Faraday)
1.0 mm

Illustration (A)

Illustration (B)

Illustration (C)

Illustration (D)

Figure 4(b). Packing Materials for DIP Packages

3.2 QFP AND LCC PACKING SPECIFICATION

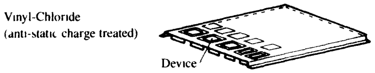


Illustration (A)†

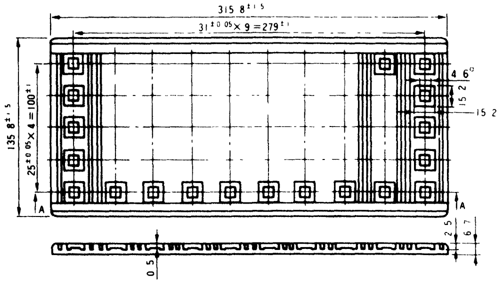


Illustration (B)†

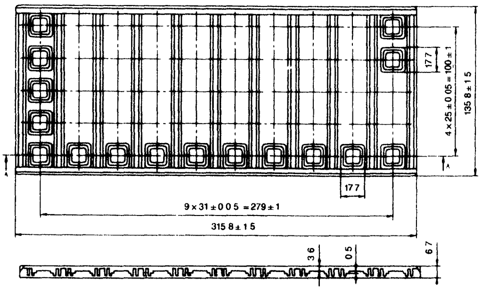


Illustration (C)†

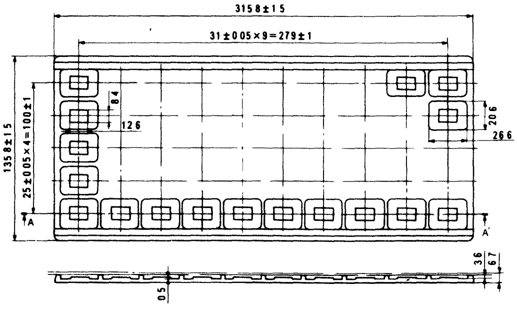


Illustration (D)‡

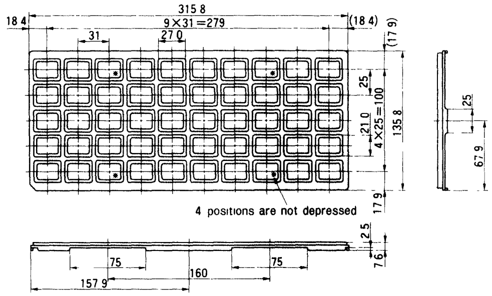
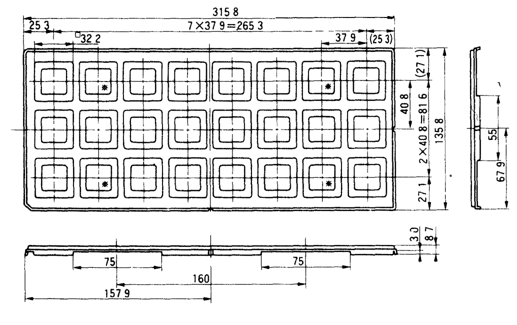


Illustration (E)‡

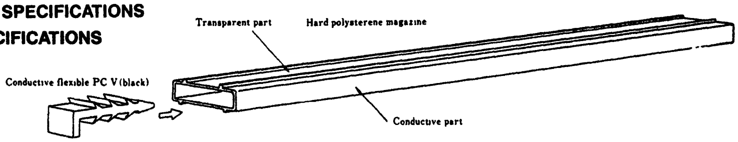


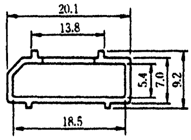
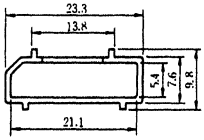
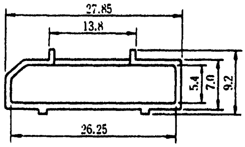
†without holes ‡with holes

Package Code	Illustration in Figure 5	Quantity		Inner Box* Dimensions W × H × L in mm. (in.)
		IC/Tray	Trays/Inner Box	
FP-54	(C or D)	50	10	147 × 70 × 325 (5¾ × 2¾ × 13)
FP-64	(C or D)	50	10	147 × 70 × 325 (5¾ × 2¾ × 13)
FP-64A	(B)	50	10	147 × 70 × 325 (5¾ × 2¾ × 13)
FP-80	(C or D)	50	10	147 × 70 × 325 (5¾ × 2¾ × 13)
FP-80B	D	50	10	147 × 70 × 325 (5¾ × 2¾ × 13)
CG-40	(A)	50	10	147 × 70 × 325 (5¾ × 2¾ × 13)

Figure 5. QFP and LCC Packing Specifications and Materials *(see Fig. 1, this section)

3.3 PACKAGE PACK SPECIFICATIONS PLCC PACKING SPECIFICATIONS

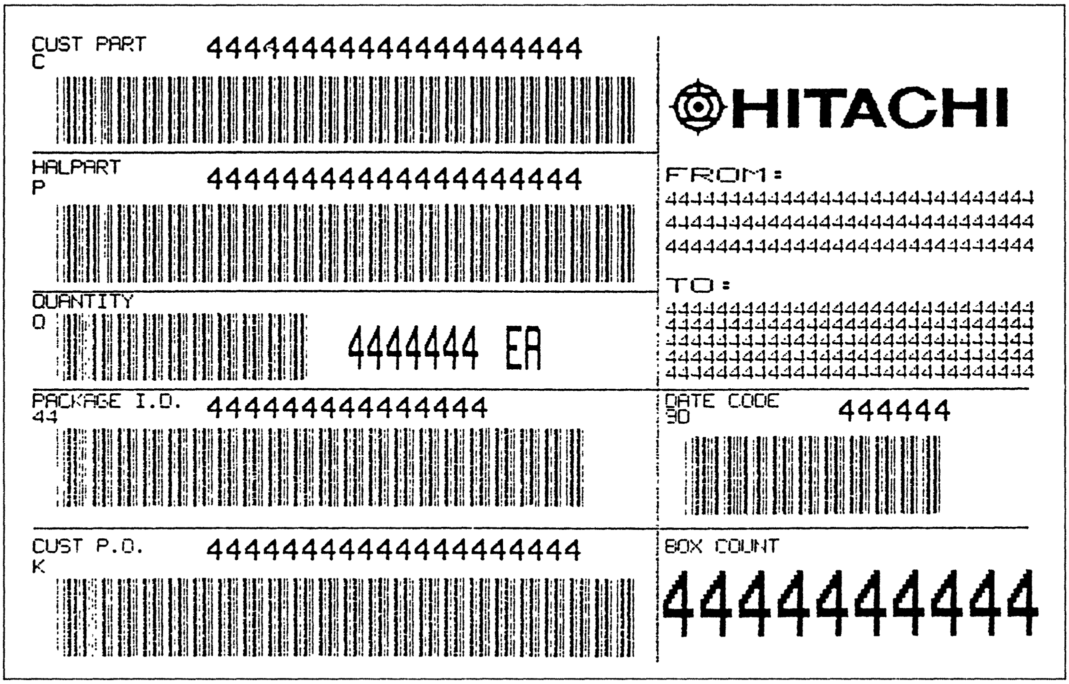


Magazine board thickness: 0.8 ± 0.3 mm length: 495 ± 3 mm	Magazine board thickness: 1.1 ± 0.3 mm length: 495 ± 3 mm	Magazine board thickness: 0.8 ± 0.3 mm length: 495 ± 3 mm
		
Dimensional tolerance: ± 0.5 mm Illustration (A)	Dimensional tolerance: ± 0.5 mm Illustration (B)	Dimensional tolerance: ± 0.5 mm Illustration (C)

Package Code	Illustration in Figure 6	Quantity		Inner Box* Dimensions W × H × L in mm. (in.)
		ICs Magazine	Magazines/Inner Box	
CP-44	(A)	26	30	113 × 56.3 × 493.8 (4½ × 2¼ × 19¾)
CP-52	(B)	23	18	118.8 × 65.6 × 500 (4½ × 2⅝ × 20)
CP-68	(C)	18	28	118.8 × 65.6 × 493.8 (4½ × 2⅝ × 20)
CP-84	(D)	15		

Figure 6. PLCC Packing Specifications and Materials *(see Fig. 1, this section)

4.0 PACKING LABELS



CUST PART
C
44444444444444444444

HALPART
P
44444444444444444444

QUANTITY
Q
4444444 EA

PACKAGE I.D.
44
444444444444444

CUST P.O.
K
44444444444444444444

HITACHI

FROM:
444444444444444444444444444444
444444444444444444444444444444
444444444444444444444444444444

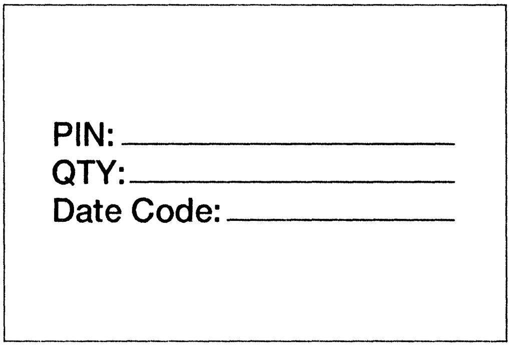
TO:
444444444444444444444444444444
444444444444444444444444444444
444444444444444444444444444444
444444444444444444444444444444
444444444444444444444444444444

DATE CODE
3D
444444

BOX COUNT
444444444

Figure 7. Outer Box Label

Figures 1 and 2 on pages 13 and 14 show the outer box label placement on the end and left adjacent side of the box. Placement is within one-half inch (1/2") of the box's corner. Figures 1 and 2 show the inner box label placement is on the end of the inner box. A packing list is afixed to the left adjacent side of the outer box's end and next to the bar code label.



PIN: ____________________

QTY: ____________________

Date Code: ________________

Figure 8. Inner Box Label

RELIABILITY AND QUALITY ASSURANCE

1. VIEWS ON QUALITY AND RELIABILITY

Basic views on quality in Hitachi are to meet individual user's purchase purpose and quality required, and to be at the satisfied quality level considering general marketability. Quality required by users is specifically clear if the contract specification is provided. If not, quality required is not always definite. In both cases, efforts are made to assure the reliability so that semiconductor devices delivered can perform their ability in actual operating circumstances. To realize such quality in manufacturing process, the key points should be to establish quality control system in the process and to enhance morale for quality.

In addition, quality required by users on semiconductor devices is going toward higher level as performance of electronic system in the market is going toward higher one and is expanding size and application fields. To cover the situation, actual bases Hitachi is performing is as follows;

(1) Build the reliability in design at the stage of new product development.
(2) Build the quality at the sources of manufacturing process.
(3) Execute the harder inspection and reliability confirmation of final products.
(4) Make quality level higher with field data feed back.
(5) Cooperate with research laboratories for higher quality and reliability.

With the views and methods mentioned above, utmost efforts are made for users' requirements.

2. RELIABILITY DESIGN OF SEMICONDUCTOR DEVICES

2.1 Reliability Targets

Reliability target is the important factor in manufacture and sales as well as performance and price. It is not practical to rate reliability target with failure rate at the certain common test condition. The reliability target is determined corresponding to character of equipments taking design, manufacture, inner process quality control, screening and test method, etc. into consideration, and considering operating circumstances of equipments the semiconductor device used in, reliability target of system, derating applied in design, operating condition, maintenance, etc.

2.2 Reliability Design

To achieve the reliability required based on reliability targets, timely sude and execution of design standardization, device design (including process design, structure design), design review, reliability test are essential.

(1) Design Standardization

Establishment of design rule, and standardization of parts, material and process are necessary. As for design rule, critical items on quality and reliability are always studied at circuit design, device design, layout design, etc. Therefore, as long as standardized process, material, etc. are used, reliability risk is extremely small even in new development devices, only except for in the case special requirements in function needed.

(2) Device Design

It is important for device design to consider total balance of process design, structure design, circuit and layout design. Especially in the case new process and new material are employed, technical study is deeply executed prior to device development.

(3) Reliability Evaluation by Test Site

Test site is sometimes called Test Pattern. It is useful method for design and process reliability evaluation of IC and LSI which have complicated functions.

1. Purposes of Test Site are as follows;
   - Making clear about fundamental failure mode
   - Analysis of relation between failure mode and manufacturing process condition
   - Search for failure mechanism analysis
   - Establishment of QC point in manufacturing
2. Effectiveness of evaluation by Test Site are as follows;
   - Common fundamental failure mode and failure mechanism in devices can be evaluated.
   - Factors dominating failure mode can be picked up, and comparison can be made with process having been experienced in field.
   - Able to analyze relation between failure causes and manufacturing factors.
   - Easy to run tests.

   etc.

2.3 Design Review

Design review is organized method to confirm that design satisfies the performance required including users' and design work follows the specified ways, and whether or not technical improved items accumulated in test data of individual major fields and field data are effectively built in. In addition, from the standpoint of enhancement of competitive power of products, the major purpose of design review is to ensure quality and reliability of the products. In Hitachi, design review is performed from the planning stage for new products and even for design changed products. Items discussed and determined at design review are as follows;

(1) Description of the products based on specified design documents.
(2) From the standpoint of specialty of individual participants, design documents are studied, and if unclear matter is found, sub-program of calculation, experiments, investigation, etc. will be carried out.
(3) Determine contents of reliability and methods, etc. based on design document and drawing.
(4) Check process ability of manufacturing line to achieve design goal.
(5) Discussion about preparation for production.
(6) Planning and execution of sub-programs for design change proposed by individual specialist, and for tests, experiments and calculation to confirm the design change.
(7) Reference of past failure experiences with similar devices, confirmation of method to prevent them, and planning and execution of test program for confirmation of them. These studies and decisions are made using check lists made individually depending on the objects.

3. QUALITY ASSURANCE SYSTEM OF SEMICONDUCTOR DEVICES

3.1 Activity of Quality Assurance

General views of overall quality assurance in Hitachi are as follows;

(1) Problems in individual process should be solved in the

process. Therefore, at final product stage, the potential failure factors have been already removed.

(2) Feedback of information should be made to ensure satisfied level of process ability.

(3) To assure reliability required as an result of the things mentioned above is the purpose of quality assurance.

The followings are regarding device design, quality approval at mass production, inner process quality control, product inspection and reliability tests.

3.2 Quality Approval

To ensure quality and reliability required, quality approval is carried out at trial production stage of device design and mass production stage based on reliability design described at section 2.

The views on quality approval are as follows;

(1) The third party performs approval objectively from the standpoint of customers.

(2) Fully consider past failure experiences and information from field.

(3) Approval is needed for design change and work change.

(4) Intensive approval is executed on parts material and process.

(5) Study process ability and fluctuation factor, and set up control points at mass production stage.

Considering the views mentioned above, quality approval shown in Fig. 1 is performed.

3.3 Quality and Reliability Control at Mass Production

For quality assurance of products in mass production, quality control is executed with organic division of functions in manufacturing department, quality assurance department, which are major, and other departments related. The total function flow is shown in Fig. 2. The main points are described below.

3.3.1 Quality Control of Parts and Material

As the performance and the reliability of semiconductor devices are getting higher, importance is increasing in quality control of material and parts, which are crystal, lead frame, fine wire for wire bonding, package, to build products, and materials needed in manufacturing process, which are mask pattern and chemicals. Besides quality approval on parts and materials stated in section 3.2, the incoming inspection is, also, key in quality control of parts and materials. The incoming inspection is performed based on incoming inspection specification following purchase specification and drawing, and sampling inspection is executed based on MIL-STD-105D mainly.

The other activities of quality assurance are as follows:

(1) Outside Vendor Technical Information Meeting

(2) Approval on outside vendors, and guidance of outside vendors

(3) Physical chemical analysis and test

The typical check points of parts and materials are shown in Table 1.

3.3.2 Inner Process Quality Control

Inner process quality control is performing very important function in quality assurance of semiconductor devices. The following is description about control of semi-final products, final products, manufacturing facilities, measuring equipments,

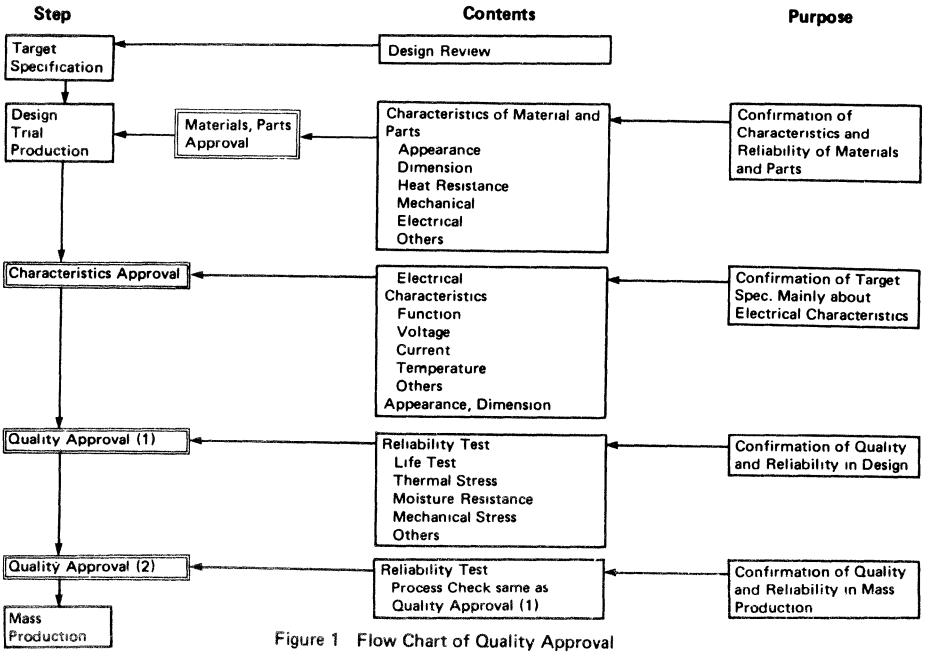

Step
Contents
Purpose
Target Specification
Design Review
Design Trial Production
Materials, Parts Approval
Characteristics of Material and Parts
Appearance
Dimension
Heat Resistance
Mechanical
Electrical
Others
Confirmation of Characteristics and Reliability of Materials and Parts
Characteristics Approval
Electrical Characteristics
Function
Voltage
Current
Temperature
Others
Appearance, Dimension
Confirmation of Target Spec. Mainly about Electrical Characteristics
Quality Approval (1)
Reliability Test
Life Test
Thermal Stress
Moisture Resistance
Mechanical Stress
Others
Confirmation of Quality and Reliability in Design
Quality Approval (2)
Reliability Test
Process Check same as Quality Approval (1)
Confirmation of Quality and Reliability in Mass Production
Mass Production

Figure 1 Flow Chart of Quality Approval

circumstances and sub-materials. The quality control in the manufacturing process is shown in Fig. 3 corresponding to the manufacturing process.

(1) Quality Control of Semi-final Products and Final Products

Potential failure factors of semiconductor devices should be removed preventively in manufacturing process. To achieve it, check points are set-up in each process, and products which have potential failure factor are not transfer to the next process. Especially, for high reliability semiconductor devices, manufacturing line is rigidly selected, and the quality control in the manufacturing process is tightly executed – rigid check in each process and each lot, 100% inspection in appropriate ways to remove failure factor caused by manufacturing fluctuation, and execution of screening needed, such as high temperature aging and temperature cycling. Contents of inner process quality control are as follows;

- Condition control on individual equipments and workers, and sampling check of semifinal products.
- Proposal and carrying-out improvement of work
- Education of workers
- Maintenance and improvement of yield
- Picking-up of quality problems, and execution of countermeasures
- Transmission of information about quality

(2) Quality Control of Manufacturing Facilities and Measuring Equipment

Equipments for manufacturing semiconductor devices have been developing extraordinarily with necessary high performance devices and improvement of production, and are important factors to determine quality and reliability. In Hitachi, automatization of manufacturing equipments are promoted to improve manufacturing fluctuation, and controls are made to maintain proper operation of high performance equipments and perform the proper function. As for maintenance inspection for quality control, there are daily inspection which is performed daily based on specification related, and periodical inspection which is performed periodically. At the inspection, inspection points listed in the specification are checked one by one not to make any omission. As for adjustment and maintenance of measuring equipments, maintenance number, specification are checked one by one to maintain and improve quality.

(3) Quality Control of Manufacturing Circumstances and Sub-materials

Quality and reliability of semiconductor device is highly

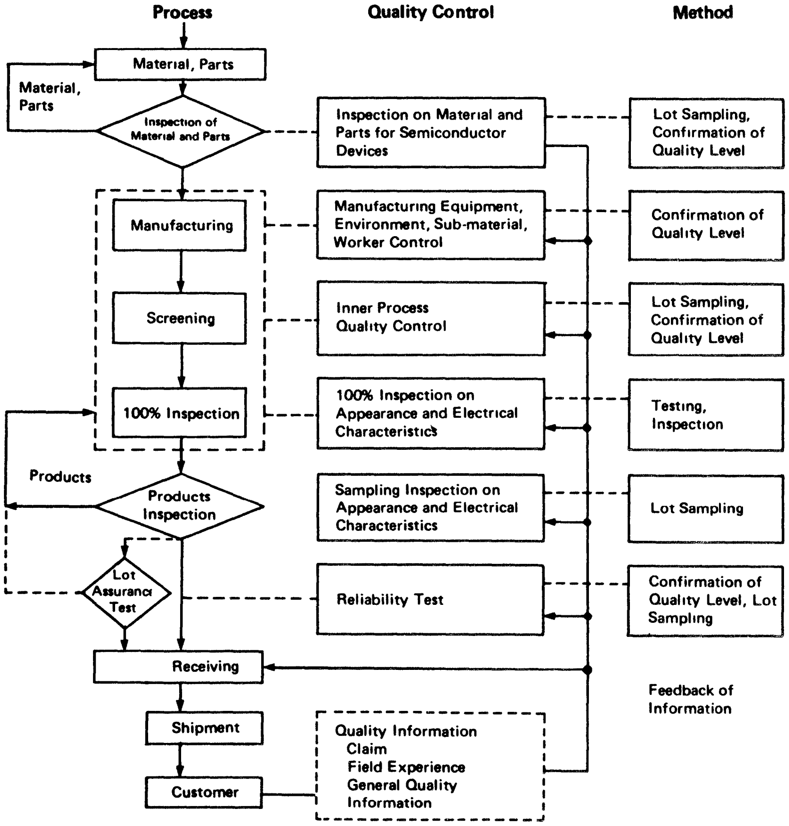

Process
Quality Control
Method
Material, Parts
Material, Parts
Inspection of Material and Parts
Inspection on Material and Parts for Semiconductor Devices
Lot Sampling, Confirmation of Quality Level
Manufacturing
Manufacturing Equipment, Environment, Sub-material, Worker Control
Confirmation of Quality Level
Screening
Inner Process Quality Control
Lot Sampling, Confirmation of Quality Level
100% Inspection
100% Inspection on Appearance and Electrical Characteristics
Testing, Inspection
Products
Products Inspection
Sampling Inspection on Appearance and Electrical Characteristics
Lot Sampling
Lot Assurance Test
Reliability Test
Confirmation of Quality Level, Lot Sampling
Receiving
Feedback of Information
Shipment
Customer
Quality Information
Claim
Field Experience
General Quality
Information

Figure 2 Flow Chart of Quality Control in Manufacturing Process

affected by manufacturing process. Therefore, the controls of manufacturing circumstances – temperature, humidity, dust – and the control of submaterials – gas, pure water – used in manufacturing process are intensively executed. Dust control is described in more detail below.

Dust control is essential to realize higher integration and higher reliability of devices. In Hitachi, maintenance and improvement of cleanness in manufacturing site are executed with paying intensive attention on buildings, facilities, air-conditioning systems, materials delivered-in, clothes, work, etc., and periodical inspection on floating dust in room, falling dusts and dirtiness of floor.

3.3.3 Final Product Inspection and Reliability Assurance

(1) Final Product Inspection

Lot inspection is done by quality assurance department for products which were judged as good products in 100% test, which is final process in manufacturing department. Though 100% of good products is expected, sampling inspection is executed to prevent mixture of failed products by mistake of work, etc. The inspection is executed not only to confirm that the products meet users' requirement, but to consider potential factors. Lot inspection is executed based on MIL-STD-105D.

(2) Reliability Assurance Tests

To assure reliability of semiconductor devices, periodical reliability tests and reliability tests on individual manufacturing lot required by user are performed.

Table 1 Quality Control Check Points of Material and Parts (Example)

Material, Parts	Important Control Items	Point for Check
Wafer	Appearance	Damage and Contamination on Surface
	Dimension	Flatness
	Sheet Resistance	Resistance
	Defect Density	Defect Numbers
	Crystal Axis	
Mask	Appearance	Defect Numbers, Scratch
	Dimension	Dimension Level
	Resistoration	
	Gradation	Uniformity of Gradation
Fine Wire for Wire Bonding	Appearance	Contamination, Scratch, Bend, Twist
	Dimension	
	Purity	Purity Level
	Elongation Ratio	Mechanical Strength
Frame	Appearance	Contamination, Scratch
	Dimension	Dimension Level
	Processing Accuracy	
	Plating	Bondability, Solderability
	Mounting Characteristics	Heat Resistance
Ceramic Package	Appearance	Contamination, Scratch
	Dimension	Dimension Level
	Leak Resistance	Airtightness
	Plating	Bondability, Solderability
	Mounting Characteristics	Heat Resistance
	Electrical Characteristics	
	Mechanical Strength	Mechanical Strength
Plastic	Composition	Characteristics of Plastic Material
	Electrical Characteristics	
	Thermal Characteristics	
	Molding Performance	Molding Performance
	Mounting Characteristics	Mounting Characteristics

Process	Control Point		Purpose of Control
▽ Purchase of Material			
—Wafer →	Wafer	Characteristics, Appearance	Scratch, Removal of Crystal Defect Wafer
○ Surface Oxidation	Oxidation		Assurance of Resistance
□ Inspection on Surface Oxidation		Appearance, Thickness of Oxide Film	Pinhole, Scratch
○ Photo Resist	Photo Resist		
□ Inspection on Photo Resist		Dimension, Appearance	Dimension Level
◇ PQC Level Check			Check of Photo Resist
○ Diffusion	Diffusion	Diffusion Depth, Sheet Resistance	Diffusion Status
□ Inspection on Diffusion		Gate Width	Control of Basic Parameters
◇ PQC Level Check		Characteristics of Oxide Film Breakdown Voltage	(V_{TH}, etc.) Cleanness of surface, Prior Check of V_{IH} Breakdown Voltage Check
○ Evaporation	Evaporation	Thickness of Vapor Film, Scratch, Contamination	Assurance of Standard Thickness
□ Inspection on Evaporation			
◇ PQC Level Check			
□ Wafer Inspection	Wafer	Thickness, V_{TH} Characteristics	Prevention of Crack, Quality Assurance of Scribe
□ Inspection on Chip Electrical Characteristics	Chip	Electrical Characteristics	
○ Chip Scribe		Appearance of Chip	
□ Inspection on Chip Appearance			
◇ PQC Lot Judgement			
—Frame → ○ Assembling	Assembling	Appearance after Chip Bonding Appearance after Wire Bonding	Quality Check of Chip Bonding Quality Check of Wire Bonding
◇ PQC Level Check		Pull Strength, Compression Width, Shear Strength	Prevention of Open and Short
□ Inspection after Assembling		Appearance after Assembling	
◇ PQC Lot Judgement			
—Package → ○ Sealing	Sealing	Appearance after Sealing Outline, Dimension	Guarantee of Appearance and Dimension
◇ PQC Level Check	Marking	Marking Strength	
○ Final Electrical Inspection			
◇ Failure Analysis		Analysis of Failures, Failure Mode, Mechanism	Feedback of Analysis Information
□ Appearance Inspection			
△ Sampling Inspection on Products			
○ Receiving			
Shipment			

Figure 3 Example of Inner Process Quality Control

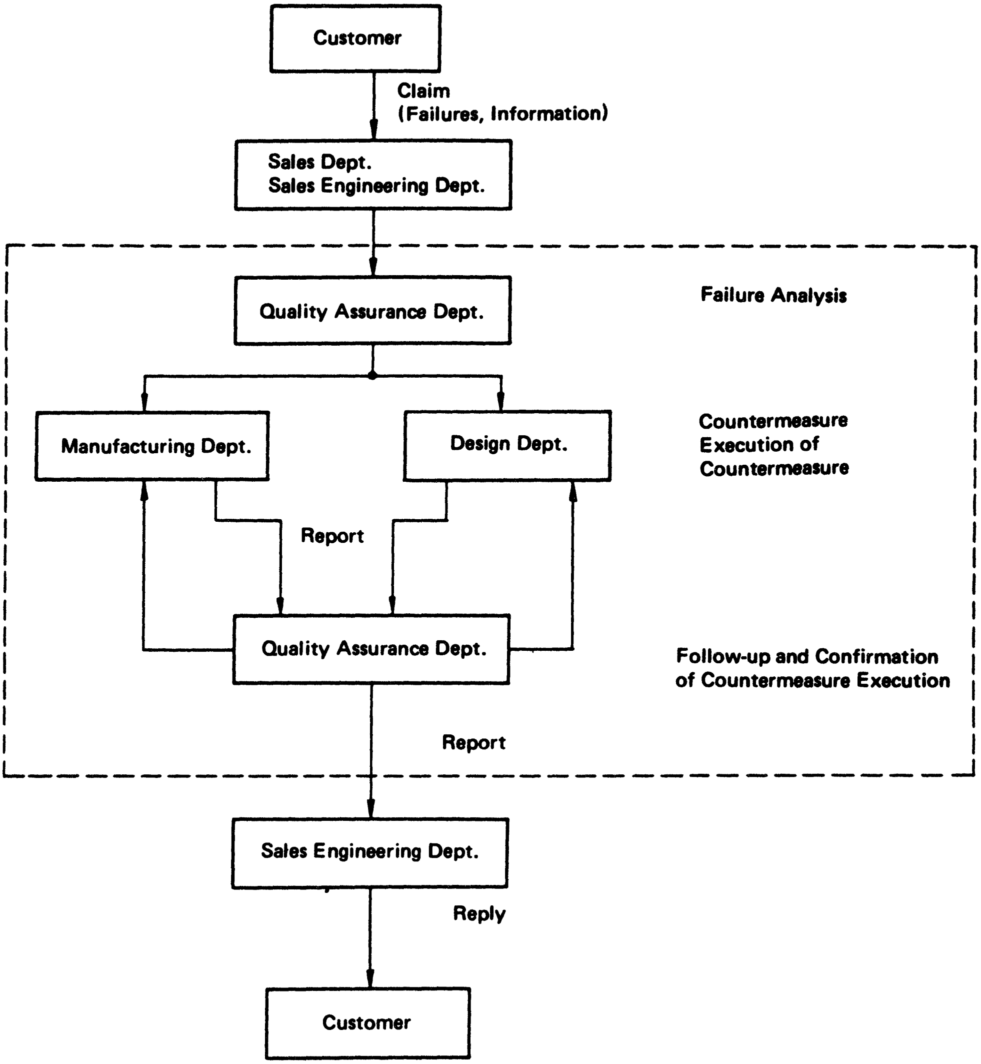


Figure 4 Process Flow Chart of Field Failure

RELIABILITY TEST DATA OF MICROCOMPUTER

1. INTRODUCTION

Microcomputer is required to provide higher reliability and quality with increasing function, enlarging scale and widening application. To meet this demand, Hitachi is improving the quality by evaluating reliability, building up quality in process, strengthening inspection and analyzing field data etc..

This chapter describes reliability and quality assurance data for Hitachi 8-bit and 16-bit multi-chip microcomputer based on test and failure analysis results. More detail data and new information will be reported in another reliability data sheet.

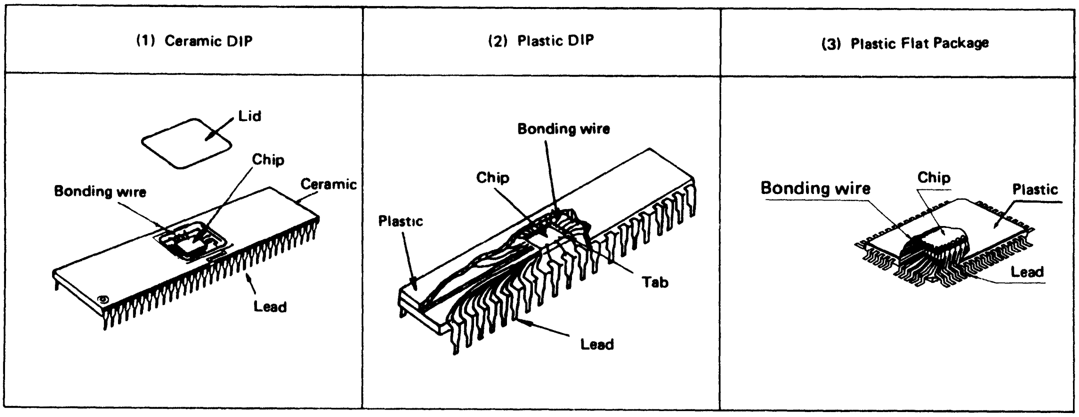
2. PACKAGE AND CHIP STRUCTURE

2.1 Package

The reliability of plastic molded type has been greatly improved, recently their applications have been expanded to automobiles measuring and control systems, and computer terminal equipment operated under relatively severe conditions and production output and application of plastic molded type will continue to increase.

To meet such requirements, Hitachi has considerably improved moisture resistance, operation stability, and chip and plastic manufacturing process.

Plastic and ceramic package type structure are shown in Figure 1 and Table 1.



(1) Ceramic DIP

(2) Plastic DIP

(3) Plastic Flat Package

Figure 1 Package Structure

Table 1 Package Material and Properties

Item	Ceramic DIP	Plastic DIP	Plastic Flat Package
Package	Alumina	Epoxy	Epoxy
Lead	Tin plating Brazed Alloy 42	Solder dipping Alloy 42 or Cu	Solder plating Alloy 42
Seal	Au-Sn Alloy	N.A	N.A
Die bond	Au-Si	Au-Si or Ag paste	Au-Si or Ag paste
Wire bond	Ultrasonic	Thermo compression	Thermo compression
Wire	Al	Au	Au

2.2 Chip Structure

Hitachi microcomputers are produced in NMOS E/D technology or low power CMOS technology. Si-gate process is used in both types because of high reliability and high density.

Chip structure and basic circuit are shown in Figure 2.

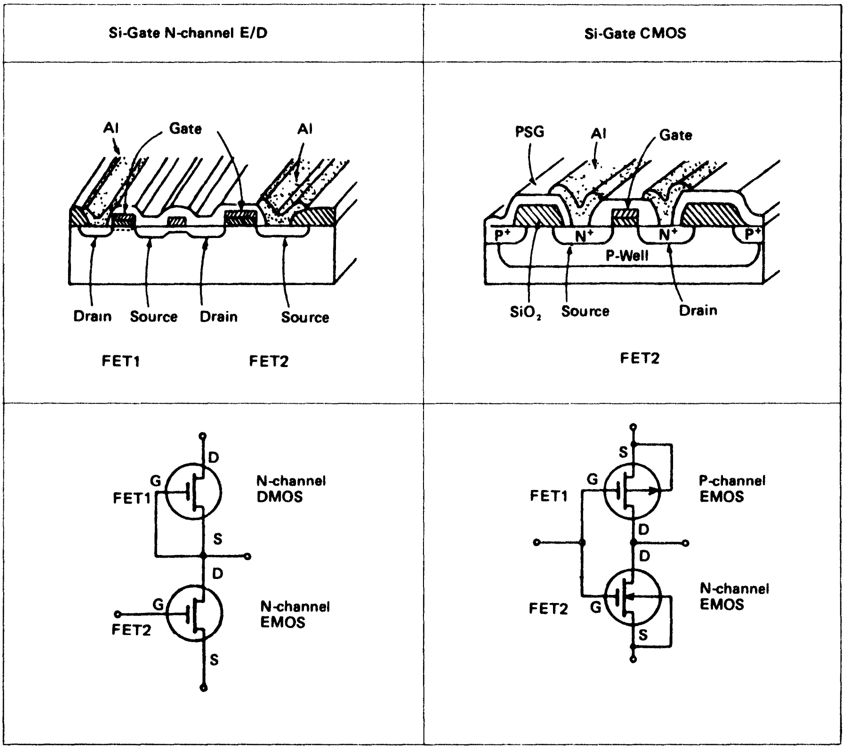


Figure 2 Chip Structure and Basic Circuit

3. QUALITY QUALIFICATION AND EVALUATION

3.1 Reliability Test Methods

Reliability test methods shown in Table 2 are used to qualify and evaluate the new products and new process.

Table 2 Reliability Test Methods

Test Items	Test Condition	MIL-STD-883B Method No.
Operating Life Test	125°C, 1000hr	1005,2
High Temp, Storage	Tstg max, 1000hr	1008,1
Low Temp, Storage	Tstg min, 1000hr	
Steady State Humidity	65°C 95%RH, 1000hr	
Steady State Humidity Biased	85°C 85%RH, 1000hr	
Temperature Cycling	-55°C ~ 150°C, 10 cycles	1010,4
Temperature Cycling	-20°C ~ 125°C, 200 cycles	
Thermal Shock	0°C ~ 100°C, 100 cycles	1011,3
Soldering Heat	260°C, 10 sec	
Mechanical Shock	1500G 0.5 msec, 3 times/X, Y, Z	2002,2
Vibration Fatigue	60Hz 20G, 32hrs/X, Y, Z	2005,1
Variable Frequency	20~2000Hz 20G, 4 min/X, Y, Z	2007,1
Constant Acceleration	20000G, 1 min/X, Y, Z	2001,2
Lead Integrity	225gr, 90° 3 times	2004,3

3.2 Reliability Test Result

Reliability test result of 8-bit microprocessors is shown in Table 3 to Table 7, that of 16-bit microprocessors in Table 8, Table 9. There is little difference according to device series, as the design and production process, etc. are standardized.

Table 3 Dynamic Life Test (8-bit microprocessor)

Device Type	Sample Size	Component Hours	Failures
HD6800	248 pcs	248000	0
HD6802	452	153712	1*
HD6809	85	85000	0
Total	785	486712	1

*leakage current

Estimated Field Failure Rate
= 0.01% / 1000 hrs at Ta = 75°C
(Activation Energy = 0.7eV, Confidence Level 60%)

Table 4 High Temperature, High Humidity Test (8-bit microprocessor) (Moisture Resistance Test)

(1) 85°C 85%RH Bias Test

Device Type	Vcc Bias	168 hrs	500 hrs	1000 hrs
HD6800P	5.5V	0/45	0/45	0/45
HD6802P	5.5V	0/38	0/38	0/38
HD6809P	5.5V	0/22	0/22	0/22
Total		0/105	0/105	0/105

(2) High Temperature-High Humidity Storage Life Test

Device Type	Condition	168 hrs	500 hrs	1000 hrs
HD6800P	65°C 95%RH	0/22	0/22	0/22
HD6802P	80°C 90%RH	0/22	0/22	0/22
HD6802P	65°C 95%RH	0/38	0/38	0/38
HD6809P	65°C 95%RH	0/45	0/45	0/45

(3) Pressure Cooker Test
(Condition ; 2atm 121°C)

Device Type	40 hrs	60 hrs	100 hrs
HD6800P	0/42	0/42	0/42
HD6802P	0/22	0/22	0/22

(4) MIL-STD-883B Moisture Resistance Test
(Condition; 65°C ~ -10°C, over 90%RH, Vcc = 5.5V)

Device Type	10 cycles	20 cycles	40 cycles
HD6800P	0/25	0/25	0/25
HD6802P	0/25	0/25	0/25

Table 5 Temperature Cycling Test (8-bit microprocessor) (−55°C ~ 25°C ~ 150°C)

Device Type	10 cycles	100 cycles	200 cycles
HD6800P	0/453	0/44	0/44
HD6802P	0/502	0/77	0/77
HD6809P	0/202	0/45	0/45

Table 6 High Temperature, Low Temperature Storage Life Test (8-bit microprocessor)

Device	Temperature	168 hrs	500 hrs	1000 hrs
MPU total	150°C	0/88	0/88	0/88
	-55°C	0/76	0/76	0/76

Table 7 Mechanical and Environmental Test (8-bit microprocessor)

Test Item	Condition	Plastic DIP		Flat Plastic Package	
		Sample Size	Failure	Sample Size	Failure
Thermal Shock	0°C ~ 100°C 10 cycles	110	0	100	0
Soldering Heat	260°C, 10 sec.	180	0	20	0
Salt Water Spray	35°C, NaCl 5% 24 hrs	110	0	20	0
Solderability	230°C, 5 sec. Rosin flux	159	0	34	0
Drop Test	75cm, maple board 3 times	110	0	20	0
Mechanical Shock	1500G, 0.5 ms 3 times/X, Y, Z	110	0	20	0
Vibration Fatigue	60 Hz, 20G 32 hrs/X, Y, Z	110	0	20	0
Vibration Variable Freq.	100 ~ 2000 Hz 20G, 4 times/X, Y, Z	110	0	20	0
Lead Integrity	225 g, 90° Bonding 3 times	110	0	20	0

Table 8 Dynamic Life Test (16-bit microprocessor)

Device Type	Condition		168 hrs	500 hrs	1000 hrs
	Ta	Vcc			
HD68000	125°C	5.5V	0/62	0/62	0/62
	150°C	5.5V	0/52	0/52	0/52

Estimated Field Failure Rate
= 0.013%/1000 hrs at Ta = 75°C
(Activation Energy 0.7eV, Confidence Level 60%)

Table 9 Mechanical and Environmental Test (16-bit microprocessor)

Test Item	Condition	Device Type	
		Sample Size	Failure
High Temperature Storage	Ta = 295°C, 1000 hrs	42	0
Low Temperature Storage	Ta = -55°C, 1000 hrs	42	0
Temperature Cycling (1)	-55°C ~ 25°C ~ 150°C 10 cycles	189	0
Temperature Cycling (2)	-20°C ~ 25°C ~ 125°C 500 cycles	44	0
Thermal Shock	-55°C ~ 125°C 15 cycles	44	0
Soldering heat	260°C, 10 sec	44	0
Solderability	230°C, 5 sec	44	0
Mechanical Shock	1500G, 0.5 msec 3 times/X, Y, Z	44	0
Vibration Variable Freq.	20 ~ 2000 Hz, 20G 3 times/X, Y, Z	44	0
Constant Acceleration	20000G 1 min/X, Y, Z	44	0

4. PRECAUTION

4.1 Storage

It is preferable to store semiconductor devices in the following ways to prevent detrioration in their electrical characteristics, solderability, and appearance, or breakage.

(1) Store in an ambient temperature of 5 to 30°C, and in a relative humidity of 40 to 60%.
(2) Store in a clean air environment, free from dust and active gas.
(3) Store in a container which does not induce static electricity.
(4) Store without any physical load.
(5) If semiconductor devices are stored for a long time, store them in the unfabricated form. If their lead wires are formed beforehand, bent parts may corrode during storage.
(6) If the chips are unsealed, store them in a cool, dry, dark, and dustless place. Assemble them within 5 days after unpacking. Storage in nitrogen gas is desirable. They can be stored for 20 days or less in dry nitrogen gas with a dew point at -30°C or lower. Unpacked devices must not be stored for over 3 months.
(7) Take care not to allow condensation during storage due to rapid temperature changes.

4.2 Transportation

As with storage methods, general precautions for other electronic component parts are applicable to the transportation of semiconductors, semiconductor-incorporating units and other similar systems. In addition, the following considerations must be given, too:

(1) Use containers or jigs which will not induce static electricity as the result of vibration during transportation. It is desirable to use an electrically conductive container or aluminium foil.
(2) In order to prevent device breakage from clothes-induced static electricity, workers should be properly grounded with a resistor while handling devices. The resistor of about 1 M ohm must be provided near the worker to protect from electric shock.
(3) When transporting the printed circuit boards on which semiconductor devices are mounted, suitable preventive measures against static electricity induction must be taken; for example, voltage built-up is prevented by shorting terminal circuit. When a belt conveyor is used, prevent the conveyor belt from being electrically charged by applying some surface treatment.
(4) When transporting semiconductor devices or printed circuit boards, minimize mechanical vibration and shock.

4.3 Handling for Measurement

Avoid static electricity, noise and surge-voltage when semiconductor devices are measured. It is possible to prevent breakage by shorting their terminal circuits to equalize electrical potential during transportation. However, when the devices are to be measured or mounted, their terminals are left open to provide the possibility that they may be accidentally touched by a worker, measuring instrument, work bench, soldering iron, belt conveyor, etc. The device will fail if it touches something which leaks current or has a static charge. Take care not to allow curve tracers, synchroscopes, pulse generators, D.C. stabilizing power supply units etc. to leak current through their terminals or housings.

Especially, while the devices are being tested, take care not to apply surge voltage from the tester, to attach a clamping circuit to the tester, or not to apply any abnormal voltage through a bad contact from a current source.

During measurement, avoid miswiring and short-circuiting. When inspecting a printed circuit board, make sure that no soldering bridge or foreign matter exists before turning on the power switch.

Since these precautions depend upon the types of semiconductor devices, contact Hitachi for further details.

4.4 Soldering

Semiconductor devices should not be left at high temperatures for a long time. Regardless of the soldering method, soldering must be done in a short time and at the lowest possible temperature. Soldering work must meet soldering heat test conditions, namely, 260°C for 10 seconds and 350°C for 3 seconds at a point 1 to 1.5 mm away from the end of the device package.

Use of a strong alkali or acid flux may corrode the leads, deteriorating device characteristics. The recommended soldering iron is the type that is operated with a secondary voltage supplied by a transformer and grounded to protect from lead current. Solder the leads at the farthest point from the device package.

4.5 Removing Residual Flux

To ensure the reliability of electronic systems, residual flux must be removed from circuit boards. Detergent or ultrasonic cleaning is usually applied. If chloric detergent is used for the plastic molded devices, package corrosion may occur. Since cleaning over extended periods or at high temperatures will cause swollen chip coating due to solvent permeation, select the type of detergent and cleaning condition carefully. Lotus Solvent and Dyfron Solvent are recommended as a detergent. Do not use any trichloroethylene solvent. For ultrasonic cleaning, the following conditions are advisable:

- Frequency: 28 to 29 kHz (to avoid device resonation)
- Ultrasonic output: 15W/ℓ
- Keep the devices out of direct contact with the power generator.
- Cleaning time: Less than 30 seconds

■ PROGRAM DEVELOPMENT AND SUPPORT SYSTEM OF 8-BIT/16-BIT MICROPROCESSOR

H680SD200 is prepared as system development device to develop software and hardware of various types of microcomputer system.

Fig. 1 shows the program development procedure using this system development device.

H680SD200 loads a universal OS, CP/M-68K® developed jointly with Digital Research Inc. and operates with the existing CP/M®.

*CP/M® and CP/M-68K® are registered trademarks of Digital Research Inc.

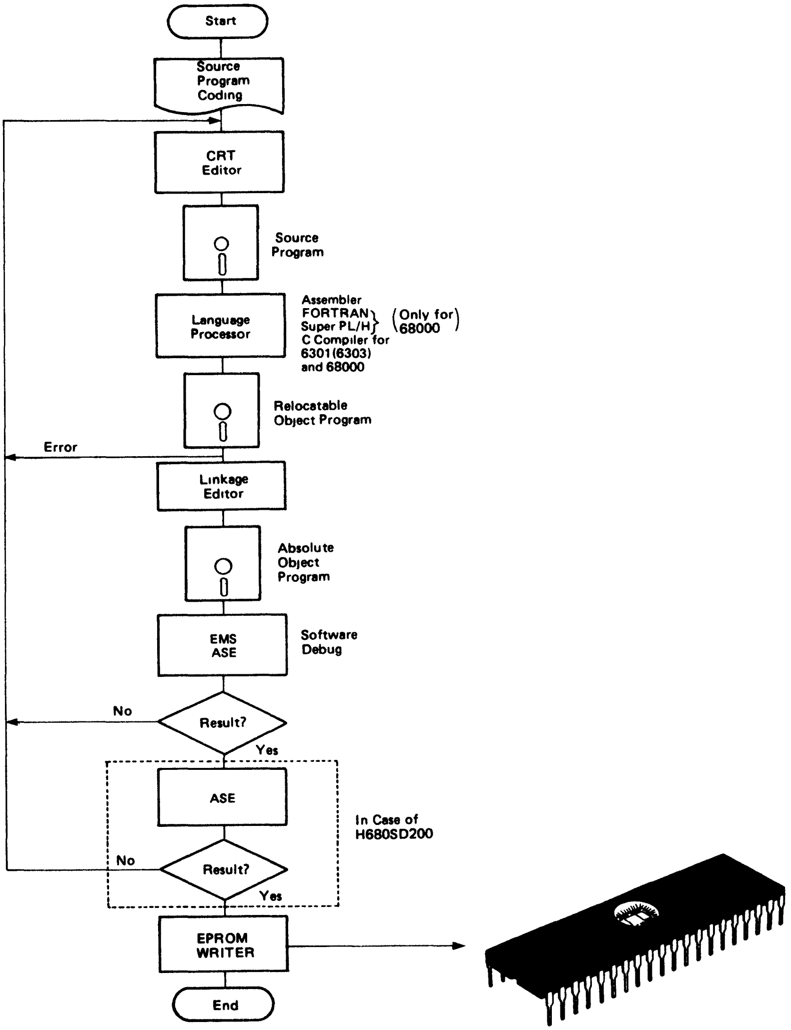


Fig. 1 Program Development Procedure

Table 1 System Development Equipment SD200

MPU	Product Name	Product Code	Function	Note
—	CP/M-68K	S680CPM3F	• Single user Operating System • 68000 Assembler, C compiler, Screen editor and Linker are included	
—	VAX—11 Interface Program	S680CLC3F	• Interface Program between the SD200 and theVAX-11. • File transfer function. • VT52 Terminal Emulation.	Option
	DATA I/O EPROM Programmer Interface Program	S680CDI1F	• Interface Program between the SD200 and the DATA I/O EPROM Programmer model 22/29.	
	PKW-1000/7000 EPROM Programmer Interface Program	S680CPK2F	• Interface Program between the SD200 and the PKW-1000/7000 EPROM Programmer (Aval Corp. Japan).,	
16-bit MPU HD68000	FORTRAN	S680CFR1F	• FORTRAN Compiler. (Subset of FORTRAN77)	Option
	Super PL/H	S680CPL1F	• Super PL/H Compiler.	Option
	Symbolic Debugger	S680CSD2F	• Symbolic Debugger for programs written in 68000 Assembler or Super PL/H.	Option
8 bit MPU/MCU	64180ASE	S180CAS1F	• Realtime In-circuit Emulator for 64180.	Supplied with H180AS01E
	6305/63L05/6805 Macro Assembler	S35XAS6-F	• 6305Z/63L05/6805 Macro Assembler. • Linkage editor is included.	Option
	6301/6801/6800 Macro Assembler	S31XAS6-F	• 6301/6801/6800 Macro Assembler. • Linkage editor is included.	Option
	6301 C Compiler	S31CCLN-F	• C Compiler for 6301(6303).	Option

Table 2 Cross System

MPU	Machine	OS	Product Name	Product Code	Function
8-bit MCU	Intel MDS	ISIS-II	6305/63L05/6805 Assembler	S35MDS1-F	· 6305/63L05/6805 Assembler. · Object code is absolute address format. · Conditional assemble function.
		CP/M	6305/63L05/6805 Assembler	S35MDS2-F	· 6305/63L05/6805 Assembler. · Object code is absolute address format. · Conditional assemble function.
		ISIS-II	6301 Assembler	S31MDS1-F	· 6301/6801 Assembler. · Object code is absolute address format. · Conditional Assemble function.
		CP/M	6301 Assembler	S31MDS2-F	· 6301/6801 Assembler. · Object code is absolute address format. · Conditional Assemble function.
	IBM-PC	PC-DOS	6301 Macro Assembler	S31IAS1-F*	· 6301 Macro Assembler. · Linkage Editor is included.
			6305 Macro Assembler	S35IAS1-F*	· 6305 Macro Assembler. · Linkage Editor is included.

Table 3 Third Parties' Products

Assemblers for HITACHI's microcomputers are provided by the other companies. Hitachi introduce some venders and their products listed below. Please contact those venders directly if you have questions or requests to purchase these products.

Vender Name	Product Name	OS/System	Product Code
MICROTEC 505W Olive, Suite 325 Sunnyvale, CA94086 (408)733-2919 U.S.A.	6301 Assembler	VAX11	ASM68
	6305 Assembler		ASM05
	6809 Assembler		ASM69
	68000 Assembler		ASM68K
	64180 Assembler		ASM180
	64180 Simulator		INT180
	64180 C		MCC180
	64180 Pascal		PAS180
	6301 Assembler	IBM-PC	ASM68
	6305 Assembler		ASM05
	64180 Assembler		ASM180
	64180 C		MCC180
	64180 Pascal		PAS180
CAMELOT 79 London Road Knebworth Herts, SG3 6HG, England Stevenage (0438) 812215	6301 Assembler	IBM-PC	
	6305 Assembler		*
AVOCET SYSTEMS, INC. 804 South State St. Dover, DE19901 (302) 734-0151 U.S.A.	6800/6801/6301 Assembler	MS-DOS, CP/M CP/M-86	XASM-68
	6805 Assembler	MS-DOS, CP/M CP/M-86	XASM-05
	6309/6809 Assembler	MS-DOS, CP/M CP/M-86	XASM-09
	64180 Assembler	MS-DOS, CP/M CP/M-86	XASM-180
MICROWARE SYSTEMS CORPORATION 5835 Grand Avenue Dos Moines, IA50312 (512) 279-8844 U.S.A.	6309/6809 Assembler	OS-9	–
	68000/68HC000 Assembler	OS-9	KCRS

*Under development

- **Development System for 4-Bit, 8-Bit, and 16-Bit Microcomputers <H680SD200>**

The H680SD200 is a development system for Hitachi 4-bit, 8-bit and 16-bit microcomputers. It is a desktop system in which a 16-bit microprocessor HD68000 is loaded as the CPU. Its standard system configuration includes a CRT, a keyboard, and two floppy disk drives. An assembler, compiler, and in-circuit emulator (ASE) associated with the user's MCU are available as options.

APPLICABLE DEVICES

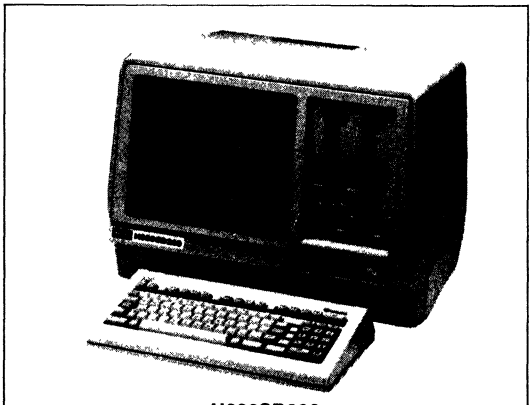
- HMCS400 series
- HD6305U, HD6305V
- HD6301V, HD6301X, HD6301Y
- HD64180
- HD68000, HD68HC000

(Other 4-bit and 8-bit microcomputers will be supported in the future)

- **FEATURES**

- Adopts general CP/M-68K® operating system
- Two internal 8 inch floppy disk drives (double-sided, double-density) and a 40M byte hard disk (available as an option) make it possible to provide substantial external memory.
- Since CRT editor (screen editor) is included in the standard system, efficient programming, editing, and debugging of source programs are possible.
- C compiler for HD68000 is included. FORTRAN and Super PL/H for HD68000 and C Compiler for HD6301 (HD6303) are available as options.
- User prototype system can easily be debugged using incircuit emulator (ASE) associated with the user's MCU.
- With connection of VAX-11® to RS-232C interface, H680SD200 operates as VAX-11® (OS, VMS) work station.
- When 2M byte memory board is connected, high-speed operation can be realized.
- Following interface are included
  (1) EPROM programmer
  (2) Printer (Centronics specification)
  (3) Serial interface emulator for 4-bit and 8-bit single chip microcomputers

*CP/M® is a registered trade mark of Digital Research Inc.
**VAX-11® is a registered trade mark of Digital Equipment Corp.



H680SD200

HD64180 THIRD-PARTY DEVELOPMENT TOOLS

Product: Cross-Assemblers and Cross-Compilers

Company	ASM	S/W SIM	C COMP	PASCAL	BASIC
American Automation (714-731-1661)	V/I	V/I	V/I		
Microtec Research (408-733-2919)	V/I	V/I	V/I	V/I	
Avocet Systems (800-448-8500)	C/I	C/I			
2500AD Software (303-369-5001)	C/I				
BSO (617-894-7800)	V	V	V	V	
Sumitronics (408-737-7683)	V				
SLR Systems (800-833-3061)	C				
Uniware/SDS (312-971-8170)	V				
Softaid (800-433-8812)					C
Allen Ashley (818-793-5748)	I				

[I = IBM-PC, V = VAX, C = CP/M]

Product: Support Tools

Company	Product Description
Electronic Molding (401-769-3800)	Shrink-DIP Adapter for Breadboarding P/N 28764-72-341
Robinson Nugent (812-945-0211)	Shrink-DIP Socket P/N TSS-6475-TNG
Yamaiche/Nepenthe (415-856-9332)	Shrink-DIP Socket P/N IC 38-64075-G4 S-D Test Socket P/N IC 76-64075-G4
Methode Electronics (312-392-3500)	PLCC Adapter for Hitachi's ASE
TSI, Inc (800-874-2288)	64180 IBM-PC Card with DSD80 Remote Software Debugger
Micromint (800-635-3355)	64180 Evaluation Board P/N SB180

Product: Operating Systems

Company	Type of Operating System
Echelon (415-948-3820)	ZCPR3 (CP/M)
JMI Software (215-628-0840)	C Executive/80 Multi-Tasking Kernel
Decmation (408-980-1678)	Quick-Task Realtime Executive
Hunter & Ready (415-326-2950)	VRTX/80 Multi-Tasking Kernel (Z80)
IPI (516-938-6600)	MTOS/80 Multi-Tasking Kernel (Z80)

HITACHI ORDERING INFORMATION

Part Number	Description
H180ASE02	Adaptive System Evaluator, ASE-II
H680SM01S	256K Byte Emulation Memory Board (Option)
H180ABX	8 MHz Buffer Box for ASE-I User, Includes V2.0 System Software
H180CP01	PLCC-68 MPU Adapter for 1 Mbyte Addressing (Option)

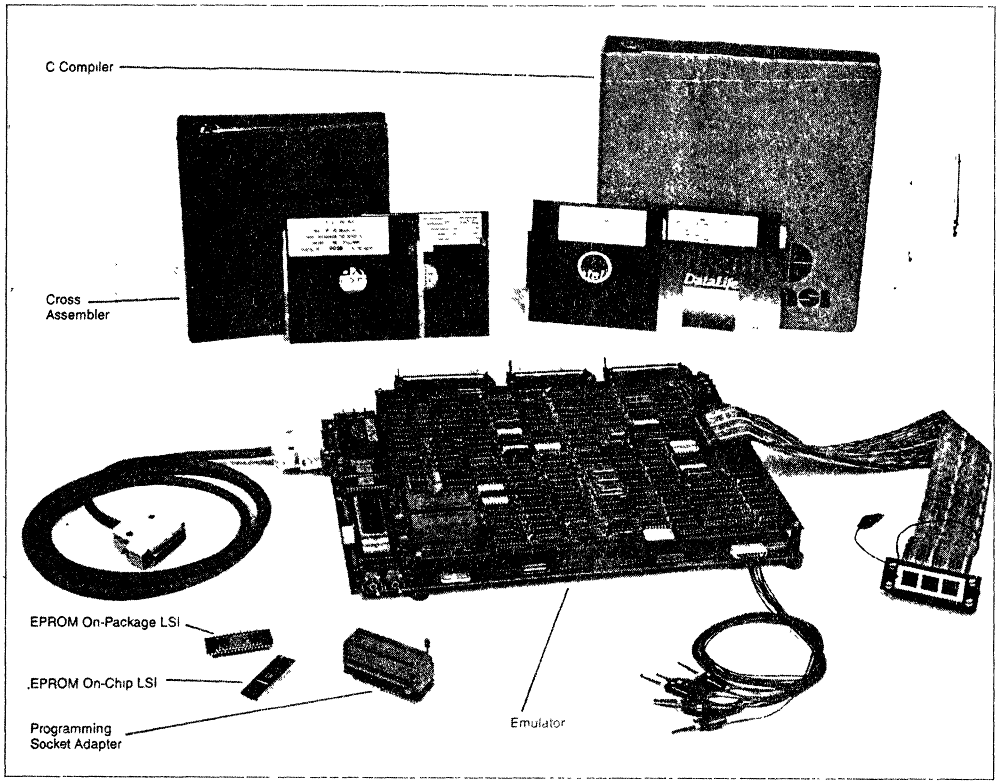
Device Availability

	Description	HD6303R	HD6303X	HD6303Y	HD6305X2 HD6305Y2	HD64180R	HD64180S
HARDWARE	Emulator	HS31VEML04H(1) (H31MIX4)(2)	HS31XEML02H(1) (H31MIX2)(2)	HS31YEML03H(1) (H31MIX3)(2)	HD35YEML05H(1) (H35MIX5)(2)	HD180ABX02H(4)	HS180ABX05H(4)
	ASE (Adaptive System Emulator)					HS180AST01H	HS180AST01H
	User Cable					HS180ACUC1H	
	Emulation Memory Board up to 64K Byte		H64EMB02	H64EMB02			
	Emulation Memory Board					H680SM01S(6)	H680SM01S(6)
	Evaluation Board						US180EVB01H
	Programming Socket Adapter			HS31YESS11H			
	Programming Socket Adapter						
	Programming Socket Adapter						
SOFTWARE	Cross Assembler (IBM-PC)	S31IBMPC(3)	S31IBMPC(3)	S31IBMPC(3)	S35IBMPC(3)		
	C Compiler (IBM-PC)	US31PCLI1SF	US31PCLI1SF	US31PCLI1SF			
LITERATURE	Data Sheet	M21T006 M21T132	M21T006 M21T132	M21T006 M21T132	M21T006 M21T132	M21T132	M21T132
	Hand Book	M21T019(5)	M21T019(5)	M21T019(5)	M21T020(5)		
	Specification Sheet					M21T011	M21T013
	Hardware Manual					M21T053	M21T053
	Product Brief					M21T025	

HD641180X HD643180X HD647180X	HD64180Z	HD648180W*	HD6309**	HD68HC000**	HD68000**	HD6802**	HD6803	HD6809**
HS180ABX04H[4]	HS180ABX03H[4]						HG1EVT2[1]	
HS180AST01H	HS180AST01H							
	HS180ACUC1M							
							H64EMB01	
H680SM01S[6]	H680SM01S[7]							
HS18XESF01H HS18XESC01H								
HS18XESS01H								
HS18XESG01H								
M21T132	M21T132	M21T132	M21T132	M21T132 M22T003	M21T132 M22T003	M21T132	M21T006 M21T132	M21T132
M21T012	M21T011							
M21T113	M21T053							
M21T026	M21T025							

(Note) 1. Emulator includes an RS-232 port for connection to IBM PC or PC compatible machines. Software not included.
2. Same as footnote (1) above, but shipped with cross assembler (8″ floppy disk) that operates with Hitachi's H68SD5 development system.
3. Developed by one of Hitachi's engineering subsidiaries Cross assembler include software utility to download/upload code between PC host and emulator.
4. Must be used with ASE station (P/N: HS180AST01H)
5. Includes user's manual, hardware and software application notes, and other relevant information
6. 64K bytes user memory is provided in standard configuration. Can be expanded up to 512K bytes. Maximum of two 256K byte memory boards can be installed (optional).

*Contact Marketing.
**Refers to third-party support tools.



Typical Support Tools for HD6303/05 Series

■ **SINGLE-CHIP MICROCOMPUTER SUPPORT SYSTEMS**

Hitachi and its engineering subsidiaries make hardware and software support tools to operate with many popular host computers and expedite the development of the microcomputer-based target system. The support system includes in-circuit emulators, cross assemblers, passive socket adapters for easily programming EPROM on-chip devices, and documentation.

In addition to hardware and software support, Hitachi has Field Application Engineers (FAE) to help identify the most cost-effective IC(s) for your application and answer your technical questions

■ **IN-CIRCUIT EMULATOR FUNCTIONS FOR HD6303/05 Series***

- Serial interface connection to many host computers via RS-232C port.
- Executes user's program in real-time on some emulators, or when loaded in emulator's memory starting from a selected address. Execution is interrupted when breakpoints are detected, or when RESET or ABORT is switched
- Single step tracing of user's program is possible. Data in registers and data in memory are displayed after every execution
- Breakpoints can be set in user's program by using the program counter address, data bus, or external signal probes Breakpoints can be displayed and changed.
- Data in internal registers of the subject microcomputer can be displayed or changed
- Real-time tracing is possible on most emulators, the emulator stores and displays bus data and external signals for up to 1011 machine cycles on some emulators, or 2035 machine cycles on other emulators before and after the address where a breakpoint is set
- Line assembler and disassembler on some emulators

*Functions listed in the overview may not exactly apply to all emulators Refer to the applicable emulator user's manual for further information

■ **CROSS ASSEMBLER FUNCTIONS (PC-DOS)**

The software is divided into six main parts.

- **Structured Relocatable Cross Macro Assembler**

  The cross assembler is designed to meet the specification outlined in Hitachi's HD6303 and HD6305 assembler user's manual, which means that mnemonic, macro and directive compatibility is maintained

  The assembler also offers a structured code facility, similar to that found in some high level languages The main structured features are listed below

  IF . . THEN . ELSE . . DO . WHILE . . . REPEAT . . UNTIL . . FOR . . TO

  CALL (with parameters passed on the stack)
- **Linker**

  The linker concatenates and locates all relocatable modules into an executable object file (Motorola S-type format). Start addresses of relocatable program and data sections can be entered at linkage time

- **Macro Librarian**
  Named libraries of useful macros can be built by the user, saving time during generation of source code. The macro librarian is searched during assembly time for the appropriate macro definitions that do not appear in the source file
- **Object Module Librarian**
  Named libraries of useful object modules can be built by the user. The libraries called up at linkage time are searched by the linker to see if unsatisfied external references can be resolved. Object modules which satisfy the unresolved references are automatically included in the executable object file (S-record format).
- **Emulator Interface Software**
  The interface software allows connection between Hitachi's serially linked emulators and the IBM PC using an RS232 asynchronous interface.

  Commands from the PC keyboard are directed to the emulator and responses are displayed on the screen. File upload and download in Motorola S-type format enables assembled and linked programs to be run on the emulator. Real time trace facilities are available on all serial linked emulators.
- **EPROM Programmer Interface Software**
  The interface software allows connection to most proprietary EPROM Programmers for downloading (or uploading) executable object modules in Motorola S-type* data format. The programmers can be run either in REMOTE CONTROL or LOCAL mode.

  In local mode, programmer commands can be entered on the programmer keyboard and upload/download of object modules can be activated using the IBM PC keyboard.

  In remote control mode, all programmer commands are entered via the IBM PC keyboard

  All programmer commands will be specific to the particular programmer used.

■ C COMPILER FUNCTIONS (PC-DOS)

The HD6303 and HD6305** compiler comprises three programs, a pre-processor, the main compiler and an optimiser. The system also provides standard library files (which facilitates I/O and floating point operations), the standard "include" files which contain the necessary declarations for the usage of library function. Runtime object files for integer and floating point arithmetic are included. Compatible with Hitachi and Microtec Research*** assemblers.

- **Compiler Options**
  The following tables indicate the options available during pre-processing and compiling.

Table 1. Pre-processor Options

No.	Option	Description
1	A	Issues error messages to the pre-processor source program file
2	D	Defines a macro name
3	L	Inserts the original source program lines into the pre-processed source program as comments

Table 2. Compiler Options

No.	Option	Description
1	P	Generates object code which calls a profiler routine (a routine which profiles the history of the program execution) everytime a function is called (see Note 1).
3	L	Generates object code which calls a stack check routine everytime a function is called (see Note 1)

Note 1: The profiler routine and the stack check routine should be prepared in a separate module for your own target system.

- **Limits in Compilation**

  (1) Length of an input line 512 characters
  (2) Length of a character string. 510 characters
  (3) Number of external names 156
  (4) Effective length of identifiers. 8 characters
  (5) Effective length of external identifiers 6
  (6) Nest of conditional complication 32 level
  (7) Nesting of file inclusion. 14 level
  (8) Number of macro parameters. 32
  (9) Length of a macro definition. 512 characters
  (10) Recursive expansion of a macro name: 32 times

- **Data Size**

  (1) Char type: 8 bit
  (2) Short type, int type. 16 bit
  (3) Long type: 32 bit
  (4) Float type: 32 bit
  (5) Double type: 64 bit
  (6) Pointer type: 16 bit

  No data alignment is done in allocation of structured data

*Motorola S-type is a trademark of Motorola, Inc.

**Conforms to Kerninghan and Ritchie C programming language standard rather than ANSI C programming language standard.

***Microtec Research is a trademark of Microtec Research, Inc.

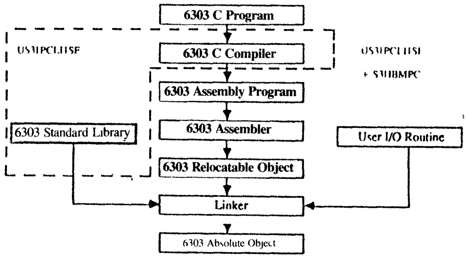
(3) *Character Handling Library Functions*
isalnum, isalpha, issascii, iscntl, isdigit, islower, isprint, ispunct, isspace, isupper, tolower, toupper.

(4) *Character String Handling Library Functions*
index, rindex, strcat, strcmp, strcpy, strlen, strncat, strncopy

(5) *Data Conversion Library Function*
atoi, atol

(6) *Memory Allocation Library Functions (see Note 2)*
malloc, calloc, free, cfree

(7) *Miscellaneous Library Functions*
NOTE 2: To use the I/O library functions and Memory allocation library functions, low level routines must be prepared by the user according to the target system requirements.



■ IN-CIRCUIT EMULATOR FOR HD64180 SERIES

Hitachi's hardware emulator consists of the Adaptive System Emulator (ASE) plus emulator box for the corresponding microprocessor. The emulator supports hardware and software development when connected to a Vax-II, IBM-PC, or PC compatible host machine.

■ ASE FEATURES

- Serial connection to host computer, or console via RS-232C port allows loading, saving, and verifying of user programs.
- Object formats· Intel HEX; and Motorola S.
- Connects to centronics printer.
- Includes 3.5 inch floppy disk drive.

■ EMULATOR BOX FEATURES

- Executes program in realtime from 0.5 MHz to 8 MHz for all emulators except HS180 ABX05H which executes in realtime from 0.5 MHz to 10 MHz.
- Memory:
  —Includes 64-kbyte user memory
  —Expandable to 512 kbytes with an optional memory board (up to 6 MHz without wait states)

■ EMULATOR BOX FUNCTIONS

- Executes user's program loaded in emulator's memory:
  —Realtime
  —Single step
- Breaks on combination of specified number of the following conditions:
  —Program counter (logical or physical address)
  —Access to specified memory area
  —DMAC transfer request or completion
  —Eight external probe signals
- Up to 255 software breakpoints on RAM area
- Multi-break function: In multi-MPU system using several ASEs, an ASE break acts as a trigger which causes other ASEs to break. (HS180ABX05H).
- Sequential break: Analyzes order in which up to 4 software breakpoints were passed (HS180ABX04H/05H).
- Realtime tracing.
  —Stores or displays bus information, external signal, or I/O signals for up to 2048 machine cycles
  —Traces by bus cycle or 125 ns after user program execution stops at breakpoint
  —Trace starting or extracting condition can be specified
- Pseudo-I/O emulation function (HS180ABX04H/05H)
- Disassembler
- Line assembler
- Symbolic debugger
- Execution time measurement
- Displays, sets, changes, or transfers data in memory

Section Two

HD6300, HD6800 8-Bit Microcomputer Family

HD6303R, HD63A03R, HD63B03R CMOS MPU (Micro Processing Unit)

The HD6303R is an 8-bit CMOS micro processing unit which has the completely compatible instruction set with the HD6301V1 128 bytes RAM, Serial Communication Interface (SCI), parallel I/O ports and multi function timer are incorporated in the HD6303R It is bus compatible with HMCS6800 and can be expanded up to 65k bytes Like the HMCS6800 family, I/O level is TTL compatible with +5 0V single power supply As the HD6303R is CMOS MPU, power dissipation is extremely low And also HD6303R has Sleep Mode and Stand-by Mode as lower power dissipation mode. Therefore, flexible low power consumption application is possible.

■ FEATURES

- Object Code Upward Compatible with the HD6800, HD6801, HD6802
- Multiplexed Bus ($D_0/A_0 \sim D_7/A_7 A_8 \sim A_{15}$), Non Multiplexed Bus ($D_0 \sim D_7$, $A_0 \sim A_{15}$)
- Abundant On-Chip Functions Compatible with the HD6301V1; 128 Bytes RAM, 13 Parallel I/O Lines, 16-bit Timer, Serial Communication Interface (SCI)
- Low Power Consumption Mode, Sleep Mode, Stand-By Mode
- Minimum Instruction Execution Time
  1µs (f=1MHz), 0.67µs (f=1 5MHz), 0.5µs (f=2.0MHz)
- Bit Manipulation, Bit Test Instruction
- Error Detecting Function, Address Trap, Op Code Trap
- Up to 65k Bytes Address Space
- Wide Operation Range
  V_{CC}=3 to 6V (f = 0 1 ~ 0.5 MHz)
  f = 0.1 to 2.0 MHz (V_{CC} = 5V ± 10%)

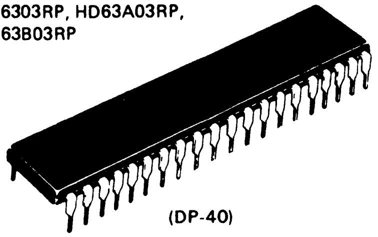
■ TYPE OF PRODUCTS

Type No.	Bus Timing
HD6303R	1.0 MHz
HD63A03R	1.5 MHz
HD63B03R	2.0 MHz

■ PROGRAM DEVELOPMENT SUPPORT TOOLS

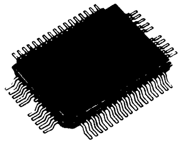
- Cross assembler and C compiler software for IBM PCs and compatibles
- In circuit emulator for use with IBM PCs and compatibles

HD6303RP, HD63A03RP, HD63B03RP



(DP-40)

HD6303RF, HD63A03RF, HD63B03RF

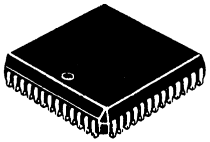


(FP-54)

HD6303RCG, HD63A03RCG, HD63B03RCG

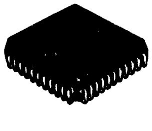
(CG-40)

HD6303RCP, HD63A03RCP HD63B03RCP



(CP-52)

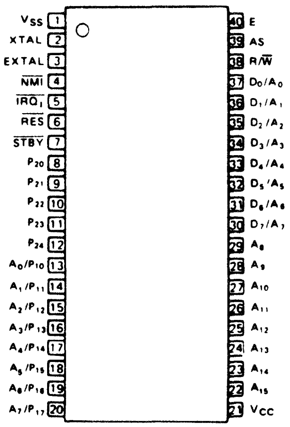
HD6303RL, HD63A03RL HD63B03RL



(CP-44)

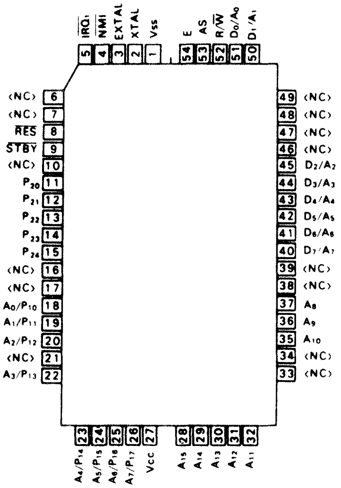
■ PIN ARRANGEMENT

- HD6303RP, HD63A03RP, HD63B03RP



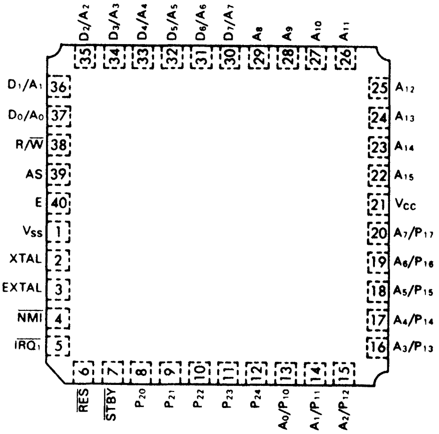
(Top View)

- HD6303RF, HD63A03RF, HD63B03RF



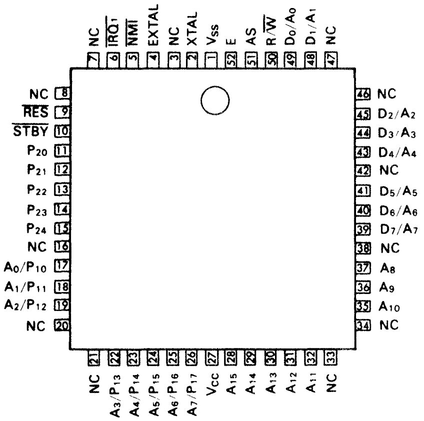
(Top View)

- HD6303RCG, HD63A03RCG, HD63B03RCG



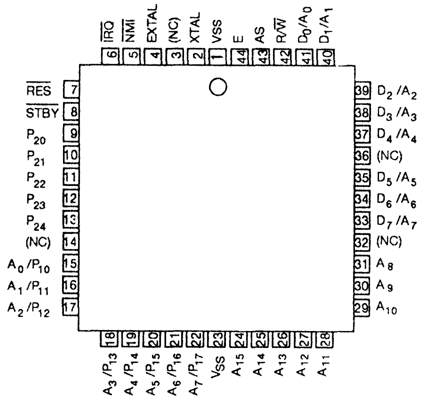
(Top View)

- HD6303RCP, HD63A03RCP, HD63B03RCP



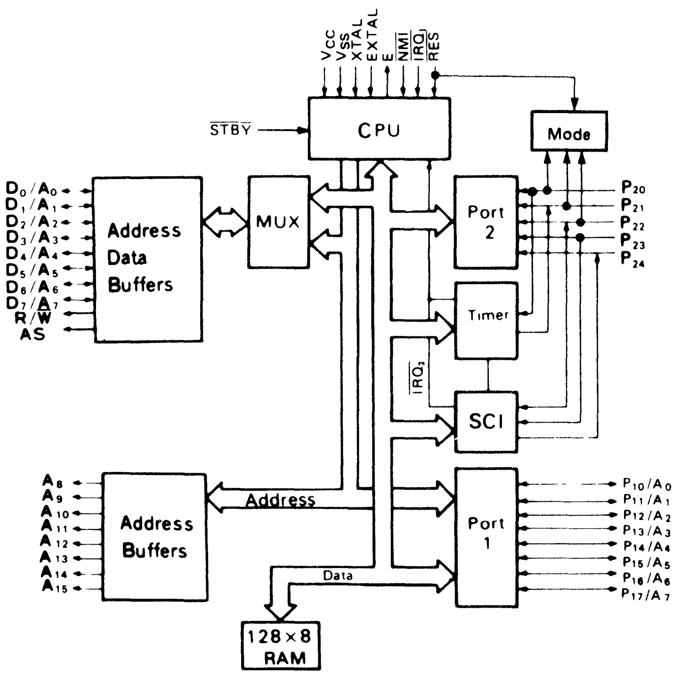
(Top View)

- HD 6303RL, HD63A03RL HD63B03RL



(Top View)

- **BLOCK DIAGRAM**



■ ABSOLUTE MAXIMUM RATINGS

Item	Symbol	Value	Unit
Supply Voltage	V_{CC}	-0.3 ~ +7.0	V
Input Voltage	V_{in}	-0.3 ~ V_{CC}+0.3	V
Operating Temperature	T_{opr}	0 ~ +70	°C
Storage Temperature	T_{stg}	-55 ~ +150	°C

(NOTE) This product has protection circuits in input terminal from high static electricity voltage and high electric field. But be careful not to apply overvoltage more than maximum ratings to these high input impedance protection circuits. To assure the normal operation, we reccmmend V_{in}, V_{out} · $V_{SS} \leqq (V_{in}$ or $V_{out}) \leqq V_{CC}$.

■ ELECTRICAL CHARACTERISTICS

● DC CHARACTERISTICS (V_{CC} = 5.0V±10%, V_{SS} = 0V, Ta = 0~+70°C, unless otherwise noted.)

Item		Symbol	Test Condition	min	typ	max	Unit
Input "High" Voltage	$\overline{RES}$, $\overline{STBY}$	V_{IH}		V_{CC}-0.5	–	V_{CC} +0.3	V
	EXTAL			V_{CC}x0.7	–		
	Other Inputs			2.0	–		
Input "Low" Voltage	All Inputs	V_{IL}		-0.3	–	0.8	V
Input Leakage Current	$\overline{NMI}$, $\overline{IRQ_1}$, $\overline{RES}$, $\overline{STBY}$	$\|I_{in}\|$	V_{in} =0.5~V_{CC}-0.5V	–	–	1.0	μA
Three State (off-state) Leakage Current	P_{10}~P_{17}, P_{20}~P_{24}, D_0~D_7, A_8~A_{15}	$\|I_{TSI}\|$	V_{in} =0.5~V_{CC}-0.5V	–	–	1.0	μA
Output "High" Voltage	All Outputs	V_{OH}	I_{OH} = -200μA	2.4	–	–	V
			I_{OH} = -10μA	V_{CC}-0.7	–	–	V
Output "Low" Voltage	All Outputs	V_{OL}	I_{OL} = 1.6mA	–	–	0.55	V
Input Capacitance	All Inputs	C_{in}	V_{in}=0V, f=1.0MHz, Ta = 25°C	–	–	12.5	pF
Standby Current	Non Operation	I_{CC}	V_{IL} ($\overline{STBY}$) =0~0.6V V_{IH} ($\overline{RES}$) = V_{CC} −0.5~V_{CC}V V_{IL} ($\overline{RES}$) = 0 ~ 0.6V	–	2.0	15.0	μA
Current Dissipation*		I_{CC}	Operating (f=1MHz**)	–	6.0	10.0	mA
			Sleeping (f=1MHz**)	–	1.0	2.0	
RAM Stand-By Voltage		V_{RAM}		2.0	–	–	V

* V_{IH} min = V_{CC}-1.0V, V_{IL} max = 0.8V

** Current Dissipation of the operating or sleeping condition is proportional to the operating frequency So the typ. or max. values about Current Dissipations at f = *x* MHz operation are decided according to the following formula,

typ value (f = *x*MHz) = typ. value (f = 1MHz) x *x*
max. value (f = *x*MHz) = max. value (f = 1MHz) x *x*
(both the sleeping and operating)

- **AC CHARACTERISTICS** (V_{CC} = 5.0V±10%, V_{SS} = 0V, Ta = 0~+70°C, unless otherwise noted.)

BUS TIMING

Item		Symbol	Test Condition	HD6303R			HD63A03R			HD63B03R			Unit
				min	typ	max	min	typ	max	min	typ	max	
Cycle Time		t_{cyc}	Fig. 1 Fig. 2	1	–	10	0.666	–	10	0.5	–	10	μs
Address Strobe Pulse Width * "High"		PW_{ASH}		220	–	–	150	–	–	110	–	–	ns
Address Strobe Rise Time		t_{ASr}		–	–	20	–	–	20	–	–	20	ns
Address Strobe Fall Time		t_{ASf}		–	–	20	–	–	20	–	–	20	ns
Address Strobe Delay Time *		t_{ASD}		60	–	–	40	–	–	20	–	–	ns
Enable Rise Time		t_{Er}		–	–	20	–	–	20	–	–	20	ns
Enable Fall Time		t_{Ef}		–	–	20	–	–	20	–	–	20	ns
Enable Pulse Width "High" Level*		PW_{EH}		450	–	–	300	–	–	220	–	–	ns
Enable Pulse Width "Low" Level*		PW_{EL}		450	–	–	300	–	–	220	–	–	ns
Address Strobe to Enable Delay* Time		t_{ASED}		60	–	–	40	–	–	20	–	–	ns
Address Delay Time		t_{AD1}		–	–	250	–	–	190	–	–	160	ns
		t_{AD2}		–	–	250	–	–	190	–	–	160	ns
Address Delay Time for Latch*		t_{ADL}		–	–	250	–	–	190	–	–	160	ns
Data Set-up Time	Write	t_{DSW}		230	–	–	150	–	–	100	–	–	ns
	Read	t_{DSR}		80	–	–	60	–	–	50	–	–	ns
Data Hold Time	Read	t_{HR}		0	–	–	0	–	–	0	–	–	ns
	Write	t_{HW}		20	–	–	20	–	–	20	–	–	ns
Address Set-up Time for Latch*		t_{ASL}		60	–	–	40	–	–	20	–	–	ns
Address Hold Time for Latch		t_{AHL}		30	–	–	20	–	–	20	–	–	ns
Address Hold Time		t_{AH}		20	–	–	20	–	–	20	–	–	ns
A_0 ~ A_7 Set-up Time Before E*		t_{ASM}		200	–	–	110	–	–	60	–	–	ns
Peripheral Read Access Time	Non-Multiplexed Bus *	(t_{ACCN})		–	–	650	–	–	395	–	–	270	ns
	Multiplexed Bus*	(t_{ACCM})		–	–	650	–	–	395	–	–	270	ns
Oscillator stabilization Time		t_{RC}	Fig. 8	20	–	–	20	–	–	20	–	–	ms
Processor Control Set-up Time		t_{PCS}	Fig. 9	200	–	–	200	–	–	200	–	–	ns

*These timings change in approximate proportion to t_{cyc}. The figures in this characteristics represent those when t_{cyc} is minimum (= in the highest speed operation)

PERIPHERAL PORT TIMING

Item		Symbol	Test Condition	HD6303R			HD63A03R			HD63B03R			Unit
				min	typ	max	min	typ	max	min	typ	max	
Peripheral Data Set-up Time	Port 1, 2	t_{PDSU}	Fig. 3	200	–	–	200	–	–	200	–	–	ns
Peripheral Data Hold Time	Port 1, 2	t_{PDH}	Fig. 3	200	–	–	200	–	–	200	–	–	ns
Delay Time, Enable Negative Transition to Peripheral Data Valid	Port 1, 2*	t_{PWD}	Fig. 4	–	–	300	–	–	300	–	–	300	ns

* Except P_{21}

TIMER, SCI TIMING

Item	Symbol	Test Condition	HD6303R			HD63A03R			HD63B03R			Unit
			min	typ	max	min	typ	max	min	typ	max	
Timer Input Pulse Width	t_{PWT}	Fig. 5	2.0	–	–	2.0	–	–	2.0	–	–	t_{cyc}
Delay Time, Enable Positive Transition to Timer Out	t_{TOD}		–	–	400	–	–	400	–	–	400	ns
SCI Input Clock Cycle	t_{Scyc}		2.0	–	–	2.0	–	–	2.0	–	–	t_{cyc}
SCI Input Clock Pulse Width	t_{PWSCK}		0.4	–	0.6	0.4	–	0.6	0.4	–	0.6	t_{Scyc}

MODE PROGRAMMING

Item	Symbol	Test Condition	HD6303R			HD63A03R			HD63B03R			Unit
			min	typ	max	min	typ	max	min	typ	max	
$\overline{RES}$ "Low" Pulse Width	PW_{RSTL}	Fig. 6	3	–	–	3	–	–	3	–	–	t_{cyc}
Mode Programming Set-up Time	t_{MPS}		2	–	–	2	–	–	2	–	–	t_{cyc}
Mode Programming Hold Time	t_{MPH}		150	–	–	150	–	–	150	–	–	ns

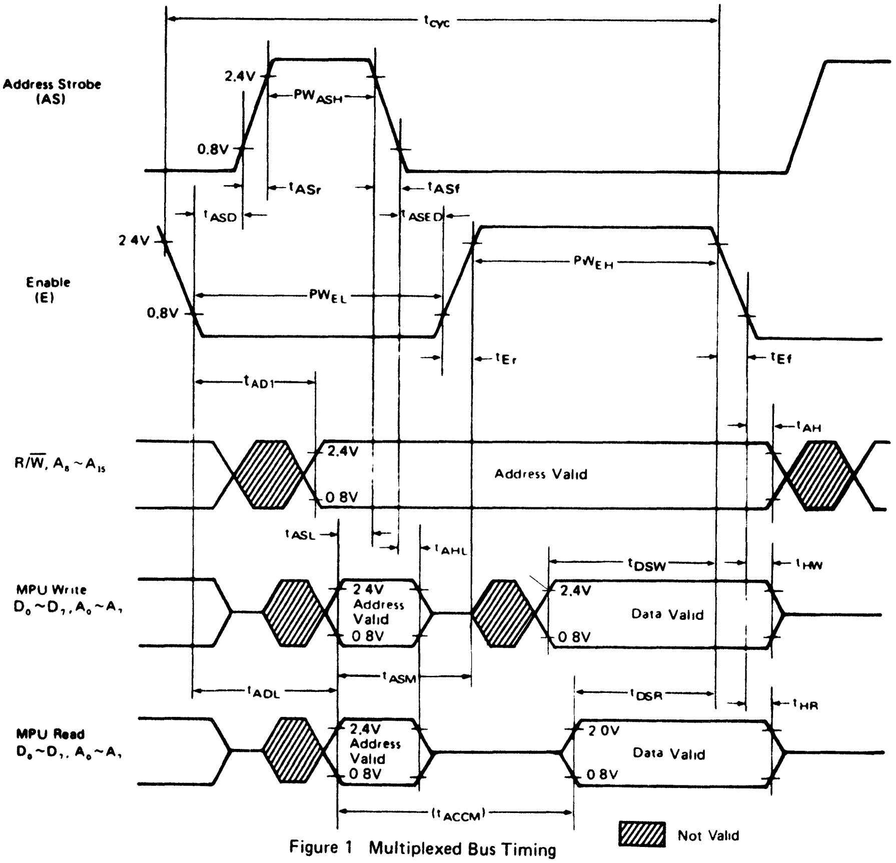


Figure 1 Multiplexed Bus Timing

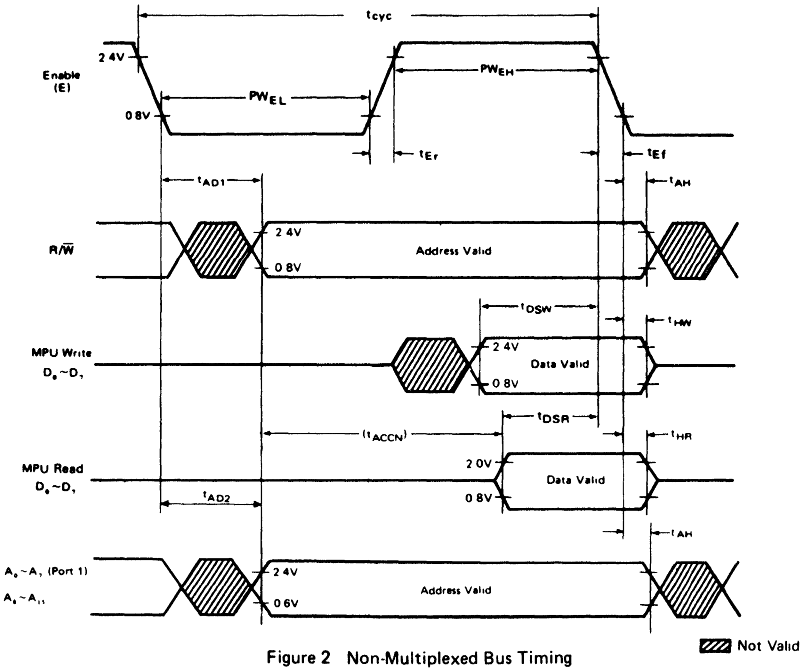


Figure 2 Non-Multiplexed Bus Timing

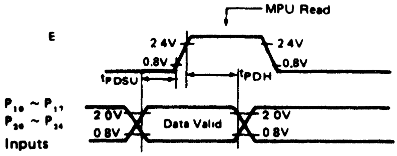
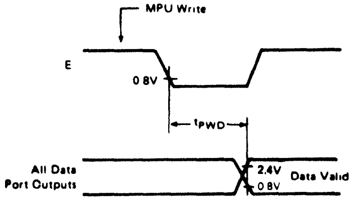


Figure 3 Port Data Set-up and Hold Times (MPU Read)



Note) Port 2: Except P_{21}

Figure 4 Port Data Delay Times (MPU Write)

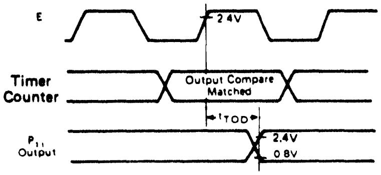


Figure 5 Timer Output Timing

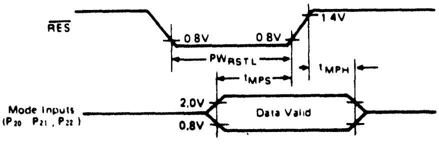


Figure 6 Mode Programming Timing

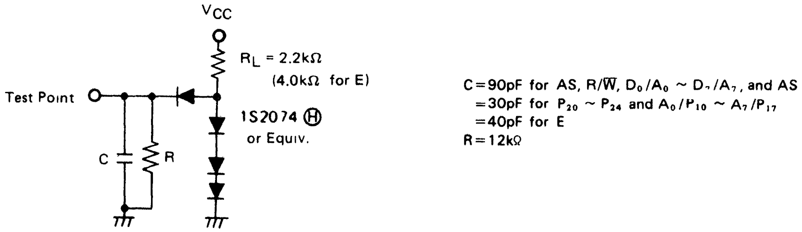

VCC
RL = 2.2kΩ
(4.0kΩ for E)
Test Point
1S2074 Ⓗ
or Equiv.
C
R
C=90pF for AS, R/W̄, D0/A0 ~ D7/A7, and AS
=30pF for P20 ~ P24 and A0/P10 ~ A7/P17
=40pF for E
R=12kΩ

Figure 7 Bus Timing Test Loads (TTL Load)

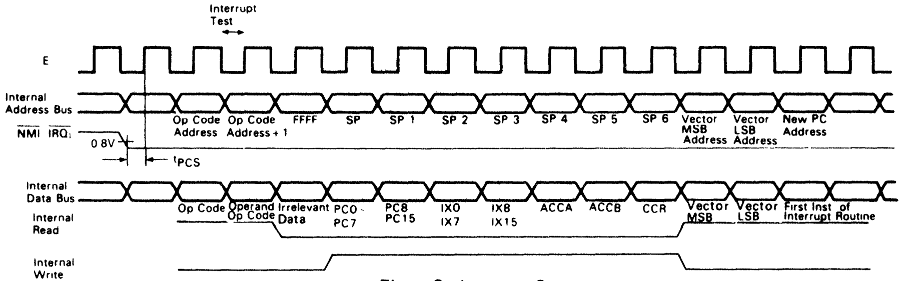


Figure 8 Interrupt Sequence

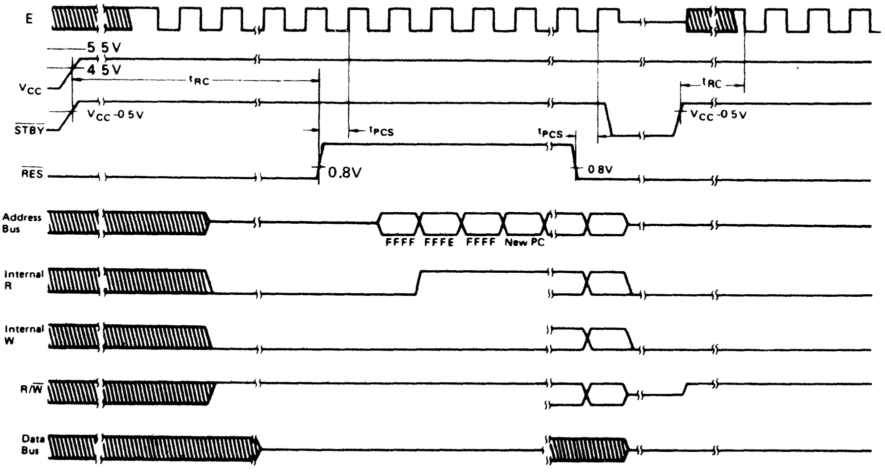


Figure 9 Reset Timing

■ FUNCTIONAL PIN DESCRIPTION

• V_{CC}, V_{SS}

These two pins are used for power supply and GND. Recommended power supply voltage is 5V ± 10%. 3. to 6V can be used for low speed operation (100 ~ 500 kHz).

• XTAL, EXTAL

These two pins are connected with parallel resonant fundamental crystal, AT cut. For instance, in order to obtain the system clock 1MHz, a 4MHz resonant fundamental crystal is used because the devide-by-4 circuitry is included. An example of the crystal interface is shown in Fig. 10. EXTAL accepts an external clock input of duty 45% to 55% to drive. For external clock, XTAL pin should be open. The crystal and capacitors should be mounted as close as possible to the pins.

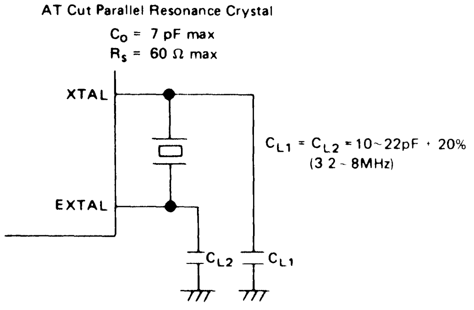

AT Cut Parallel Resonance Crystal
C_0 = 7 pF max
R_s = 60 Ω max
XTAL
EXTAL
C_{L1} = C_{L2} = 10~22pF ± 20%
(3.2~8MHz)
C_{L2}
C_{L1}

Figure 10 Crystal Interface

- **Standby ($\overline{STBY}$)**

This pin is used to place the MPU in the standby mode If this goes to "Low" level, the oscillation stops, the internal clock is tied to V_{SS} or V_{CC} and the MPU is reset. In order to retain information in RAM during standby, write "0" into RAM enable bit (RAME) RAME is bit 6 of the RAM Control Register at address $0014 This disables the RAM, so the contents of RAM is guaranteed For details of the standby mode, see the Standby section

- **Reset ($\overline{RES}$)**

This input is used to reset the MPU. $\overline{RES}$ must be held "Low" for at least 20ms when the power starts up. It should be noted that, before clock generator stabilize, the internal state and I/O ports are uncertain, because MPU can not be reset without clock. To reset the MPU during system operation, it must be held "Low" for at least 3 system clock cycles From the third cycle, all address buses become "high-impedance" and it continues while $\overline{RES}$ is "Low" If $\overline{RES}$ goes to "High", CPU does the following.

(1) I/O Port 2 bits 2,1,0 are latched into bits PC2, PC1, PC0 of program control register.

(2) The contents of the two Start Addresses, $FFFE, $FFFF are brought to the program counter, from which program starts (see Table 1).

(3) The interrupt mask bit is set In order to have the CPU recognize the maskable interrupts $\overline{IRQ_1}$ and $\overline{IRQ_2}$, clear it before those are used

- **Enable (E)**

This output pin supplies system clock. Output is a single-phase, TTL compatible and 1/4 the crystal oscillation frequency. It will drive two LS TTL load and 40pF capacitance

- **Non Maskable Interrupt ($\overline{NMI}$)**

When the falling edge of the input signal of this pin is recognized, NMI sequence starts. The current instruction is continued to complete, even if $\overline{NMI}$ signal is detected. Interrupt mask bit in Condition Code Register has no effect on NMI detection. In response to NMI interrupt, the information of Program Counter, Index Register, Accumulators, and Condition Code Register are stored on the stack On completion of this sequence, vectoring address $FFFC and $FFFD are generated to load the contents to the program counter. Then the CPU branch to a non maskable interrupt service routine

- **Interrupt Request ($\overline{IRQ_1}$)**

This level sensitive input requests a maskable interrupt sequence. When $\overline{IRQ_1}$ goes to "Low", the CPU waits until it completes the current instruction that is being executed. Then, if the interrupt mask bit in Condition Code Register is not set, CPU begins interrupt sequence, otherwise, interrupt request is neglected

Once the sequence has started, the information of Program Counter, Index Register, Accumulator, Condition Code Register are stored on the stack Then the CPU sets the interrupt mask bit so that no further maskable interrupts may be responded

Table 1 Interrupt Vectoring memory map

	Vector		Interrupt
	MSB	LSB	
Highest Priority	FFFE	FFFF	$\overline{RES}$
	FFEE	FFEF	TRAP
	FFFC	FFFD	$\overline{NMI}$
	FFFA	FFFB	Software Interrupt (SWI)
	FFF8	FFF9	$\overline{IRQ_1}$ (or $\overline{IS3}$)
	FFF6	FFF7	ICF (Timer Input Capture)
	FFF4	FFF5	OCF (Timer Output Compare)
	FFF2	FFF3	TOF (Timer Overflow)
Lowest Priority	FFF0	FFF1	SCI (RDRF + ORFE + TDRE)

At the end of the cycle, the CPU generates 16 bit vectoring addresses indicating memory addresses $FFF8 and $FFF9, and loads the contents to the Program Counter, then branch to an interrupt service routine.

The Internal Interrupt will generate signal ($\overline{IRQ_2}$) which is quite the same as $\overline{IRQ_1}$ except that it will use the vector address $FFF0 to $FFF7

When $\overline{IRQ_1}$ and $\overline{IRQ_2}$ are generated at the same time, the former precedes the latter Interrupt Mask Bit in the condition code register, if being set, will keep the both interrupts off.

On occurrence of Address error or Op-code error, TRAP interrupt is invoked This interrupt has priority next to $\overline{RES}$. Regardless of the interrupt Mask Bit condition, the CPU will start an interrupt sequence The vector for this interrupt will be $FFEE, $FFEF

- **Read/Write (R/$\overline{W}$)**

This TTL compatible output signals peripheral and memory devices whether CPU is in Read ("High"), or in Write ("Low"). The normal stand-by state is Read ("High") Its output will drive one TTL load and 90pF capacitance

- **Address Strobe (AS)**

In the multiplexed mode, address strobe signal appears at this pin It is used to latch the lower 8 bits addresses multiplexed with data at D_0/A_0 ~ D_7/A_7. The 8-bit latch is controlled by address strobe as shown in Figure 15 Thereby, D_0/A_0 ~ D_7/A_7 can become data bus during E pulse. The timing chart of this signal is shown in Figure 1

Address Strobe (AS) is sent out even if the internal address is accessed.

■ PORTS

There are two I/O ports on HD6303R MPU (one 8-bit ports and one 5-bit port). Each port has an independent write-only data direction register to program individual I/O pins for input or output.*

When the bit of associated Data Direction Register is "1", I/O pin is programmed for output, if "0", then programmed for an input.

There are two ports: Port 1, Port 2. Addresses of each port and associated Data Direction Register are shown in Table 2.

* Only one exception is bit 1 of Port 2 which becomes either a data input or a timer output. It cannot be used as an output port.

Table 2 Port and Data Direction Register Addresses

Ports	Port Address	Data Direction Register Address
I/O Port 1	$0002	$0000
I/O Port 2	$0003	$0001

- **I/O Port 1**

This is an 8-bit port, each bit being defined individually as input or outputs by associated Data Direction Register. The 8-bit output buffers have three-state capability, maintaining in high impedance state when they are used for input. In order to be read accurately, the voltage on the input lines must be more than 2.0V for logic "1" and less than 0.8V for logic "0".

These are TTL compatible. After the MPU has been reset, all I/O lines are configured as inputs in Multiplexed mode. In Non Multiplexed mode, Port 1 will be output line for lower order address lines ($A_0 \sim A_7$), which can drive one TTL load and 30 pF capacitance.

- **I/O Port 2**

This port has five lines, whose I/O direction depends on its data direction register. The 5-bit output buffers have three-state capability, going high impedance state when used as inputs. In order to be read accurately, the voltage on the input lines must be more than 2.0V for logic "1" and less than 0.8V for logic "0". After the MPU has been reset, I/O lines are configured as inputs These pins on Port 2 ($P_{20} \sim P_{22}$ of the chip) are used to program the mode of operation during reset. The values of these three pins during reset are latched into the upper 3 bits (bit 7, 6 and 5) of Port 2 Data Register which is explained in the MODE SELECTION section

In all modes, Port 2 can be configured as I/O lines. This port also provides access to the Serial I/O and the Timer. However, note that bit 1 (P_{21}) is the only pin restricted to data input or Timer output.

- **BUS**
- **$D_0/A_0 \sim D_7/A_7$**

This TTL compatible three-state buffer can drive one TTL load and 90 pF capacitance

Non Multiplexed Mode

In this mode, these pins become only data bus ($D_0 \sim D_7$)

Multiplexed Mode

These pins becomes both the data bus ($D_0 \sim D_7$) and lower bits of the address bus ($A_0 \sim A_7$) An address strobe output is "High" when the address is on the pins.

- **$A_8 \sim A_{15}$**

Each line is TTL compatible and can drive one TTL load and 90 pF capacitance After reset, these pins become output for upper order address lines ($A_8 \sim A_{15}$)

- **MODE SELECTION**

The operation mode after the reset must be determined by the user wiring the P_{20}, P_{21}, and P_{22} externally. These three pins are lower order bits; I/O 0, I/O 1, I/O 2 of Port 2. They are latched into the control bits PC0, PC1, PC2 of I/O Port 2 register when $\overline{\text{RES}}$ goes "High". I/O Port 2 Register is shown below.

Port 2 DATA REGISTER

	7	6	5	4	3	2	1	0
$0003	PC2	PC1	PC0	I/O 4	I/O 3	I/O 2	I/O 1	I/O 0

An example of external hardware used for Mode Selection is shown in Figure 11. The HD14053B is used to separate the peripheral device from the MPU during reset. It is necessary if the data may conflict between peripheral device and Mode generation circuit.

No mode can be changed through software because the bits 5, 6, and 7 of Port 2 Data Register are read-only The mode selection of the HD6303R is shown in Table 3.

The HD6303R operates in two basic modes: (1) Multiplexed Mode, (2) Non Multiplexed Mode.

- **Multiplexed Mode**

The data bus and the lower order address bus are multiplexed in the $D_0/A_0 \sim D_7/A_7$ and can be separated by the Address Strobe

Port 2 is configured for 5 parallel I/O or Serial I/O, or Timer, or any combination thereof. Port 1 is configured for 8 parallel I/O

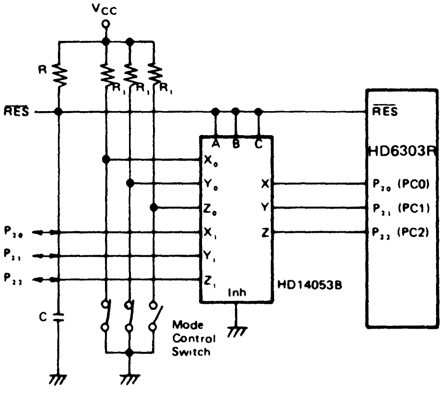
- **Non Multiplexed Mode**

In this mode, the HD6303R can directly address HMCS6800 peripherals with no address latch. $D_0/A_0 \sim D_7/A_7$ become a data bus and Port 1 becomes $A_0 \sim A_7$ address bus.

In this mode, the HD6303R is expandable up to 65k bytes with no address latch.

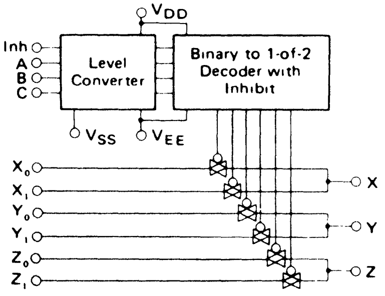
- **Lower Order Address Bus Latch**

Because the data bus is multiplexed with the lower order address bus in $D_0/A_0 \sim D_7/A_7$ in the multiplexed mode, address bits must be latched. It requires the 74LS373 Transparent octal D-type to latch the LSB. Latch connection of the HD6303R is shown in Figure 15.



Note 1) Figure of Multiplexed Mode
2) RC≈Reset Constant
3) $R_1 = 10k\Omega$

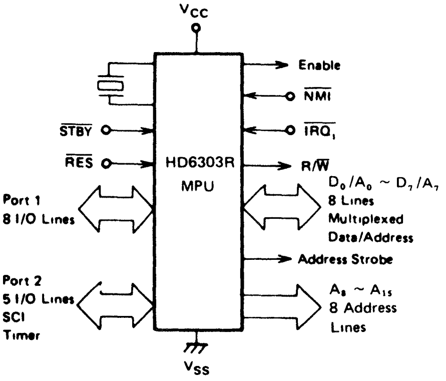
Figure 11 Recommended Circuit for Mode Selection



Truth Table

Control Input				On Switch		
Inhibit	Select					
	C	B	A	HD14053B		
0	0	0	0	Z_0	Y_0	X_0
0	0	0	1	Z_0	Y_0	X_1
0	0	1	0	Z_0	Y_1	X_0
0	0	1	1	Z_0	Y_1	X_1
0	1	0	0	Z_1	Y_0	X_0
0	1	0	1	Z_1	Y_0	X_1
0	1	1	0	Z_1	Y_1	X_0
0	1	1	1	Z_1	Y_1	X_1
1	X	X	X			

Figure 12 HD14053B Multiplexers/De-Multiplexers



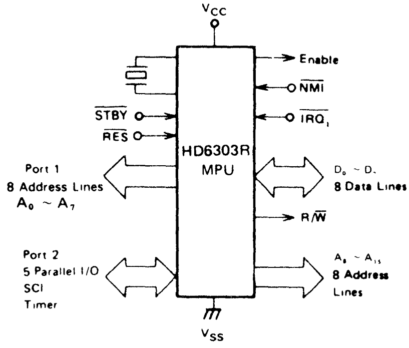
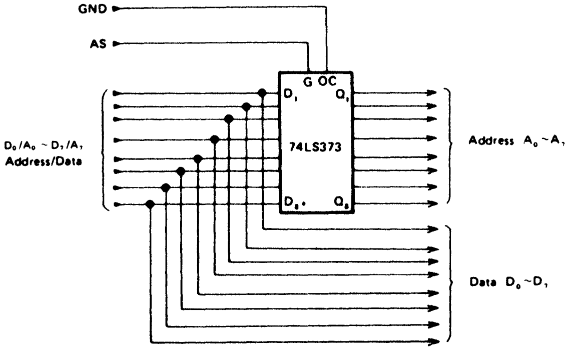


Figure 13 HD6303R MPU Multiplexed Mode

Figure 14 HD6303R MPU Non Multiplexed Mode



Function Table

Output Control (OC)	Enable G	Enable D	Output Q
L	H	H	H
L	H	L	L
L	L	X	Q_0
H	X	X	Z

Figure 15 Latch Connection

Table 3 Mode Selection

Operating Mode	P_{20}	P_{21}	P_{22}
Multiplexed Mode	L	H	L
	L	L	H
Non Multiplexed Mode	H	L	L

L: logic "0"
H: logic "1"

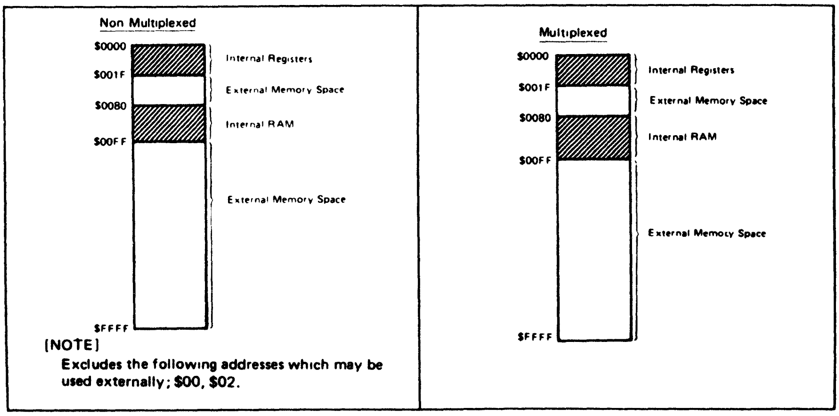
■ MEMORY MAP

The MPU can provide up to 65k byte address space. Figure 16 shows a memory map for each operating mode. The first 32 locations of each map are for the CPU's internal register only, as shown in Table 4.

Table 4 Internal Register Area

Register	Address
Port 1 Data Direction Register**	00*
Port 2 Data Direction Register**	01
Port 1 Data Register	02*
Port 2 Data Register	03
Timer Control and Status Register	08
Counter (High Byte)	09
Counter (Low Byte)	0A
Output Compare Register (High Byte)	0B
Output Compare Register (Low Byte)	0C
Input Capture Register (High Byte)	0D
Input Capture Register (Low Byte)	0E
Rate and Mode Control Register	10
Transmit/Receive Control and Status Register	11
Receive Data Register	12
Transmit Data Register	13
RAM Control Register	14
Reserved	15-1F

* External address in Non Multiplexed Mode
** 1 = Output, 0 = Input



[NOTE]
Excludes the following addresses which may be used externally; $00, $02.

Figure 16 HD6303R Memory Maps

■ PROGRAMMABLE TIMER

The HD6303R contains 16-bit programmable timer which may measure input waveform.In addition to that it can generate an output waveform by itself. For both input and output waveform, the pulse width may vary from a few microseconds to several seconds

The timer hardware consists of

- an 8-bit control and status register
- a 16-bit free running counter
- a 16-bit output compare register
- a 16-bit input capture register

A block diagram of the timer is shown in Figure 17.

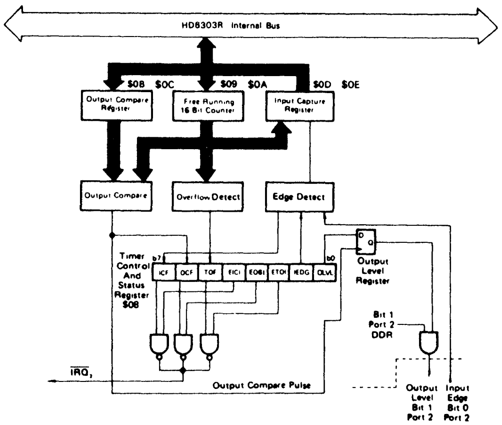


Figure 17 Programmable Timer Block Diagram

● Free Running Counter ($0009: $000A)

The key element in the programmable timer is a 16-bit free running counter, that is driven by an E (Enable) clock to increment its values. The counter value will be read out by the CPU software at any time with no effects on the counter. Reset will clear the counter.

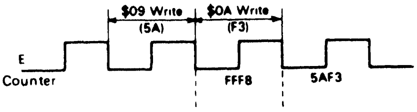
When the MSB of this counter is read, the LSB is stored in temporary latch. The data is fetched from this latch by the subsequent read of LSB. Thus consistent double byte data can be read from the counter.

When the CPU writes arbitrary data to the MSB ($09), the value of $FFF8 is being pre-set to the counter ($09, $0A) regardless of the write data value. Then the CPU writes arbitrary data to the LSB ($0A), the data is set to the "Low" byte of the counter, at the same time, the data preceedingly written in the MSB ($09) is set to "High" byte of the counter.

When the data is written to this counter, a double byte store instruction (ex. STD) must be used. If only the MSB of counter is written, the counter is set to $FFF8.

The counter value written to the counter using the double byte store instruction is shown in Figure 18.

To write to the counter can disturb serial operations, so it should be inhibited during using the SCI. If external clock mode is used for SCI, this will not disturb serial operations.



(5AF3 written to the counter)

Figure 18 Counter Write Timing

● Output Compare Register ($000B:$000C)

This is a 16-bit read/write register which is used to control an output waveform The contents of this register are constantly being compared with current value of the free running counter.

When the contents match with the value of the free running counter, a flag (OCF) in the timer control/status register (TCSR) is set and the current value of an output level Bit (OLVL) in the TCSR is transferred to Port 2 bit 1. When bit 1 of the Port 2 data direction register is "1" (output), the OLVL value will appear on the bit 1 of Port 2. Then, the value of Output Compare Register and Output level bit may be changed for the next compare.

The output compare register is set to $FFFF during reset. The compare function is inhibited at the cycle of writing to the high byte of the output compare register and at the cycle just after that to ensure valid compare. It is also inhibited in same manner at writing to the free running counter.

In order to write a data to Output Compare Register, a double byte store instruction (ex.STD) must be used.

● Input Capture Register ($000D: $000E)

The input capture register is a 16-bit read-only register used to hold the current value of free running counter captured when the proper transition of an external input signal occurs.

The input transition change required to trigger the counter transfer is controlled by the input edge bit (IEDG).

To allow the external input signal to go in the edge detect unit, the bit of the Data Direction Register corresponding to bit 0 of Port 2 must have been cleared (to zero).

To insure input capture in all cases, the width of an input pulse requires at least 2 Enable cycles.

● Timer Control/Status Register (TCSR) ($0008)

This is an 8-bit register. All 8-bits are readable and the lower 5 bits may be written. The upper 3 bits are read-only, indicating the timer status information as is shown below.

(1) A proper transition has been detected on the input pin (ICF).

(2) A match has been found between the value in the free running counter and the output compare register (OCF).

(3) When counting up to $0000 (TOF).

Each flag has an individual enable bit in TCSR which determines whether or not an interrupt request may occur ($\overline{IRQ_2}$). If the I-bit in Condition Code Register has been cleared, a prior vectored address occurs corresponding to each flag. A description of each bit is as follows.

Timer Control / Status Register

7	6	5	4	3	2	1	0	
ICF	OCF	TOF	EICI	EOCI	ETOI	IEDG	OLVL	$0008

Bit 0 OLVL (Output Level); When a match is found in the value between the counter and the output com-

pare register, this bit is transferred to the Port 2 bit 1. If the DDR corresponding to Port 2 bit 1 is set "1", the value will appear on the output pin of Port 2 bit 1.

Bit 1 IEDG (Input Edge): This bit control which transition of an input of Port 2 bit 0 will trigger the data transfer from the counter to the input capture register. The DDR corresponding to Port 2 bit 0 must be clear in advance of using this function.
When IEDG = 0, trigger takes place on a negative edge ("High"-to-"Low" transition). When IEDG = 1, trigger takes place on a positive edge ("Low"-to-"High" transition).

Bit 2 ETOI (Enable Timer Overflow Interrupt); When set, this bit enables TOF interrupt to generate the interrupt request ($\overline{IRQ_2}$). When cleared, the interrupt is inhibited.

Bit 3 EOCI (Enable Output Compare Interrupt); When set, this bit enables OCF interrupt to generate the interrupt request ($\overline{IRQ_2}$). When cleared, the interrupt is inhibited.

Bit 4 EICI (Enable Input Capture Interrupt); When set, this bit enables ICF interrupt to generate the interrupt request ($\overline{IRQ_2}$). When cleared, the interrupt is inhibited.

Bit 5 TOF (Timer Over Flow Flag); This read-only bit is set at the transition of $FFFF to $0000 of the counter. It is cleared by CPU read of TCSR (with TOF set) followed by a CPU read of the counter ($0009).

Bit 6 OCF (Output Compare Flag); This read-only bit is set when a match is found in the value between the output compare register and the counter. It is cleared by a read of TCSR (with OCF set) followed by a CPU write to the output compare register ($000B or $000C).

Bit 7 ICF (Input Capture Flag); The read-only bit is set by a proper transition on the input, and is cleared by a read of TCSR (with ICF set) followed by a CPU read of Input Capture Register ($000D).

Reset will clear each bit of Timer Control and Status Register.

■ SERIAL COMMUNICATION INTERFACE

The **HD6303R** contains a full-duplex asynchronous Serial Communication Interface (SCI). SCI may select the several kinds of the data rate It consists of a transmitter and a receiver which operate independently but with the same data format and the same data rate. Both of transmitter and receiver communicate with the CPU via the data bus and with the outside world through Port 2 bit 2, 3 and 4 Description of hardware, software and register is as follows.

● Wake-Up Feature

In typical multiprocessor applications the software protocol will usually have the designated address at the initial byte of the message. The purpose of Wake-Up feature is to have the non-selected MPU neglect the remainder of the message. Thus the non-selected MPU can inhibit the all further interrupt process until the next message begins

Wake-Up feature is re-enabled by a ten consecutive "1"s which indicates an idle transmit line Therefore software protocol must put an idle period between the messages and must prevent it within the message

With this hardware feature, the non-selected MPU is re-enabled (or "waked-up") by the next message.

● Programmable Options

The HD6303R has the following programmable features.
- data format; standard mark/space (NRZ)
- clock source, external or internal
- baud rate; one of 4 rates per given E clock frequency or 1/8 of external clock
- wake-up feature, enabled or disabled
- interrupt requests; enabled or masked individually for transmitter and receiver
- clock output; internal clock enabled or disabled to Port 2 bit 2
- Port 2 (bits 3, 4), dedicated or not dedicated to serial I/O individually

● Serial Communication Hardware

The serial communications hardware is controlled by 4 registers as shown in Figure 19. The registers include:
- an 8-bit control/status register
- a 4-bit rate/mode control register (write-only)
- an 8-bit read-only receive data register
- an 8-bit write-only transmit data register

Besides these 4 registers, Serial I/O utilizes Port 2 bit 3 (input) and bit 4 (output). Port 2 bit 2 can be used when an option is selected for the internal-clock-out or the external-clock-in.

● Transmit/Receive Control Status Register (TRCSR)

TRCS Register consists of 8 bits which all may be read while only bits 0 to 4 may be written. The register is initialized to $20 on $\overline{RES}$. The bits of the TRCS Register are explained below.

Transmit / Receive Control Status Register

7	6	5	4	3	2	1	0	
RDRF	ORFE	TDRE	RIE	RE	TIE	TE	WU	ADDR $0011

Bit 0 WU (Wake Up); Set by software and cleared by hardware on receipt of ten consecutive "1"s. While this bit is "1", RDRF and ORFE flags are not set even if data are received or errors are detected. Therefore received data are ignored. It should be noted that RE flag must have already been set in advance of WU flag's set.

Bit 1 TE (Transmit Enable); This bit enables transmitter. When this bit is set, bit 4 of Port 2 DDR is also forced to be set. It remains set even if TE is cleared. Preamble of ten consecutive "1"s is transmitted just after this bit is set, and then transmitter becomes ready to send data. If this bit is cleared, the transmitter is disabled and serial I/O affects nothing on Port 2 bit 4.

Bit 2 TIE (Transmit Interrupt Enable); When this bit is set, TDRE (bit 5) causes an $\overline{IRQ_2}$ interrupt. When cleared, TDRE interrupt is masked.

Bit 3 RE (Receive Enable); When set, Port 2 bit 3 can be used as an input of receive regardless of DDR value for this bit When cleared, the receiver is disabled.

Bit 4 RIE (Receive Interrupt Enable); When this bit is set, RDRF (bit 7) or ORFE (bit 6) cause an $\overline{IRQ_2}$ interrupt. When cleared, this interrupt is masked.

Bit 5 TDRE (Transmit Data Register Empty); When the data is transferred from the Transmit Data Register to Output Shift Register, this bit is set by hardware. The bit is cleared by reading the status register followed by writing the next new data into the Transmit Data Register. TDRE is initialized to 1 by $\overline{\text{RES}}$.

Bit 6 ORFE (Over Run Framing Error); When overrun or framing error occurs (receive only), this bit is set by hardware. Over Run Error occurs if the attempt is made to transfer the new byte to the receive data register while the RDRF is "1". Framing Error occurs when the bit counter is not synchronized with the boundary of the byte in the receiving bit stream. When Framing Error is detected, RDRF is not set. Therefore Framing Error can be distinguished from Overrun Error. That is, if ORFE is "1" and RDRF is "1", Overrun Error is detected. Otherwise Framing Error occurs. The bit is cleared by reading the status register followed by reading the receive data register, or by $\overline{\text{RES}}$.

Bit 7 RDRF (Receive Data Register Full); This bit is set by hardware when the data is transferred from the receive shift register to the receive data register. It is cleared by reading the status register followed by reading the receive data register, or by $\overline{\text{RES}}$.

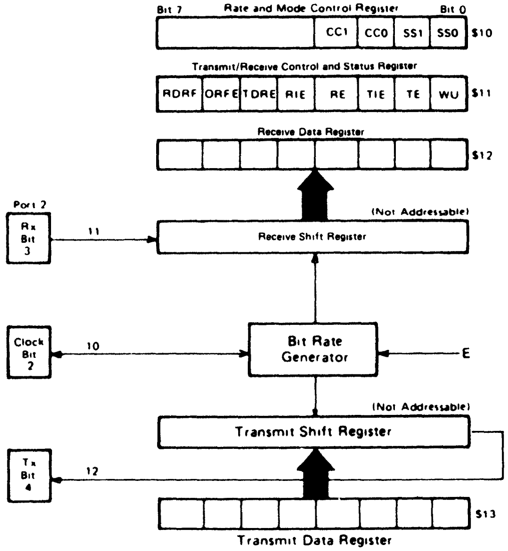


Figure 19 Serial I/O Register

7	6	5	4	3	2	1	0	
X	X	X	X	CC1	CC0	SS1	SS0	ADDR $0010

Transfer Rate / Mode Control Register

Table 5 SCI Bit Times and Transfer Rates

		XTAL	2 4576 MHz	4 0 MHz	4 9152MHz
SS1	SS0	E	614 4 kHz	1 0 MHz	1 2288MHz
0	0	E - 16	26 μs/38,400 Baud	16 μs/62,500 Baud	13 μs/76,800Baud
0	1	E - 128	208μs/4,800 Baud	128 μs/7812.5 Baud	104 2μs/ 9,600Baud
1	0	E - 1024	1 67ms/600 Baud	1 024ms/976.6 Baud	833 3μs/ 1,200Baud
1	1	E - 4096	6 67ms/150 Baud	4 096ms/244 1 Baud	3 333ms/ 300Baud

Table 6 SCI Format and Clock Source Control

CC1 CC0	Format	Clock Source	Port 2 Bit 2	Port 2 Bit 3	Port 2 Bit 4
0 0	–	–	–	–	–
0 1	NRZ	Internal	Not Used***	**	**
1 0	NRZ	Internal	Output*	**	**
1 1	NRZ	External	Input	**	**

* Clock output is available regardless of values for bits RE and TE
** Bit 3 is used for serial input if RE = "1" in TRCS
Bit 4 is used for serial output if TE = "1" in TRCS
*** This pin can be used as I/O port

- **Transfer Rate/Mode Control Register (RMCR)**

The register controls the following serial I/O functions
- Bauds rate •data format • clock source
- Port 2 bit 2 feature

It is 4-bit write-only register, cleared by $\overline{\text{RES}}$. The 4 bits are considered as a pair of 2-bit fields The lower 2 bits control the bit rate of internal clock while the upper 2 bits control the format and the clock select logic

Bit 0 SS0 }
Bit 1 SS1 } Speed Select

These bits select the Baud rate for the internal clock The rates selectable are function of E clock frequency of the CPU. Table 5 lists the available Baud Rates

Bit2 CC0 }
Bit 3 CC1 } Clock Control/Format Select

They control the data format and the clock select logic Table 6 defines the bit field

- **Internally Generated Clock**

If the user wish to use externally an internal clock of the serial I/O, the following requirements should be noted
- CC1, CC0 must be set to "10"
- The maximum clock rate must be E/16.
- The clock rate is equal to the bit rate
- The values of RE and TE have no effect

- **Externally Generated Clock**

If the user wish to supply an external clock to the Serial I/O, the following requirements should be noted
- The CC1, CC0 must be set to "11" (See Table 6).
- The external clock must be set to 8 times of the desired baud rate
- The maximum external clock frequency is E/2 clock

- **Serial Operations**

The serial I/O hardware must be initialized by the software before operation The sequence will be normally as follows
- Writing the desired operation control bits of the Rate and Mode Control Register
- Writing the desired operation control bits of the TRCS register

If Port 2 bit 3, 4 are used for serial I/O, TE, RE bits may be kept set When TE, RE bit are cleared during SCI operation, and subsequently set again, it should be noted that TE, RE must be kept "0" for at least one bit time of the current baud rate If TE, RE are set again within one bit time, there may be the case where the initializing of internal function for transmitter and receiver does not take place correctly

- **Transmit Operation**

Data transmission is enabled by the TE bit in the TRCS register When set, the output of the transmit shift register is connected with Port 2 bit 4 which is unconditionally configured as an output

After $\overline{\text{RES}}$, the user should initialize both the RMC register and the TRCS register for desired operation Setting the TE bit causes a transmission of ten-bit preamble of "1"s Following the preamble, internal synchronization is established and the transmitter is ready to operate Then either of the following states exists

(1) If the transmit data register is empty (TDRE = 1), the consecutive "1"s are transmitted indicating an idle states
(2) If the data has been loaded into the Transmit Data Register (TDRE = 0), it is transferred to the output shift register and data transmission begins.

During the data transfer, the start bit ("0") is first transferred Next the 8-bit data (beginning at bit 0) and finally the stop bit ("1") When the contents of the Transmit Data Register is transferred to the output shift register, the hardware sets the TDRE flag bit If the CPU fails to respond to the flag within the proper time, TDRE is kept set and then a continuous string of 1's is sent until the data is supplied to the data register

- **Receive Operation**

The receive operation is enabled by the RE bit The serial input is connected with Port 2 bit 3. The receiver operation is determined by the contents of the TRCS and RMC register The received bit stream is synchronized by the first "0" (start bit) During 10-bit time, the data is strobed approximately at the center of each bit If the tenth bit is not "1" (stop bit), the system assumes a framing error and the ORFE is set

If the tenth bit is "1", the data is transferred to the receive data register, and the RDRF flag is set. If the tenth bit of the next data is received and still RDRF is preserved set, then ORFE is set indicating that an overrun error has occurred

After the CPU read of the status register as a response to RDRF flag or ORFE flag, followed by the CPU read of the receive data register, RDRF or ORFE will be cleared.

■ RAM CONTROL REGISTER

The register assigned to the address $0014 gives a status information about standby RAM

RAM Control Register

	7	6	5	4	3	2	1	0
$0014	STBY PWR	RAME	X	X	X	X	X	X

Bit 0 Not used.
Bit 1 Not used.
Bit 2 Not used.

HITACHI

Bit 3 Not used.
Bit 4 Not used.
Bit 5 Not used.
Bit 6 RAM Enable (RAME).

Using this control bit, the user can disable the RAM RAM Enable bit is set on the positive edge of $\overline{\text{RES}}$ and RAM is enabled. The program can write "1" or "0" If RAME is cleared, the RAM address becomes external address and the CPU may read the data from the outside memory.

Bit 7 Standby Power Bit (STBY PWR)

This bit can be read or written by the user program. It is cleared when the V_{CC} voltage is removed. Normally this bit is set by the program before going into stand-by mode When the CPU recovers from stand-by mode, this bit should be checked If it is "1", the data of the RAM is retained during stand-by and it is valid.

■ GENERAL DESCRIPTION OF INSTRUCTION SET

The HD6303R has an upward object code compatible with the HD6801 to utilize all instruction sets of the HMCS6800 The execution time of the key instruction is reduced to increase the system through-put In addition, the bit operation instruction, the exchange instruction between the index and the accumulator, the sleep instruction are added. This section describes

- CPU programming model (See Fig. 20)
- Addressing modes
- Accumulator and memory manipulation instructions (See Table 7)
- New instructions
- Index register and stack manipulation instructions (See Table 8)
- Jump and branch instructions (See Table 9)
- Condition code register manipulation instructions (See Table 10)
- Op-code map (See Table 11)
- Cycle-by-cycle operation (See Table 12)

● CPU Programming Model

The programming model for the HD6303R is shown in Figure 20. The double accumulator is physically the same as the accumulator A concatenated with the accumulator B, so that the contents of A and B is changed with executing operation of an accumulator D.

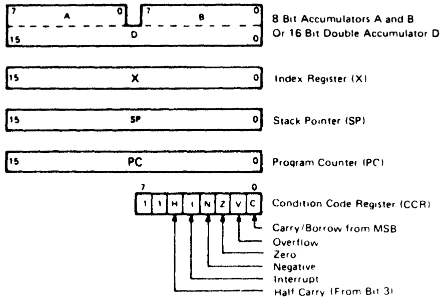


Figure 20 CPU Programming Model

● CPU Addressing Modes

The HD6303R has seven address modes which depend on both of the instruction type and the code. The address mode for every instruction is shown along with execution time given in terms of machine cycles (Table 7 to 11). When the clock frequency is 4 MHz, the machine cycle will be microseconds.

Accumulator (ACCX) Addressing

Only the accumulator (A or B) is addressed. Either accumulator A or B is specified by one-byte instructions.

Immediate Addressing

In this mode, the operand is stored in the second byte of the instruction except that the operand in LDS and LDX, etc are stored in the second and the third byte. These are two or three-byte instructions.

Direct Addressing

In this mode, the second byte of instruction indicates the address where the operand is stored. Direct addressing allows the user to directly address the lowest 256 bytes in the machine; locations zero through 255. Improved execution times are achieved by storing data in these locations. For system configuration, it is recommended that these locations should be RAM and be utilized preferably for user's data realm. These are two-byte instructions except the AIM, OIM, EIM and TIM which have three-byte.

Extended Addressing

In this mode, the second byte indicates the upper 8 bit addresses where the operand is stored, while the third byte indicates the lower 8 bits. This is an absolute address in memory. These are three-byte instructions.

Indexed Addressing

In this mode, the contents of the second byte is added to the lower 8 bits in the Index Register. For each of AIM, OIM, EIM and TIM instructions, the contents of the third byte are added to the lower 8 bits in the Index Register. In addition, the resulting "carry" is added to the upper 8 bits in the Index Register. The result is used for addressing memory. Because the modified address is held in the Temporary Address Register, there is no change to the Index Register. These are two-byte instructions but AIM, OIM, EIM, TIM have three-byte.

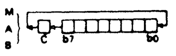
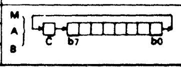
Implied Addressing

In this mode, the instruction itself gives the address; stack pointer, index register, etc. These are 1-byte instructions.

Relative Addressing

In this mode, the contents of the second byte is added to the lower 8 bits in the program counter. The resulting carry or borrow is added to the upper 8 bits. This helps the user to address the data within a range of −126 to +129 bytes of the current execution instruction. These are two-byte instructions.

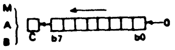
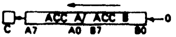
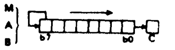
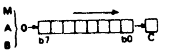
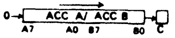
Table 7 Accumulator, Memory Manipulation Instructions

Operations	Mnemonic	Addressing Modes															Boolean/ Arithmetic Operation	Condition Code Register					
		IMMED			DIRECT			INDEX			EXTEND			IMPLIED				5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#		H	I	N	Z	V	C
Add	ADDA	8B	2	2	9B	3	2	AB	4	2	BB	4	3				A + M → A	↕	●	↕	↕	↕	↕
	ADDB	CB	2	2	DB	3	2	EB	4	2	FB	4	3				B + M → B	↕	●	↕	↕	↕	↕
Add Double	ADDD	C3	3	3	D3	4	2	E3	5	2	F3	5	3				A B + M M + 1 → A B	●	●	↕	↕	↕	↕
Add Accumulators	ABA													1B	1	1	A + B → A	↕	●	↕	↕	↕	↕
Add With Carry	ADCA	89	2	2	99	3	2	A9	4	2	B9	4	3				A + M + C → A	↕	●	↕	↕	↕	↕
	ADCB	C9	2	2	D9	3	2	E9	4	2	F9	4	3				B + M + C → B	↕	●	↕	↕	↕	↕
AND	ANDA	84	2	2	94	3	2	A4	4	2	B4	4	3				A·M → A	●	●	↕	↕	R	●
	ANDB	C4	2	2	D4	3	2	E4	4	2	F4	4	3				B·M → B	●	●	↕	↕	R	●
Bit Test	BIT A	85	2	2	95	3	2	A5	4	2	B5	4	3				A·M	●	●	↕	↕	R	●
	BIT B	C5	2	2	D5	3	2	E5	4	2	F5	4	3				B·M	●	●	↕	↕	R	●
Clear	CLR							6F	5	2	7F	5	3				00 → M	●	●	R	S	R	R
	CLRA													4F	1	1	00 → A	●	●	R	S	R	R
	CLRB													5F	1	1	00 → B	●	●	R	S	R	R
Compare	CMPA	81	2	2	91	3	2	A1	4	2	B1	4	3				A − M	●	●	↕	↕	↕	↕
	CMPB	C1	2	2	D1	3	2	E1	4	2	F1	4	3				B − M	●	●	↕	↕	↕	↕
Compare Accumulators	CBA													11	1	1	A − B	●	●	↕	↕	↕	↕
Complement, 1's	COM							63	6	2	73	6	3				$\overline{M}$ → M	●	●	↕	↕	R	S
	COMA													43	1	1	$\overline{A}$ → A	●	●	↕	↕	R	S
	COMB													53	1	1	$\overline{B}$ → B	●	●	↕	↕	R	S
Complement, 2's	NEG							60	6	2	70	6	3				00 − M → M	●	●	↕	↕	①	②
(Negate)	NEGA													40	1	1	00 − A → A	●	●	↕	↕	①	②
	NEGB													50	1	1	00 − B → B	●	●	↕	↕	①	②
Decimal Adjust, A	DAA													19	2	1	Converts binary add of BCD characters into BCD format	●	●	↕	↕	↕	③
Decrement	DEC							6A	6	2	7A	6	3				M − 1 → M	●	●	↕	↕	④	●
	DECA													4A	1	1	A − 1 → A	●	●	↕	↕	④	●
	DECB													5A	1	1	B − 1 → B	●	●	↕	↕	④	●
Exclusive OR	EORA	88	2	2	98	3	2	A8	4	2	B8	4	3				A ⊕ M → A	●	●	↕	↕	R	●
	EORB	C8	2	2	D8	3	2	E8	4	2	F8	4	3				B ⊕ M → B	●	●	↕	↕	R	●
Increment	INC							6C	6	2	7C	6	3				M + 1 → M	●	●	↕	↕	⑤	●
	INCA													4C	1	1	A + 1 → A	●	●	↕	↕	⑤	●
	INCB													5C	1	1	B + 1 → B	●	●	↕	↕	⑤	●
Load Accumulator	LDAA	86	2	2	96	3	2	A6	4	2	B6	4	3				M → A	●	●	↕	↕	R	●
	LDAB	C6	2	2	D6	3	2	E6	4	2	F6	4	3				M → B	●	●	↕	↕	R	●
Load Double Accumulator	LDD	CC	3	3	DC	4	2	EC	5	2	FC	5	3				M + 1 → B, M → A	●	●	↕	↕	R	●
Multiply Unsigned	MUL													3D	7	1	A x B → A B	●	●	●	●	●	⑪
OR, Inclusive	ORAA	8A	2	2	9A	3	2	AA	4	2	BA	4	3				A + M → A	●	●	↕	↕	R	●
	ORAB	CA	2	2	DA	3	2	EA	4	2	FA	4	3				B + M → B	●	●	↕	↕	R	●
Push Data	PSHA													36	4	1	A → Msp, SP − 1 → SP	●	●	●	●	●	●
	PSHB													37	4	1	B → Msp, SP − 1 → SP	●	●	●	●	●	●
Pull Data	PULA													32	3	1	SP + 1 → SP, Msp → A	●	●	●	●	●	●
	PULB													33	3	1	SP + 1 → SP, Msp → B	●	●	●	●	●	●
Rotate Left	ROL							69	6	2	79	6	3					●	●	↕	↕	⑥	↕
	ROLA													49	1	1		●	●	↕	↕	⑥	↕
	ROLB													59	1	1		●	●	↕	↕	⑥	↕
Rotate Right	ROR							66	6	2	76	6	3					●	●	↕	↕	⑥	↕
	RORA													46	1	1		●	●	↕	↕	⑥	↕
	RORB													56	1	1		●	●	↕	↕	⑥	↕

Note) Condition Code Register will be explained in Note of Table 10

(to be continued)

Table 7 Accumulator, Memory Manipulation Instructions

Operations	Mnemonic	Addressing Modes															Boolean/ Arithmetic Operation	Condition Code Register					
		IMMED			DIRECT			INDEX			EXTEND			IMPLIED				5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#		H	I	N	Z	V	C
Shift Left Arithmetic	ASL							68	6	2	78	6	3					●	●	↕	↕	⑥	↕
	ASLA													48	1	1		●	●	↕	↕	⑥	↕
	ASLB													58	1	1		●	●	↕	↕	⑥	↕
Double Shift Left, Arithmetic	ASLD													05	1	1		●	●	↕	↕	⑥	↕
Shift Right Arithmetic	ASR							67	6	2	77	6	3					●	●	↕	↕	⑥	↕
	ASRA													47	1	1		●	●	↕	↕	⑥	↕
	ASRB													57	1	1		●	●	↕	↕	⑥	↕
Shift Right Logical	LSR							64	6	2	74	6	3					●	●	R	↕	⑥	↕
	LSRA													44	1	1		●	●	R	↕	⑥	↕
	LSRB													54	1	1		●	●	R	↕	⑥	↕
Double Shift Right Logical	LSRD													04	1	1		●	●	R	↕	⑥	↕
Store Accumulator	STAA				97	3	2	A7	4	2	B7	4	3				A → M	●	●	↕	↕	R	●
	STAB				D7	3	2	E7	4	2	F7	4	3				B → M	●	●	↕	↕	R	●
Store Double Accumulator	STD				DD	4	2	ED	5	2	FD	5	3				A → M B → M + 1	●	●	↕	↕	R	●
Subtract	SUBA	80	2	2	90	3	2	A0	4	2	B0	4	3				A − M → A	●	●	↕	↕	↕	↕
	SUBB	C0	2	2	D0	3	2	E0	4	2	F0	4	3				B − M → B	●	●	↕	↕	↕	↕
Double Subtract	SUBD	83	3	3	93	4	2	A3	5	2	B3	5	3				A B − M M + 1 → A B	●	●	↕	↕	↕	↕
Subtract Accumulators	SBA													10	1	1	A − B → A	●	●	↕	↕	↕	↕
Subtract With Carry	SBCA	82	2	2	92	3	2	A2	4	2	B2	4	3				A − M − C → A	●	●	↕	↕	↕	↕
	SBCB	C2	2	2	D2	3	2	E2	4	2	F2	4	3				B − M − C → B	●	●	↕	↕	↕	↕
Transfer Accumulators	TAB													16	1	1	A → B	●	●	↕	↕	R	●
	TBA													17	1	1	B → A	●	●	↕	↕	R	●
Test Zero or Minus	TST							6D	4	2	7D	4	3				M − 00	●	●	↕	↕	R	R
	TSTA													4D	1	1	A − 00	●	●	↕	↕	R	R
	TSTB													5D	1	1	B − 00	●	●	↕	↕	R	R
And Immediate	AIM				71	6	3	61	7	3							M IMM → M	●	●	↕	↕	R	●
OR Immediate	OIM				72	6	3	62	7	3							M + IMM → M	●	●	↕	↕	R	●
EOR Immediate	EIM				75	6	3	65	7	3							M⊕IMM → M	●	●	↕	↕	R	●
Test Immediate	TIM				7B	4	3	6B	5	3							M IMM	●	●	↕	↕	R	●

Note) Condition Code Register will be explained in Note of Table 10

- **New Instructions**

In addition to the HD6801 Instruction Set, the HD6303R has the following new instructions:

AIM----(M) · (IMM) → (M)
Evaluates the AND of the immediate data and the memory, places the result in the memory.

OIM----(M) + (IMM) → (M)
Evaluates the OR of the immediate data and the memory, places the result in the memory.

EIM----(M) ⊕ (IMM) → (M)
Evaluates the EOR of the immediate data and the memory, places the result ın the memory.

TIM----(M) · (IMM)
Evaluates the AND of the immediate data and the memory, changes the flag of associated condition code register

Each instruction has three bytes; the first is op-code, the second is immediate data, the third is address modifier.

XGDX--(ACCD) ↔ (IX)
Exchanges the contents of accumulator and the ındex register.

SLP----The MPU is brought to the sleep mode. For sleep mode, see the "sleep mode" section.

Table 8 Index Register, Stack Manipulation Instructions

Pointer Operations	Mnemonic	Addressing Modes															Boolean/ Arithmetic Operation	Condition Code Register					
		IMMED			DIRECT			INDEX			EXTEND			IMPLIED				5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#		H	I	N	Z	V	C
Compare Index Reg	CPX	8C	3	3	9C	4	2	AC	5	2	BC	5	3				X−M M+1	●	●	↕	↕	↕	↕
Decrement Index Reg	DEX													09	1	1	X − 1 → X	●	●	●	↕	●	●
Decrement Stack Pntr	DES													34	1	1	SP − 1 → SP	●	●	●	●	●	●
Increment Index Reg	INX													08	1	1	X + 1 → X	●	●	●	↕	●	●
Increment Stack Pntr	INS													31	1	1	SP + 1 → SP	●	●	●	●	●	●
Load Index Reg	LDX	CE	3	3	DE	4	2	EE	5	2	FE	5	3				M → X_H, (M + 1) → X_L	●	●	⑦	↕	R	●
Load Stack Pntr	LDS	8E	3	3	9E	4	2	AE	5	2	BE	5	3				M → SP_H, (M+1) → SP_L	●	●	⑦	↕	R	●
Store Index Reg	STX				DF	4	2	EF	5	2	FF	5	3				X_H → M, X_L → (M + 1)	●	●	⑦	↕	R	●
Store Stack Pntr	STS				9F	4	2	AF	5	2	BF	5	3				SP_H → M, SP_L → (M + 1)	●	●	⑦	↕	R	●
Index Reg → Stack Pntr	TXS													35	1	1	X − 1 → SP	●	●	●	●	●	●
Stack Pntr → Index Reg	TSX													30	1	1	SP + 1 → X	●	●	●	●	●	●
Add	ABX													3A	1	1	B + X → X	●	●	●	●	●	●
Push Data	PSHX													3C	5	1	X_L → M_{sp}, SP − 1 → SP X_H → M_{sp}, SP − 1 → SP	●	●	●	●	●	●
Pull Data	PULX													38	4	1	SP + 1 → SP, M_{sp} → X_H SP + 1 → SP, M_{sp} → X_L	●	●	●	●	●	●
Exchange	XGDX													18	2	1	ACCD↔IX	●	●	●	●	●	●

Note) Condition Code Register will be explained ın Note of Table 10.

Table 9 Jump, Branch Instruction

Operations	Mnemonic	Addressing Modes															Branch Test	Condition Code Register					
		RELATIVE			DIRECT			INDEX			EXTEND			IMPLIED				5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#		H	I	N	Z	V	C
Branch Always	BRA	20	3	2													None	●	●	●	●	●	●
Branch Never	BRN	21	3	2													None	●	●	●	●	●	●
Branch If Carry Clear	BCC	24	3	2													C = 0	●	●	●	●	●	●
Branch If Carry Set	BCS	25	3	2													C = 1	●	●	●	●	●	●
Branch If = Zero	BEQ	27	3	2													Z = 1	●	●	●	●	●	●
Branch If ≥ Zero	BGE	2C	3	2													N ⊕ V = 0	●	●	●	●	●	●
Branch If > Zero	BGT	2E	3	2													Z + (N ⊕ V) = 0	●	●	●	●	●	●
Branch If Higher	BHI	22	3	2													C + Z = 0	●	●	●	●	●	●
Branch If ≤ Zero	BLE	2F	3	2													Z + (N ⊕ V) = 1	●	●	●	●	●	●
Branch If Lower Or Same	BLS	23	3	2													C + Z = 1	●	●	●	●	●	●
Branch If < Zero	BLT	2D	3	2													N ⊕ V = 1	●	●	●	●	●	●
Branch If Minus	BMI	2B	3	2													N = 1	●	●	●	●	●	●
Branch If Not Equal Zero	BNE	26	3	2													Z = 0	●	●	●	●	●	●
Branch If Overflow Clear	BVC	28	3	2													V = 0	●	●	●	●	●	●
Branch If Overflow Set	BVS	29	3	2													V = 1	●	●	●	●	●	●
Branch If Plus	BPL	2A	3	2													N = 0	●	●	●	●	●	●
Branch To Subroutine	BSR	8D	5	2														●	●	●	●	●	●
Jump	JMP							6E	3	2	7E	3	3					●	●	●	●	●	●
Jump To Subroutine	JSR				9D	5	2	AD	5	2	BD	6	3					●	●	●	●	●	●
No Operation	NOP													01	1	1	Advances Prog Cntr Only	●	●	●	●	●	●
Return From Interrupt	RTI													3B	10	1		—— ⑩ ——					
Return From Subroutine	RTS													39	5	1		●	●	●	●	●	●
Software Interrupt	SWI													3F	12	1		●	S	●	●	●	●
Wait for Interrupt*	WAI													3E	9	1		●	⑪	●	●	●	●
Sleep	SLP													1A	4	1		●	●	●	●	●	●

Note) *WAI puts R/$\overline{W}$ high, Address Bus goes to FFFF; Data Bus goes to the three state
Condition Code Register will be explained in Note of Table 10

Table 10 Condition Code Register Manipulation Instructions

Operations	Mnemonic	Addressing Modes			Boolean Operation	Condition Code Register					
		IMPLIED				5	4	3	2	1	0
		OP	~	#		H	I	N	Z	V	C
Clear Carry	CLC	0C	1	1	0 → C	●	●	●	●	●	R
Clear Interrupt Mask	CLI	0E	1	1	0 → I	●	R	●	●	●	●
Clear Overflow	CLV	0A	1	1	0 → V	●	●	●	●	R	●
Set Carry	SEC	0D	1	1	1 → C	●	●	●	●	●	S
Set Interrupt Mask	SEI	0F	1	1	1 → I	●	S	●	●	●	●
Set Overflow	SEV	0B	1	1	1 → V	●	●	●	●	S	●
Accumulator A → CCR	TAP	06	1	1	A → CCR	⑩					
CCR → Accumulator A	TPA	07	1	1	CCR → A	●	●	●	●	●	●

[NOTE 1] Condition Code Register Notes (Bit set if test is true and cleared otherwise)

① (Bit V) Test Result = 10000000?
② (Bit C) Test Result ≠ 00000000?
③ (Bit C) Test BCD Character of high-order byte greater than 9 ? (Not cleared if previously set)
④ (Bit V) Test Operand = 10000000 prior to execution?
⑤ (Bit V) Test Operand = 01111111 prior to execution?
⑥ (Bit V) Test Set equal to N⊕C=1 after the execution of instructions
⑦ (Bit N) Test Result less than zero? (Bit 15=1)
⑧ (All Bit) Load Condition Code Register from Stack
⑨ (Bit I) Set when interrupt occurs If previously set, a Non-Maskable Interrupt is required to exit the wait state
⑩ (All Bit) Set according to the contents of Accumulator A
⑪ (Bit C) Result of Multiplication Bit 7=1 of ACCB?

[NOTE 2] CLI instruction and interrupt
If interrupt mask-bit is set (I="1") and interrupt is requested ($\overline{IRQ_1}$ = "0" or $\overline{IRQ_2}$ = "0"), and then CLI instruction is executed, the CPU responds as follows

① The next instruction of CLI is one-machine cycle instruction
Subsequent two instructions are executed before the interrupt is responded.
That is, the next and the next of the next instruction are executed.

② The next instruction of CLI is two-machine cycle (or more) instruction.
Only the next instruction is executed and then the CPU jump to the interrupt routine

Even if TAP instruction is used, instead of CLI, the same thing occurs.

Table 11 OP-Code Map

OP CODE						ACC A	ACC B	IND	EXT / DIR*	ACCA or SP				ACCB or X				
										IMM	DIR	IND	EXT	IMM	DIR	IND	EXT	
LO	HI	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111	
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0000	0	▨	SBA	BRA	TSX	NEG				SUB								0
0001	1	NOP	CBA	BRN	INS	▨		AIM		CMP								1
0010	2	▨	▨	BHI	PULA	▨		OIM		SBC								2
0011	3	▨	▨	BLS	PULB	COM				SUBD				ADDD				3
0100	4	LSRD	▨	BCC	DES	LSR				AND								4
0101	5	ASLD	▨	BCS	TXS	▨		EIM		BIT								5
0110	6	TAP	TAB	BNE	PSHA	ROR				LDA								6
0111	7	TPA	TBA	BEQ	PSHB	ASR				▨	STA			▨	STA			7
1000	8	INX	XGDX	BVC	PULX	ASL				EOR								8
1001	9	DEX	DAA	BVS	RTS	ROL				ADC								9
1010	A	CLV	SLP	BPL	ABX	DEC				ORA								A
1011	B	SEV	ABA	BMI	RTI	▨		TIM		ADD								B
1100	C	CLC	▨	BGE	PSHX	INC				CPX				LDD				C
1101	D	SEC	▨	BLT	MUL	TST				BSR	JSR			▨	STD			D
1110	E	CLI	▨	BGT	WAI	▨	▨	JMP		LDS				LDX				E
1111	F	SEI	▨	BLE	SWI	CLR				▨	STS			▨	STX			F
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	

UNDEFINED OP CODE ▨
* Only for instructions of AIM, OIM, EIM, TIM

- **Instruction Execution Cycles**

In the HMCS6800 series, the execution cycle of each instruction is the number of cycles between the start of the current instruction fetch and just before the start of the subsequent instruction fetch.

The HD6303R uses a mechanism of the pipeline control for the instruction fetch and the subsequent instruction fetch is performed during the current instruction being executed.

Therefore, the method to count instruction cycles used in the HMCS6800 series cannot be applied to the instruction cycles such as MULT, PULL, DAA and XGDX in the HD6303R.

Table 12 provides the information about the relationship among each data on the Address Bus, Data Bus, and R/$\overline{W}$ status in cycle-by-cycle basis during the execution of each instruction.

Table 12 Cycle-by-Cycle Operation

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/$\overline{W}$	Data Bus
IMMEDIATE					
ADC ADD	2	1	Op Code Address+1	1	Operand Data
AND BIT		2	Op Code Address+2	1	Next Op Code
CMP EOR					
LDA ORA					
SBC SUB					
ADDD CPX	3	1	Op Code Address+1	1	Operand Data (MSB)
LDD LDS		2	Op Code Address+2	1	Operand Data (LSB)
LDX SUBD		3	Op Code Address+3	1	Next Op Code
DIRECT					
ADC ADD	3	1	Op Code Address+1	1	Address of Operand (LSB)
AND BIT		2	Address of Operand	1	Operand Data
CMP EOR		3	Op Code Address+2	1	Next Op Code
LDA ORA					
SBC SUB					
STA	3	1	Op Code Address+1	1	Destination Address
		2	Destination Address	0	Accumulator Data
		3	Op Code Address+2	1	Next Op Code
ADDD CPX	4	1	Op Code Address+1	1	Address of Operand (LSB)
LDD LDS		2	Address of Operand	1	Operand Data (MSB)
LDX SUBD		3	Address of Operand+1	1	Operand Data (LSB)
		4	Op Code Address+2	1	Next Op Code
STD STS	4	1	Op Code Address+1	1	Destination Address (LSB)
STX		2	Destination Address	0	Register Data (MSB)
		3	Destination Address+1	0	Register Data (LSB)
		4	Op Code Address+2	1	Next Op Code
JSR	5	1	Op Code Address+1	1	Jump Address (LSB)
		2	FFFF	1	Restart Address (LSB)
		3	Stack Pointer	0	Return Address (LSB)
		4	Stack Pointer−1	0	Return Address (MSB)
		5	Jump Address	1	First Subroutine Op Code
TIM	4	1	Op Code Address+1	1	Immediate Data
		2	Op Code Address+2	1	Address of Operand (LSB)
		3	Address of Operand	1	Operand Data
		4	Op Code Address+3	1	Next Op Code
AIM EIM	6	1	Op Code Address+1	1	Immediate Data
OIM		2	Op Code Address+2	1	Address of Operand (LSB)
		3	Address of Operand	1	Operand Data
		4	FFFF	1	Restart Address (LSB)
		5	Address of Operand	0	New Operand Data
		6	Op Code Address+3	1	Next Op Code

– Continued –

Table 12 Cycle-by-Cycle Operation (Continued)

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/$\overline{W}$	Data Bus
INDEXED					
JMP	3	1	Op Code Address + 1	1	Offset
		2	FFFF	1	Restart Address (LSB)
		3	Jump Address	1	First Op Code of Jump Routine
ADC ADD	4	1	Op Code Address + 1	1	Offset
AND BIT		2	FFFF	1	Restart Address (LSB)
CMP EOR		3	IX + Offset	1	Operand Data
LDA ORA		4	Op Code Address + 2	1	Next Op Code
SBC SUB					
TST					
STA	4	1	Op Code Address + 1	1	Offset
		2	FFFF	1	Restart Address (LSB)
		3	IX + Offset	0	Accumulator Data
		4	Op Code Address + 2	1	Next Op Code
ADDD	5	1	Op Code Address + 1	1	Offset
CPX LDD		2	FFFF	1	Restart Address (LSB)
LDS LDX		3	IX + Offset	1	Operand Data (MSB)
SUBD		4	IX + Offset + 1	1	Operand Data (LSB)
		5	Op Code Address + 2	1	Next Op Code
STD STS	5	1	Op Code Address + 1	1	Offset
STX		2	FFFF	1	Restart Address (LSB)
		3	IX + Offset	0	Register Data (MSB)
		4	IX + Offset + 1	0	Register Data (LSB)
		5	Op Code Address + 2	1	Next Op Code
JSR	5	1	Op Code Address + 1	1	Offset
		2	FFFF	1	Restart Address (LSB)
		3	Stack Pointer	0	Return Address (LSB)
		4	Stack Pointer − 1	0	Return Address (MSB)
		5	IX + Offset	1	First Subroutine Op Code
ASL ASR	6	1	Op Code Address + 1	1	Offset
COM DEC		2	FFFF	1	Restart Address (LSB)
INC LSR		3	IX + Offset	1	Operand Data
NEG ROL		4	FFFF	1	Restart Address (LSB)
ROR		5	IX + Offset	0	New Operand Data
		6	Op Code Address + 2	1	Next Op Code
TIM	5	1	Op Code Address + 1	1	Immediate Data
		2	Op Code Address + 2	1	Offset
		3	FFFF	1	Restart Address (LSB)
		4	IX + Offset	1	Operand Data
		5	Op Code Address + 3	1	Next Op Code
CLR	5	1	Op Code Address + 1	1	Offset
		2	FFFF	1	Restart Address (LSB)
		3	IX + Offset	1	Operand Data
		4	IX + Offset	0	00
		5	Op Code Address + 2	1	Next Op Code
AIM EIM	7	1	Op Code Address + 1	1	Immediate Data
OIM		2	Op Code Address + 2	1	Offset
		3	FFFF	1	Restart Address (LSB)
		4	IX + Offset	1	Operand Data
		5	FFFF	1	Restart Address (LSB)
		6	IX + Offset	0	New Operand Data
		7	Op Code Address + 3	1	Next Op Code

– Continued –

Table 12 Cycle-by-Cycle Operation (Continued)

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/W̄	Data Bus
EXTEND					
JMP	3	1	Op Code Address+1	1	Jump Address (MSB)
		2	Op Code Address+2	1	Jump Address (LSB)
		3	Jump Address	1	Next Op Code
ADC ADD TST	4	1	Op Code Address+1	1	Address of Operand (MSB)
AND BIT		2	Op Code Address+2	1	Address of Operand (LSB)
CMP EOR		3	Address of Operand	1	Operand Data
LDA ORA		4	Op Code Address+3	1	Next Op Code
SBC SUB					
STA	4	1	Op Code Address+1	1	Destination Address (MSB)
		2	Op Code Address+2	1	Destination Address (LSB)
		3	Destination Address	0	Accumulator Data
		4	Op Code Address+3	1	Next Op Code
ADDD	5	1	Op Code Address+1	1	Address of Operand (MSB)
CPX LDD		2	Op Code Address+2	1	Address of Operand (LSB)
LDS LDX		3	Address of Operand	1	Operand Data (MSB)
SUBD		4	Address of Operand+1	1	Operand Data (LSB)
		5	Op Code Address+3	1	Next Op Code
STD STS	5	1	Op Code Address+1	1	Destination Address (MSB)
STX		2	Op Code Address+2	1	Destination Address (LSB)
		3	Destination Address	0	Register Data (MSB)
		4	Destination Address+1	0	Register Data (LSB)
		5	Op Code Address+3	1	Next Op Code
JSR	6	1	Op Code Address+1	1	Jump Address (MSB)
		2	Op Code Address+2	1	Jump Address (LSB)
		3	FFFF	1	Restart Address (LSB)
		4	Stack Pointer	0	Return Address (LSB)
		5	Stack Pointer−1	0	Return Address (MSB)
		6	Jump Address	1	First Subroutine Op Code
ASL ASR	6	1	Op Code Address+1	1	Address of Operand (MSB)
COM DEC		2	Op Code Address+2	1	Address of Operand (LSB)
INC LSR		3	Address of Operand	1	Operand Data
NEG ROL		4	FFFF	1	Restart Address (LSB)
ROR		5	Address of Operand	0	New Operand Data
		6	Op Code Address+3	1	Next Op Code
CLR	5	1	Op Code Address+1	1	Address of Operand (MSB)
		2	Op Code Address+2	1	Address of Operand (LSB)
		3	Address of Operand	1	Operand Data
		4	Address of Operand	0	00
		5	Op Code Address+3	1	Next Op Code

— Continued —

Table 12 Cycle-by-Cycle Operation (Continued)

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/$\overline{W}$	Data Bus
IMPLIED					
ABA ABX ASL ASLD ASR CBA CLC CLI CLR CLV COM DEC DES DEX INC INS INX LSR LSRD ROL ROR NOP SBA SEC SEI SEV TAB TAP TBA TPA TST TSX TXS	1	1	Op Code Address+1	1	Next Op Code
DAA XGDX	2	1	Op Code Address+1	1	Next Op Code
		2	FFFF	1	Restart Address (LSB)
PULA PULB	3	1	Op Code Address+1	1	Next Op Code
		2	FFFF	1	Restart Address (LSB)
		3	Stack Pointer+1	1	Data from Stack
PSHA PSHB	4	1	Op Code Address+1	1	Next Op Code
		2	FFFF	1	Restart Address (LSB)
		3	Stack Pointer	0	Accumulator Data
		4	Op Code Address+1	1	Next Op Code
PULX	4	1	Op Code Address+1	1	Next Op Code
		2	FFFF	1	Restart Address (LSB)
		3	Stack Pointer+1	1	Data from Stack (MSB)
		4	Stack Pointer+2	1	Data from Stack (LSB)
PSHX	5	1	Op Code Address+1	1	Next Op Code
		2	FFFF	1	Restart Address (LSB)
		3	Stack Pointer	0	Index Register (LSB)
		4	Stack Pointer−1	0	Index Register (MSB)
		5	Op Code Address+1	1	Next Op Code
RTS	5	1	Op Code Address+1	1	Next Op Code
		2	FFFF	1	Restart Address (LSB)
		3	Stack Pointer+1	1	Return Address (MSB)
		4	Stack Pointer+2	1	Return Address (LSB)
		5	Return Address	1	First Op Code of Return Routine
MUL	7	1	Op Code Address+1	1	Next Op Code
		2	FFFF	1	Restart Address (LSB)
		3	FFFF	1	Restart Address (LSB)
		4	FFFF	1	Restart Address (LSB)
		5	FFFF	1	Restart Address (LSB)
		6	FFFF	1	Restart Address (LSB)
		7	FFFF	1	Restart Address (LSB)

– Continued –

Table 12 Cycle-by-Cycle Operation (Continued)

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/$\overline{W}$	Data Bus
IMPLIED					
WAI	9	1	Op Code Address+1	1	Next Op Code
		2	FFFF	1	Restart Address (LSB)
		3	Stack Pointer	0	Return Address (LSB)
		4	Stack Pointer−1	0	Return Address (MSB)
		5	Stack Pointer−2	0	Index Register (LSB)
		6	Stack Pointer−3	0	Index Register (MSB)
		7	Stack Pointer−4	0	Accumulator A
		8	Stack Pointer−5	0	Accumulator B
		9	Stack Pointer−6	0	Conditional Code Register
RTI	10	1	Op Code Address+1	1	Next Op Code
		2	FFFF	1	Restart Address (LSB)
		3	Stack Pointer +1	1	Conditional Code Register
		4	Stack Pointer+2	1	Accumulator B
		5	Stack Pointer+3	1	Accumulator A
		6	Stack Pointer+4	1	Index Register (MSB)
		7	Stack Pointer+5	1	Index Register (LSB)
		8	Stack Pointer+6	1	Return Address (MSB)
		9	Stack Pointer+7	1	Return Address (LSB)
		10	Return Address	1	First Op Code of Return Routine
SWI	12	1	Op Code Address+1	1	Next Op Code
		2	FFFF	1	Restart Address (LSB)
		3	Stack Pointer	0	Return Address (LSB)
		4	Stack Pointer − 1	0	Return Address (MSB)
		5	Stack Pointer − 2	0	Index Register (LSB)
		6	Stack Pointer − 3	0	Index Register (MSB)
		7	Stack Pointer − 4	0	Accumulator A
		8	Stack Pointer − 5	0	Accumulator B
		9	Stack Pointer − 6	0	Conditional Code Register
		10	Vector Address FFFA	1	Address of SWI Routine (MSB)
		11	Vector Address FFFB	1	Address of SWI Routine (LSB)
		12	Address of SWI Routine	1	First Op Code of SWI Routine
SLP	4	1	Op Code Address+1	1	Next Op Code
		2	FFFF	1	Restart Address (LSB)
		↑ Sleep ↓	FFFF ↓		High Impedance-Non MPX Mode Address Bus -MPX Mode ↓
		3	FFFF		Restart Address (LSB)
		4	Op Code Address+1		Next Op Code

— Continued —

Table 12 Cycle-by-Cycle Operation (Continued)

Address Mode & Instructions		Cycles	Cycle #	Address Bus	R/$\overline{W}$	Data Bus
RELATIVE						
BCC	BCS		1	Op Code Address+1	1	Branch Offset
BEQ	BGE	3	2	FFFF	1	Restart Address (LSB)
BGT	BHI		3	Branch Address ·Test="1" Op Code Address+1 ·Test="0"	1	First Op Code of Branch Routine Next Op Code
BLE	BLS					
BLT	BMT					
BNE	BPL					
BRA	BRN					
BVC	BVS					
BSR			1	Op Code Address+1	1	Offset
			2	FFFF	1	Restart Address (LSB)
		5	3	Stack Pointer	0	Return Address (LSB)
			4	Stack Pointer−1	0	Return Address (MSB)
			5	Branch Address	1	First Op Code of Subroutine

■ LOW POWER CONSUMPTION MODE

The HD6303R has two low power consumption modes; sleep and standby mode.

● Sleep Mode

On execution of SLP instruction, the MPU is brought to the sleep mode. In the sleep mode, the CPU stops its operation, but the contents of the registers in the CPU are retained. In this mode, the peripherals of CPU will remain active. So the operations such as transmit and receive of the SCI data and counter may keep in operation. In this mode, the power consumption is reduced to about 1/6 the value of a normal operation.

The escape from this mode can be done by interrupt, $\overline{\text{RES}}$, $\overline{\text{STBY}}$. The $\overline{\text{RES}}$ resets the MPU and the $\overline{\text{STBY}}$ brings it into the standby mode (This will be mentioned later). When interrupt is requested to the CPU and accepted, the sleep mode is released, then the CPU is brought in the operation mode and jumps to the interrupt routine. When the CPU has masked the interrupt, after recovering from the sleep mode, the next instruction of SLP starts to execute. However, in such a case that the timer interrupt is inhibited on the timer side, the sleep mode cannot be released due to the absence of the interrupt request to the CPU.

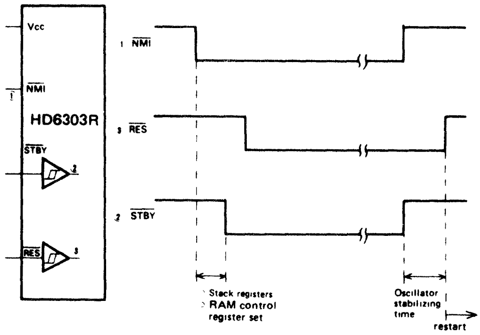


Figure 21 Standby Mode Timing

This sleep mode is available to reduce an average power consumption in the applications of the HD6303R which may not be always running.

● Standby Mode

Bringing $\overline{\text{STBY}}$ "Low", the CPU becomes reset and all clocks of the HD6303R become inactive. It goes into the standby mode. This mode remarkably reduces the power consumptions of the HD6303R.

In the standby mode, if the HD6303R is continuously supplied with power, the contents of RAM is retained. The standby mode should escape by the reset start. The following is the typical application of this mode.

First, $\overline{\text{NMI}}$ routine stacks the CPU's internal information and the contents of SP in RAM, disables RAME bit of RAM control register, sets the standby bit, and then goes into the standby mode. If the standby bit keeps set on reset start, it means that the power has been kept during stand-by mode and the contents of RAM is normally guaranteed. The system recovery may be possible by returning SP and bringing into the condition before the standby mode has started. The timing relation for each line in this application is shown in Figure 21.

■ ERROR PROCESSING

When the HD6303R fetches an undefined instruction or fetches an instruction from unusable memory area, ıt generates the highest priority internal interrupt, that may protect from system upset due to noise or a program error.

● Op-Code Error

Fetching an undefined op-code, the HD6303R will stack the CPU register as in the case of a normal interrupt and vector to **the TRAP ($FFEE, $FFEF), that has a second highest pnority ($\overline{\text{RES}}$ is the highest).**

● Address Error

When an instruction is fetched from other than a resident RAM, or an external memory area, the CPU starts the same interrupt as op-code error. In the case which the ınstruction ıs fetched from external memory area and that area is not usable, the address error can not be detected.

The address which cause address error are shown in Table 13.

This feature is applicable only to the instruction fetch, not to normal read/write of data accessing.

Transitions among the active mode, sleep mode, standby mode and reset are shown in Figure 22.

Figures 23, 24 show a system configuration.

The system flow chart of HD6303R is shown in Fıgure 25.

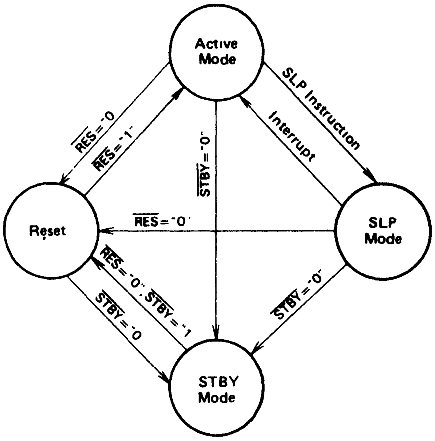

Active Mode
SLP Instruction
Interrupt
$\overline{\text{RES}}$="0
$\overline{\text{RES}}$="1"
$\overline{\text{STBY}}$="0"
Reset
$\overline{\text{RES}}$="0"
SLP Mode
$\overline{\text{RES}}$="0", $\overline{\text{STBY}}$="1
$\overline{\text{STBY}}$="0
$\overline{\text{STBY}}$="0"
STBY Mode

Figure 22 Transıtıons among Active Mode, Standby Mode, Sleep Mode, and Reset

Table 13 Address Error

Address Error
$0000 ~ $001F

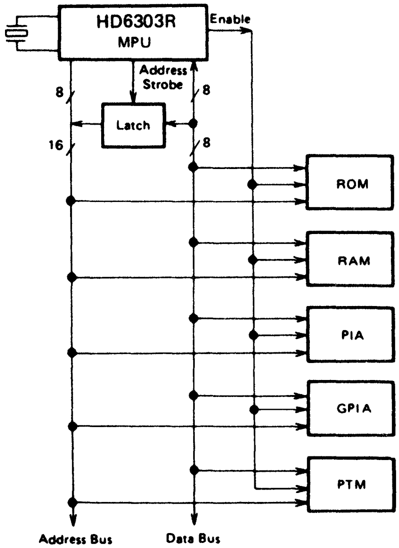


Figure 23 HD6303R MPU Multiplexed Mode

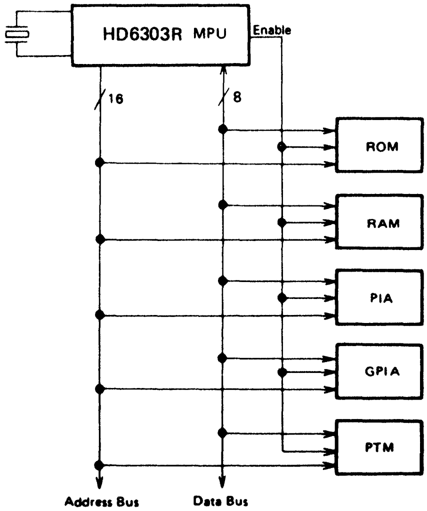


Figure 24 HD6303R MPU Non-Multiplexed Mode

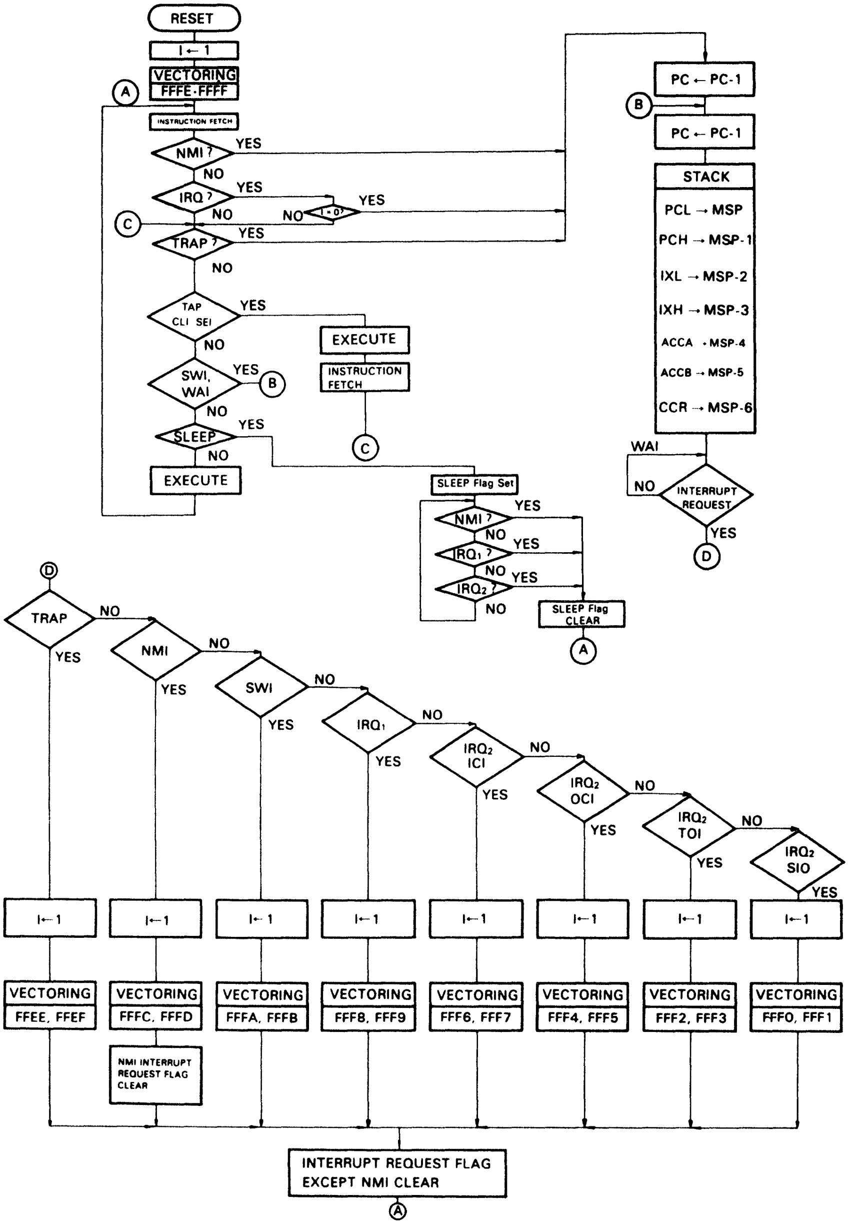
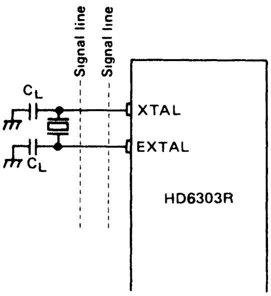


Figure 25 HD6303R System Flow Chart

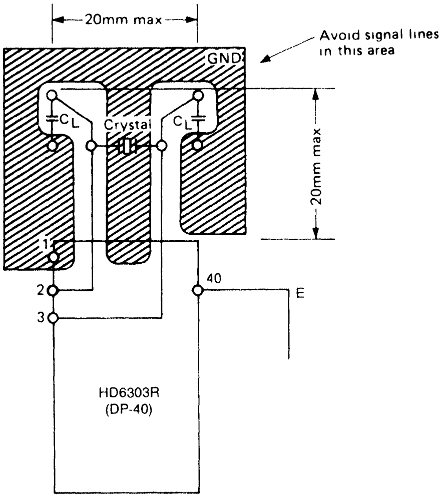
■ PRECAUTION TO THE BOARD DESIGN OF OSCILLATION CIRCUIT

As shown in Fig. 26, there is a case that the cross talk disturbs the normal oscillation if signal lines are put near the oscillation circuit. When designing a board, pay attention to this. Crystal and C_L must be put as near the HD6303R as possible.



Do not use this kind of print board design

Figure 26 Precaution to the boad design of oscillation circuit



(Top View)

Fig. 27 Example of Oscillation Circuits in Board Design

■ PIN CONDITIONS AT SLEEP AND STANDBY STATE

● Sleep State

The conditions of power supply pins, clock pins, input pins and E clock pin are the same as those of operation. Refer to Table 14 for the other pin conditions.

● Standby State

Only power supply pins and $\overline{STBY}$ are active. As for the clock pin EXTAL, its input is fixed internally so the MPU is not influenced by the pin conditions. XTAL is in "1" output. All the other pins are in high impedance.

Table 14 Pin Condition in Sleep State

Pin	Mode	Non Multiplexed Mode	Multiplexed Mode
$P_{20} \sim P_{24}$	Function	I/O Port	I/O Port
	Condition	Keep the condition just before sleep	←
$A_0/P_{10} \sim A_7/P_{17}$	Function	Address Bus ($A_0 \sim A_7$)	I/O Port
	Condition	Output "1"	Keep the condition just before sleep
$A_8 \sim A_{15}$	Function	Address Bus ($A_8 \sim A_{15}$)	Address Bus ($A_8 \sim A_{15}$)
	Condition	Output "1"	←
$D_0/A_0 \sim D_7/A_7$	Function	Data Bus ($D_0 \sim D_7$)	$\bar{E}$: Address Bus ($A_0 \sim A_7$), E: Data Bus
	Condition	High Impedance	$\bar{E}$: Output "1", E: High Impedance
R/$\overline{W}$	Function	R/$\overline{W}$ Signal	R/$\overline{W}$ Signal
	Condition	Output "1"	←
AS		–	Output AS

Table 15 Pin Condition during RESET

Pin \ Mode	Non-Multiplexed Mode	Multiplexed Mode
P_{20} ~ P_{24}	High Impedance	←
A_0/P_{10} ~ A_7/P_{17}	High Impedance	←
A_8 ~ A_{15}	High Impedance	←
D_0/A_0 ~ D_7/A_7	High Impedance	$\bar{E}$ "1" Output E "1" Output (Note) (High Impedance)
R/$\overline{W}$	"1" Output	←
AS	$\bar{E}$. "1" Output E · High Impedance	←

(Note) In the multiplexed mode, the data bus is set to "1" output state during E = "1" and it causes the conflict with the output of external memory Following 1 and 2 should be done to avoid the conflict,

(1) Construct the system that disables the external memory during reset

(2) Add 4 7 kΩ pull-down resistance to the AS pin to make AS pin "0" level during E = "1" This operation makes the data bus high impedance state

■ DIFFERENCE BETWEEN HD6303 AND HD6303R

The HD6303R is an upgraded version of the HD6303. The difference between HD6303 and HD6303R is shown in Table 16.

Table 16 Difference between HD6303 and HD6303R

Item	HD6303	HD6303R
Operating Mode	Mode 2· Not defined	Mode 2: Multiplexed Mode (Equivalent to Mode 4)
Electrical Character-istics	The electrical character-istics of 2MHz version (B version) are not spec-ified.	Some characteristics are improved. The 2MHz version is guaranteed.
Timer	Has problem in output compare function. (Can be avoided by soft-ware.)	The problem is solved.

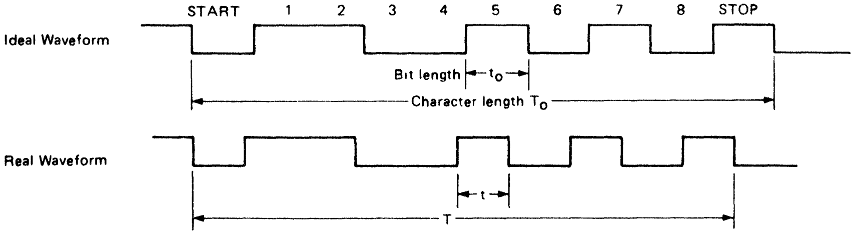
■ RECEIVE MARGIN OF THE SCI

Receive margin of the SCI contained in the HD6303R is shown in Table 17

Note SCI = Serial Communication Interface

Table 17

	Bit distortion tolerance (t−to) /to	Character distortion tolerance (T−To) /To
HD6303R	±37 5%	+3.75% -2.5%



■ APPLICATION NOTE FOR HIGH SPEED SYSTEM DESIGN USING THE HD6303R

This note describes the solutions of the potential problem caused by noise generation in the system using the HD6303R.

The CMOS ICs and LSIs featured by low power consumption and high noise immunity are generally considered to be enough with simply designed power source and the GND line.

But this does not apply to the applications configured of high speed system or of high speed parts Such high speed system may have a chance to work incorrectly because of the noise by the transient current generated during switching One of example is a system in which the HD6303R directly accesses high speed memory such as the HM6264. The noise generation owing to the over current (Sometimes it may be several hundreds mA for peak level) during switching may cause data write error

This noise problem may be observed only at the Expanded Mode (Mode 1, 2, 4, 5 and 6) of the HD6303R

Assuming the HD6303R is used as CPU in a system

I. Noise Occurrence

If the HD6303R is connected to high speed RAM, a write error may occur As shown in Fig 28, the noise is generated in address bus during write cycle and data is written into an unexpected address from the HD6303R This phenomenon causes random failures in systems whose data bus load capacitance exceeds the specification value (90 pF max.) and/or the impedance of the GND line is high

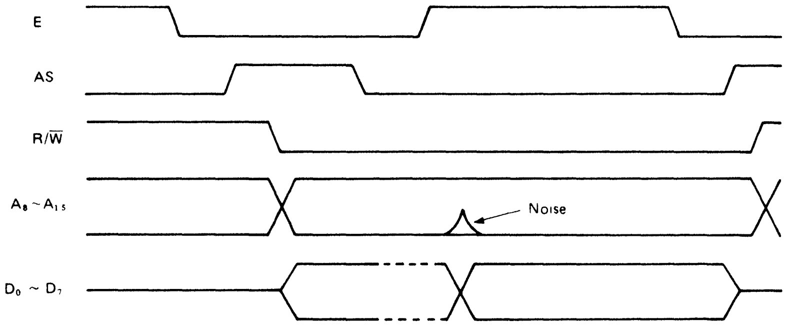


Fig. 28 Noise Occurrence in address bus during write cycle

If the data bus $D_0 \sim D_7$ changes from "FF" to "00", extremely large transient current flows through the GND line Then the noise is generated on the LSI's V_{SS} pins proportioning to the transient current and to the impedance [Zg] of the GND line.

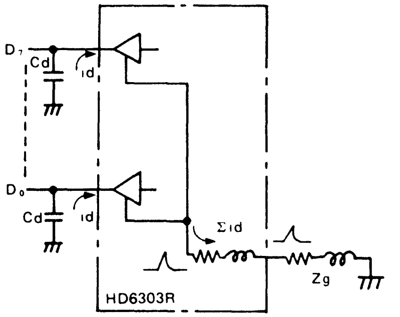
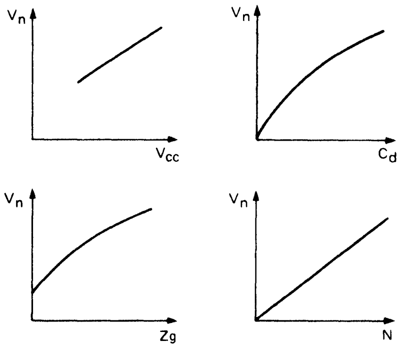


Fig 29 Noise Source

This noise level, V_n, appears on all output pins on the LSI including the address bus.

Fig. 30 shows the dependency of the noise voltage on the each parameter.



Vn Noise Voltage Zg GND Impedance
Cd Data bus load capacitance
N Number of data bus lines switching from H to L

Fig 30 Dependency of the noise voltage on each parameter

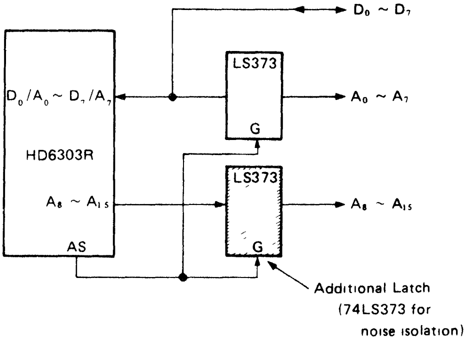
II. Noise Protection

To avoid the noise on the address bus during the system operation mentioned before, there are two solutions as follows

The one method is to isolate the HD6303R from peripheral devices so that peripherals are not affected by the noise The other is to reduce noise level to the extent of not affecting peripherals using analog method.

1. Noise Isolation

Addresses should be latched at the negative edge of the AS signal or at the positive edge of the E signal. The 74LS373 is often used in this case.



2. Noise Reduction

As the noise level depends on each parameter such Cd, V_{CC}, Zg, the noise level can be reduced to the allowable level by controlling those analog parameters.

(a) Transient Current Reduction

(1) Reduce the data bus load capacitance. If large load capacitance is expected, a bus buffer should be inserted.

(2) Lower the power supply voltage V_{CC} within specification.

(3) Increase a time constant at transient state by inserting a resistor (100 ~ 200Ω) to Data Buses in series to keep noise level down.

Table 18 shows the relationship between a series resistor and noise level or a resistor and DC/AC characteristics.

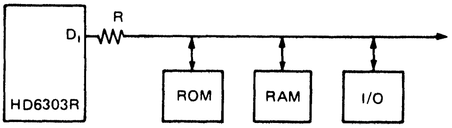


Table 18.

Item \ Resistor			No	100Ω	200Ω
Noise Voltage Level			See Fig. 31		
DC Characteristics		I_{OL}	1.6 mA	1.6 mA	1.0 mA
AC Charac-teristics	f = 1 MHz		No change		
	f = 1 5 MHz	t_{ADL}	190 ns	190 ns	210 ns
		t_{ACCM}	395 ns	395 ns	375 ns
	f = 2 MHz	t_{ADL}	160 ns	180 ns	200 ns
		t_{ASL}	20 ns	20 ns	0 ns
		t_{ACCM}	270 ns	250 ns	230 ns

Fig 31 shows an example of the dependency of the noise voltage on the load capacitance of the data bus.*

*Note The value of series resistor should be carefully selected because it heavily depends on each parameter of actual application system

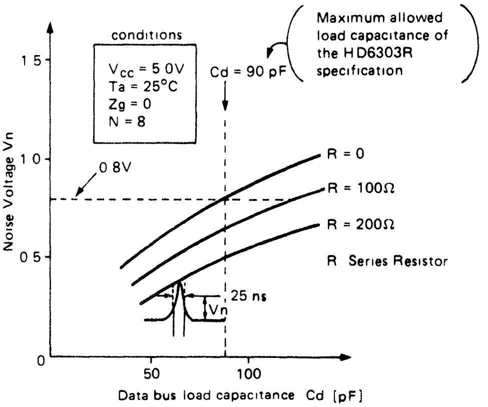


Fig. 31

Fig. 32 shows the typical wave form of the noise.

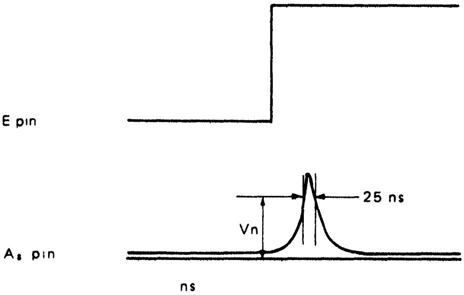


Fig. 32

(b) Reduction of GND line impedance
(1) Widen the GND line width on the PC board.
(2) Place the HD6303R close by power source.
(3) Insert a bypass capacitor between the V_{CC} line and the GND of the HD6303R. A tantalum capacitor (about 0.1μF) is effective on the reduction.

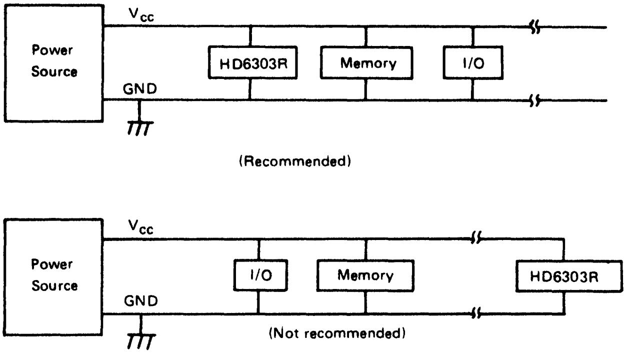


Fig. 33 Layout of the HD6303R on the PC board

■ WARNING CONCERNING POWER START-UP

$\overline{\text{RES}}$ must be held low for at least 20 ms when the power starts up. In this case, the internal reset function is not effective until the oscillation begins at power-on. The $\overline{\text{RES}}$ signal is input to the LSI in synchronism with the internal clock ϕ (shown in Figure 34.)

Therefore, after power starts up, the LSI conditions such as its I/O ports and operating mode, are unstable. Fix the level of I/O ports by means of an external circuit to determine the level for system operation during the oscillator stabilization time.

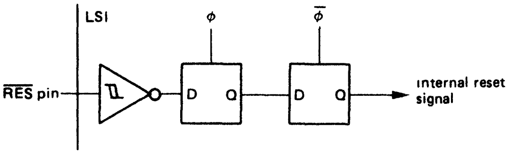


Figure 34 $\overline{\text{RES}}$ circuit

■ WRITE-ONLY REGISTER

When the CPU reads a write-only register, the read data is always $FF, regardless of the value in the write-only register. Therefore, be careful of the results of instructions which read a write-only register and perform an arithmetic or logical operation on its contents, such as AIM, ADD, or ROL, is executed, because the arithmetic or logical operation is always done with the data $FF. In particular, don't use the AIM, OIM or EIM instruction to manipulate the DDR bit of PORT.

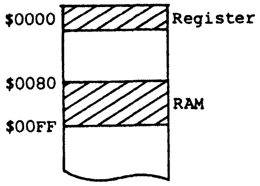
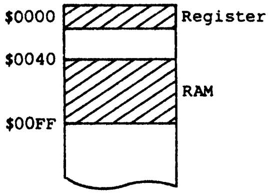
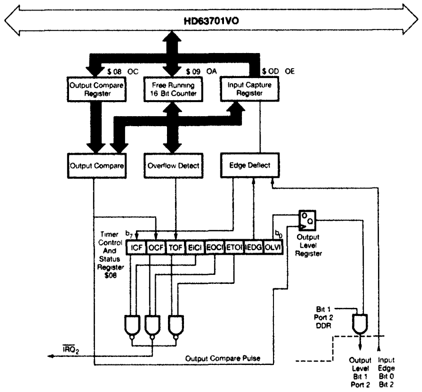
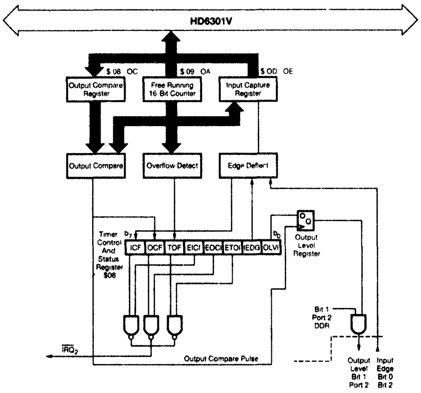
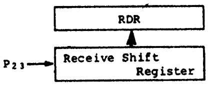
■ NOTICE ON HD6303R

The HD6303R is the same die as the HD6301V1. The on-chip Mask ROM is disabled by mask option; therefore not all modes of operation are available on the HD6303R. Please note that wherever HD6301V1 is referenced, the information also applies to the HD6303R.

■ NOTICE ON HD6303R1

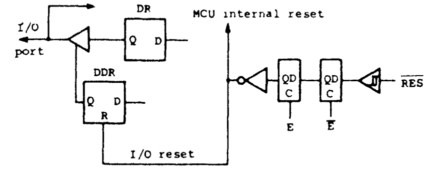
The HD6303R has been upgraded to HD6303R1. Refer to the following figures for differences between the devices. All other characteristics remain the same.

■ DIFFERENCES BETWEEN HD6301V1, HD6303R, HD6303R1, HD63P01M1, AND HD63701V0

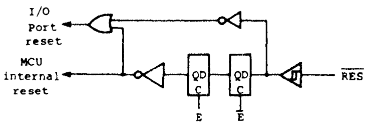
	Item	HD6301V		HD63701V0
Function	RAM	RAM Size: 128-byte Address: $0080-$00FF  		RAM Size: 192-byte Address: $0040-$00FF 
	Operation Mode	Mode 4: Expanded Multiplexed Mode = Mode 2		HD63701V0 does not have Mode 4
	Timer	After providing supply voltage, output level is undefined (0 or 1) unless the contents of the Output Compare Register matches with those of the Free Running Counter. The Output Level Register is not initialized by reset.  HD63701VO $ 08 OC $ 09 OA $ OD OE Output Compare Register Free Running 16 Bit Counter Input Capture Register Output Compare Overflow Detect Edge Deflect Timer Control And Status Register $08 b7 ICF OCF TOF EICI EOCI ETOI IEDG OLVI b0 D Q Output Level Register Bit 1 Port 2 DDR IRQ2 Output Compare Pulse Output Level Bit 1 Port 2 Input Edge Bit 0 Bit 2 Figure 20 Programmable Timer Block Diagram		The Output Level Register is initialized to 0 by reset.  HD6301V $ 08 OC $ 09 OA $ OD OE Output Compare Register Free Running 16 Bit Counter Input Capture Register Output Compare Overflow Detect Edge Deflect Timer Control And Status Register $08 b7 ICF OCF TOF EICI EOCI ETOI IEDG OLVI b0 D Q Output Level Register Bit 1 Port 2 DDR IRQ2 Output Compare Pulse Output Level Bit 1 Port 2 Input Edge Bit 0 Bit 2 Figure 20 Programmable Timer Block Diagram
	SCI	HD6301V1, HD6303R, HD63P01M1	HD6303R1	Receive data is transferred from Receive Shift Register to RDR even if framing error occurs.
		When framing error occurs, receive data is not transferred from the Receive Shift Register to Receive Data Register (RDR).  	Receive data is transferred from Receive Shift Register to RDR even if framing error occurs.	

HITACHI

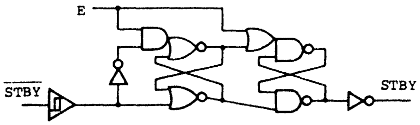
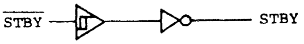
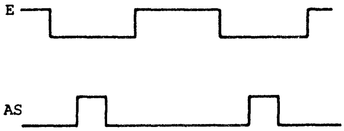
■ DIFFERENCES BETWEEN HD6301V1, HD6303R, HD6303R1, HD63P01M1, AND HD63701V0 (Continued)

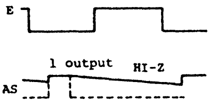
<table>
<tr><th colspan="2">Item</th><th colspan="2">HD6301V</th><th>HD63701V0</th></tr>
<tr><td rowspan="6">Function</td><td>Port Reset</td><td colspan="2">The DDR of port is reset synchronously with E clock I/O state is undefined from providing power supply till oscillation start (max 20ms)


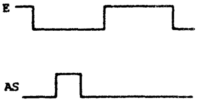
DR
I/O
port
Q D
DDR
Q D
R
MCU internal reset
QD
C
QD
C
RES
E
E
I/O reset

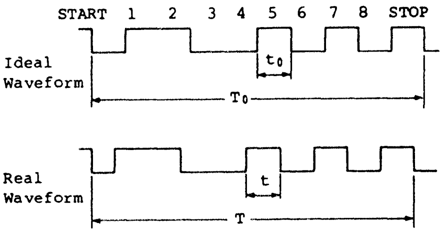
</td><td>The DDR of port is reset asynchronously with E clock. CPU enters into high impedance state (input state) by bringing RES Low
Reset release and MCU internal reset is performed synchronously with E clock


I/O
Port
reset
MCU
internal
reset
QD
C
QD
C
RES
E
E

</td></tr>
<tr><td>Standby Mode</td><td colspan="2">STBY signal is latched synchronously with E clock

 </td><td>STBY signal is latched asynchronously with E clock CPU enters into standby state by bringing STBY low

 </td></tr>
<tr><td rowspan="2">AS
(Address
Strobe)</td><td>HD63P01M1</td><td>HD6301V1, HD6303R,
HD6303R1</td><td rowspan="2">

During reset, AS functions normally</td></tr>
<tr><td>

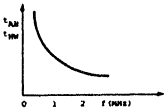
In Expanded Multiplexed Mode (mode 0, 2, 4 or 6), AS becomes high impedance state for a half E clock cycle during reset.
Therefore, I/O Port 3 functions as data bus during reset</td><td>

During reset, AS functions normally.</td></tr>
<tr><td rowspan="2">SCI Receive
Margin</td><td>HD6301V1, HD6303R,
HD6303R1</td><td>HD63P01M1</td><td rowspan="2">The SCI receive margin is shown below


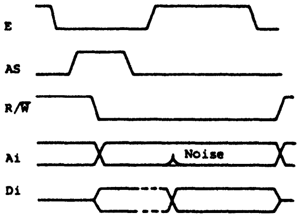
START 1 2 3 4 5 6 7 8 STOP
Ideal
Waveform
t0
T0
Real
Waveform
t
T

<table><tr><td>Bit distortion tolerance (t-t0)/t0</td><td>±37.5%</td></tr><tr><td>Character distortion tolerance (T-T0)/T0</td><td>±3.75%</td></tr></table></td></tr>
<tr><td>The SCI receive margin is shown below.
<table><tr><td>Bit distortion tolerance (t-t0)/t0</td><td>±37.5%</td></tr><tr><td>Character distortion tolerance (T-T0)/T0</td><td>+3.5%
-2.5%</td></tr></table></td><td>The SCI receive margin is shown below
<table><tr><td>Bit distortion tolerance (t-t0)/t0</td><td>±25%</td></tr><tr><td>Character distortion tolerance (T-T0)/T0</td><td>±3.75%</td></tr></table></td></tr>
</table>

■ DIFFERENCES BETWEEN HD6301V1, HD6303R, HD6303R1, HD63P01M1, AND HD63701V0 (Continued)

	Item	HD6301V		HD63701V0
		HD6301V1, HD6303R, HD6303R1	HD63P01M1	
Function	Supply Voltage	V_{CC} = 5V ± 10% (f = 0.1 ~ 2 MHz) V_{CC} = 3 ~ 6V (f = 0.1 ~ 0.5 MHz)	V_{CC} = 5V ± 10% (f = 0.1 ~ 1 MHz)	V_{CC} = 5V ± 10% (f = 0.1 ~ 2 MHz)
	Address/Data Hold Time (t_{AH}, t_{HW})	t_{AH} = 20 ns min. t_{HW} = 20 ns min. t_{AH} and t_{HW} are constant independently of operating frequency.		t_{AH}, t_{HW} = 60 ns (f = 1 MHz) = 40 ns (f = 1.5 MHz) = 30 ns (f = 2 MHz) t_{AH} and t_{HW} are proportion to 1/f. (f = operating frequency) 
Specification	Address Delay Time	(1) t_{AD1} and t_{AD2} are constant ındependently of operating frequency. In HD63B01V (B version of HD6301V), t_{AD1} and t_{AD2} are 160 ns max. at 0.1 MHz through 2 MHz operation. (2) t_{ADL} ıs related to operating frequency. (t_{ADL} is in proportion to 1/f. f = operating frequency)		t_{AD1}, t_{AD2} and t_{ADL} are related to operating frequency (They are in proportion to 1/f. f = operating frequency). Therefore, if HD637B01V operates at lower operating frequency, t_{AD1}, t_{AD2} and t_{ADL} will become 160 ns or more. t_{AD1}, t_{AD2} and t_{ADL} are calculated as follows. t_{AD} (f MHz) ≒ 250 ns (1 MHz) × 1/f (MHz)
	I_{in} and C_{in} of $\overline{RES}$	I_{in} = 1.0 μA max., C_{in} = 12.5 pF max.		I_{in} = 10 μA max. C_{in} = 50 pF max. Since $\overline{RES}$ is multiplexed with V_{PP}, C_{in} and I_{in} are larger than those of HD6301V.
	Load Capacitance of E	2 - LSTTL + 40pF I_{OL} = 0.8 mA, I_{OH} = -200 μA		1 - TTL + 90pF I_{OL} = 1.6 mA, I_{OH} = -200 μA
	Load Capacitance of Port 1	1 - TTL + 30pF		1 - TTL + 90pF
	Spec. of Crystal Oscillator	Spec. R_s = 60Ω max.		Spec. Clock frequency (MHz): 2.5 / 4.0 / 6.0 / 8.0 Rs max. (Ω): 500 / 120 / 80 / 60
	Storage Temperature	T_{stg} = -55 - +150°C		T_{stg} = -55 - +125°C

■ DIFFERENCES BETWEEN HD6301V1, HD6303R, HD6303R1, HD63P01M1, AND HD63701V0 (Continued)

<table>
<tr><th></th><th>Item</th><th colspan="2">HD6301V</th><th>HD63701V0</th></tr>
<tr><td rowspan="3">Function</td><td rowspan="2">GND Noise</td><td>HD6301V1, HD6303R</td><td>HD6303R1,
HD63P01M1</td><td rowspan="2">Noise is reduced by 50%.</td></tr>
<tr><td> If load capacitance in each data line and GND impedance are large, noise may appear on address bus during MCU write cycle and data won't be written into RAM correctly. The noise is caused by GND impedance which becomes large when large transient current flows into GND at High to Low transition of data line.</td><td>Noise is reduced by 33%.</td></tr>
<tr><td>Miscellaneous</td><td colspan="3">Chip design and manufacturing process of the HD6301V differ from those of the HD63701V0. Therefore, actual spec. and margin are different between the HD6301V and the HD63701V0. Please carefully examine your system before applying HD6301V or HD63701V0 to your system.</td></tr>
</table>

HD6303X,HD63A03X, HD63B03X
CMOS MPU (Micro Processing Unit)

The HD6303X is a CMOS 8-bit micro processing unit (MPU) which includes a CPU compatible with the HD6301V1, 192 bytes of RAM, 24 parallel I/O pins, a Serial Communication Interface (SCI) and two timers on chip

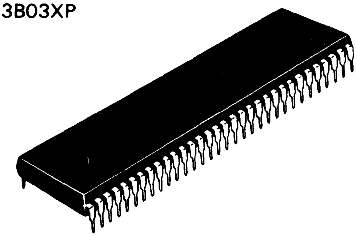
■ FEATURES

- Instruction Set Compatible with the HD6301V1
- 192 Bytes of RAM
- 24 Parallel I/O Pins
  16 I/O Pins-Port 2, 6
  8 Input Pins-Port 5
- Darlington Transistor Drive (Port 2, 6)
- 16-Bit Programmable Timer
  Input Capture Register x 1
  Free Running Counter x 1
  Output Compare Register x 2
- 8-Bit Reloadable Timer
  External Event Counter Square Wave Generation
- Serial Communication Interface
- Memory Ready
- Halt
- Error-Detection (Address Trap, Op-Code Trap)
- Interrupts . . . 3 External, 7 Internal
- Up to 65k Bytes Address Space
- Low Power Dissipation Mode
  Sleep Mode
  Standby Mode
- Minimum Instruction Execution Time -0.5μs (f = 2.0 MHz)
- Wide Range of Operation
  V_{CC} = 3 ~ 6V (f = 0.1 ~ 0.5 MHz).
  V_{CC} = 5V±10% { f = 0.1 ~ 1.0 MHz; HD6303X
  f = 0.1 ~ 1.5 MHz; HD63A03X
  f = 0.1 ~ 2.0 MHz; HD63B03X

■ PROGRAM DEVELOPMENT SUPPORT TOOLS

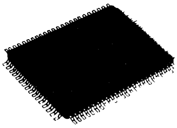
- Cross assembler and C compiler software for IBM PCs and compatibles
- In circuit emulator for use with IBM PCs and compatibles

HD6303XP,HD63A03XP,
HD63B03XP



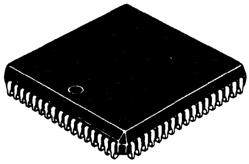
(DP-64S)

HD6303XF,HD63A03XF,
HD63B03XF



(FP-80)

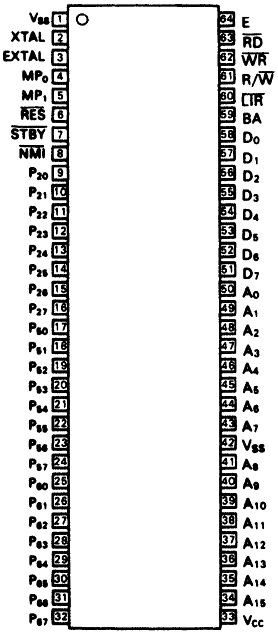
HD6303XCP,HD63A03XCP,
HD63B03XCP



(CP-68)

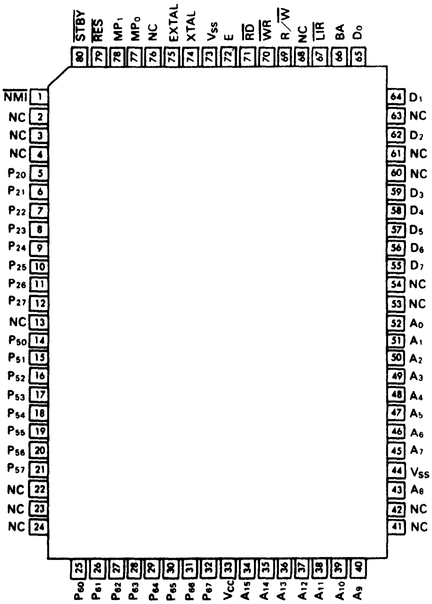
■ PIN ARRANGEMENT

● HD6303XP,HD63A03XP,HD63B03XP



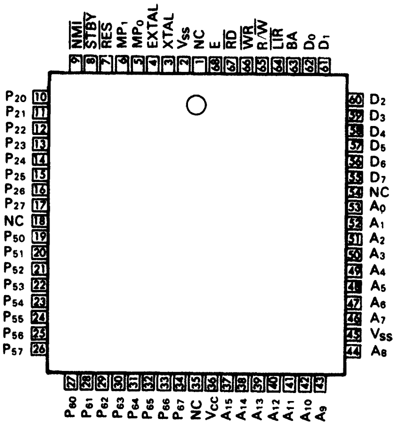
(Top View)

● HD6303XF,HD63A03XF,HD63B03XF



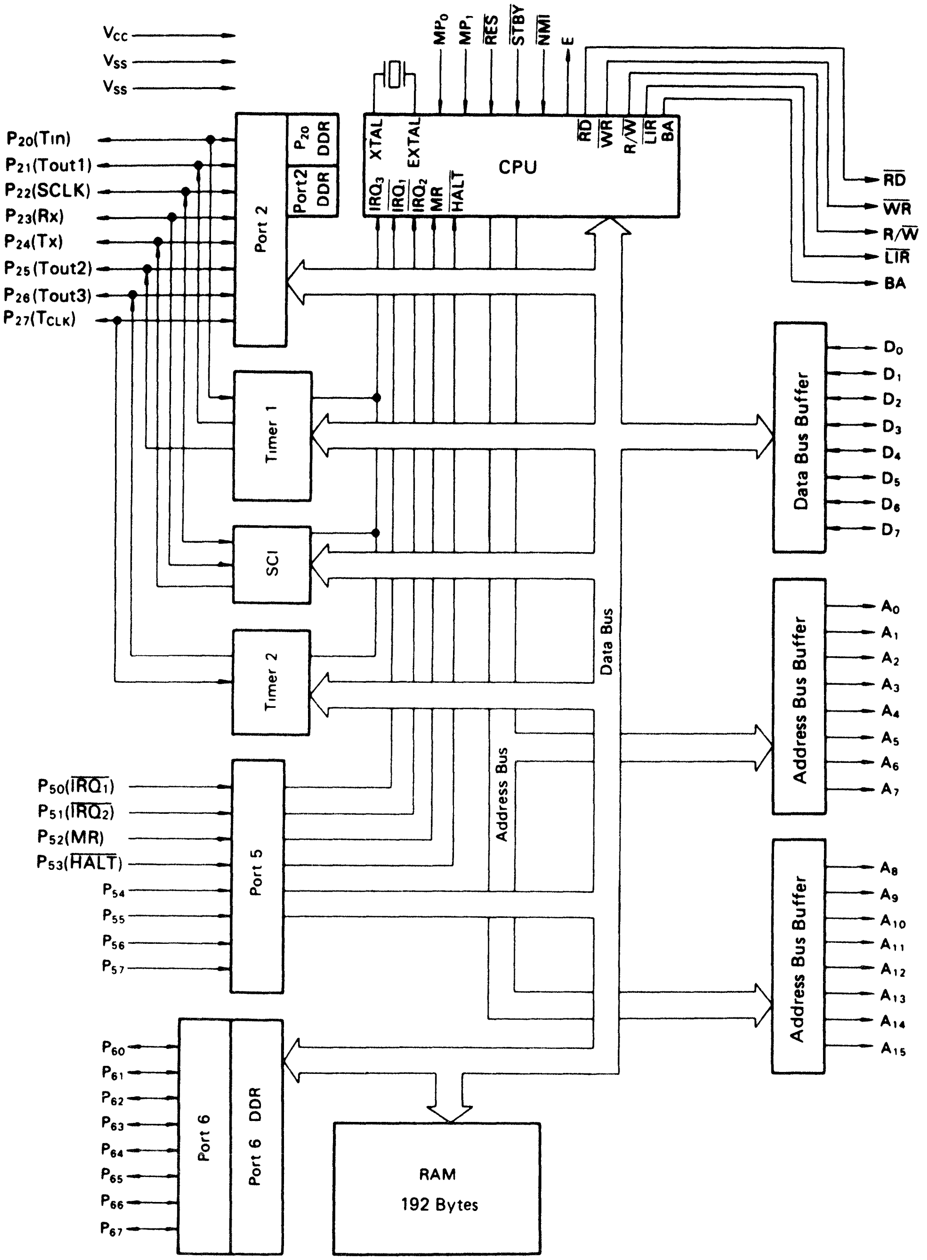
(Top View)

● HD6303XCP,HD63A03XCP,HD63B03XCP



(Top View)

■ BLOCK DIAGRAM



■ ABSOLUTE MAXIMUM RATINGS

Item	Symbol	Value	Unit
Supply Voltage	V_{CC}	-0.3 ~ +7.0	V
Input Voltage	V_{in}	-0.3 ~ V_{CC}+0.3	V
Operating Temperature	T_{opr}	0 ~ +70	°C
Storage Temperature	T_{stg}	-55 ~ +150	°C

(NOTE) This product has protection circuits in input terminal from high static electricity voltage and high electric field. But be careful not to apply overvoltage more than maximum ratings to these high input impedance protection circuits. To assure the normal operation, we recommend V_{in}, V_{out}: $V_{SS} \leqq (V_{in} \text{ or } V_{out}) \leqq V_{CC}$

■ ELECTRICAL CHARACTERISTICS

● DC CHARACTERISTICS (V_{CC} = 5.0V±10%, V_{SS} = 0V, Ta = 0 ~ +70°C, unless otherwise noted.)

Item		Symbol	Test Condition	min	typ	max	Unit
Input "High" Voltage	$\overline{RES}$, $\overline{STBY}$	V_{IH}		V_{CC}-0.5	–	V_{CC} +0.3	V
	EXTAL			V_{CC}x0.7	–		
	Other Inputs			2.0	–		
Input "Low" Voltage	All Inputs	V_{IL}		-0.3	–	0.8	V
Input Leakage Current	$\overline{NMI}$, $\overline{RES}$, $\overline{STBY}$, MP_0, MP_1, Port 5	$\lvert I_{in}\rvert$	V_{in} = 0.5 ~ V_{CC}-0.5V	–	–	1.0	μA
Three State (off-state) Leakage Current	A_0~A_{15}, D_0~D_7, $\overline{RD}$, $\overline{WR}$, R/$\overline{W}$, Port 2, Port 6	$\lvert I_{TSI}\rvert$	V_{in} = 0.5 ~ V_{CC}-0.5V	–	–	1.0	μA
Output "High" Voltage	All Outputs	V_{OH}	I_{OH} = -200μA	2.4	–	–	V
			I_{OH} = -10μA	V_{CC}-0.7	–	–	V
Output "Low" Voltage	All Outputs	V_{OL}	I_{OL} = 1.6mA	–	–	0.4	V
Darlington Drive Current	Ports 2, 6	$-I_{OH}$	Vout = 1.5V	1.0	–	10.0	mA
Input Capacitance	All Inputs	C_{in}	V_{in} = 0V, f = 1MHz, Ta = 25°C	–	–	12.5	pF
Standby Current	Non Operation	I_{STB}		–	3.0	15.0	μA
Current Dissipation*		I_{SLP}	Sleeping (f = 1MHz**)	–	1.5	3.0	mA
			Sleeping (f = 1.5MHz**)	–	2.3	4.5	mA
			Sleeping (f = 2MHz**)	–	3.0	6.0	mA
		I_{CC}	Operating (f = 1MHz**)	–	7.0	10.0	mA
			Operating (f = 1.5MHz**)	–	10.5	15.0	mA
			Operating (f = 2MHz**)	–	14.0	20.0	mA
RAM Standby Voltage		V_{RAM}		2.0	–	–	V

* V_{IH} min = V_{CC}-1 0V, V_{IL} max = 0 8V , All output terminals are at no load

** Current Dissipation of the operating or sleeping condition is proportional to the operating frequency. So the typ or max values about Current Dissipations at *x* MHz operation are decided according to the following formula,

typ value (f = *x* MHz) = typ value (f = 1MHz) x *x*
max value (f = *x* MHz) = max value (f = 1MHz) x *x*
(both the sleeping and operating)

- **AC CHARACTERISTICS (V_{CC} = 5.0V±10%, V_{SS} = 0V, Ta = 0 ~ +70°C, unless otherwise noted.)**

BUS TIMING

Item		Symbol	Test Condition	HD6303X			HD63A03X			HD63B03X			Unit
				min	typ	max	min	typ	max	min	typ	max	
Cycle Time		t_{cyc}	Fig. 1	1	–	10	0.666	–	10	0.5	–	10	µs
Enable Rise Time		t_{Er}		–	–	25	–	–	25	–	–	25	ns
Enable Fall Time		t_{Ef}		–	–	25	–	–	25	–	–	25	ns
Enable Pulse Width "High" Level*		PW_{EH}		450	–	–	300	–	–	220	–	–	ns
Enable Pulse Width "Low" Level*		PW_{EL}		450	–	–	300	–	–	220	–	–	ns
Address, R/$\overline{W}$ Delay Time*		t_{AD}		–	–	250	–	–	190	–	–	160	ns
Data Delay Time	Write	t_{DDW}		–	–	200	–	–	160	–	–	120	ns
Data Set-up Time	Read	t_{DSR}		80	–	–	70	–	–	70	–	–	ns
Address, R/$\overline{W}$ Hold Time*		t_{AH}		80	–	–	50	–	–	35	–	–	ns
Data Hold Time	Write*	t_{HW}		80	–	–	50	–	–	40	–	–	ns
	Read	t_{HR}		0	–	–	0	–	–	0	–	–	ns
$\overline{RD}$, $\overline{WR}$ Pulse Width*		PW_{RW}		450	–	–	300	–	–	220	–	–	ns
$\overline{RD}$, $\overline{WR}$ Delay Time		t_{RWD}		–	–	40	–	–	40	–	–	40	ns
$\overline{RD}$, $\overline{WR}$ Hold Time		t_{HRW}		–	–	30	–	–	30	–	–	25	ns
$\overline{LIR}$ Delay Time		t_{DLR}		–	–	200	–	–	160	–	–	120	ns
$\overline{LIR}$ Hold Time		t_{HLR}		10	–	–	10	–	–	10	–	–	ns
MR Set-up Time*		t_{SMR}	Fig. 2	400	–	–	280	–	–	230	–	–	ns
MR Hold Time*		t_{HMR}		–	–	90	–	–	40	–	–	0	ns
E Clock Pulse Width at MR		PW_{EMR}		–	–	9	–	–	9	–	–	9	µs
Processor Control Set-up Time		t_{PCS}	Fig. 3, 10, 11	200	–	–	200	–	–	200	–	–	ns
Processor Control Rise Time		t_{PCr}	Fig 2, 3	–	–	100	–	–	100	–	–	100	ns
Processor Control Fall Time		t_{PCf}		–	–	100	–	–	100	–	–	100	ns
BA Delay Time		t_{BA}	Fig. 3	–	–	250	–	–	190	–	–	160	ns
Oscillator Stabilization Time		t_{RC}	Fig. 11	20	–	–	20	–	–	20	–	–	ms
Reset Pulse Width		PW_{RST}		3	–	–	3	–	–	3	–	–	t_{cyc}

* These timings change in approximate proportion to t_{cyc} The figures in this characteristics represent those when t_{cyc} is minimum (= in the highest speed operation)

PERIPHERAL PORT TIMING

Item		Symbol	Test Condition	HD6303X			HD63A03X			HD63B03X			Unit
				min	typ	max	min	typ	max	min	typ	max	
Peripheral Data Set-up Time	Ports 2, 5, 6	t_{PDSU}	Fig. 5	200	–	–	200	–	–	200	–	–	ns
Peripheral Data Hold Time	Ports 2, 5, 6	t_{PDH}	Fig 5	200	–	–	200	-	–	200	–	–	ns
Delay Time (Enable Negative Transition to Peripheral Data Valid)	Ports 2, 6	t_{PWD}	Fig. 6	–	–	300	-	--	300	–	–	300	ns

TIMER, SCI TIMING

Item		Symbol	Test Condition	HD6303X			HD63A03X			HD63B03X			Unit
				min	typ	max	min	typ	max	min	typ	max	
Timer 1 Input Pulse Width		t_{PWT}	Fig. 8	2.0	–	–	2.0	–	–	2.0	–	–	t_{cyc}
Delay Time (Enable Positive Transition to Timer Output)		t_{TOD}	Fig. 7	–	–	400	–	–	400	–	–	400	ns
SCI Input Clock Cycle	Async. Mode	t_{Scyc}	Fig. 8	1.0	–	–	1.0	–	–	1.0	–	–	t_{cyc}
	Clock Sync.		Fig. 4, 8	2.0	–	–	2.0	–	–	2.0	–	–	t_{cyc}
SCI Transmit Data Delay Time (Clock Sync. Mode)		t_{TXD}	Fig. 4	–	–	200	–	–	200	–	–	200	ns
SCI Receive Data Set-up Time (Clock Sync. Mode)		t_{SRX}		290	–	–	290	–	–	290	–	–	ns
SCI Receive Data Hold Time (Clock Sync. Mode)		t_{HRX}		100	–	–	100	–	–	100	–	–	ns
SCI Input Clock Pulse Width		t_{PWSCK}	Fig. 8	0.4	–	0.6	0.4	–	0.6	0 4	–	0.6	t_{Scyc}
Timer 2 Input Clock Cycle		t_{tcyc}		2.0	–	–	2.0	–	–	2.0	–	–	t_{cyc}
Timer 2 Input Clock Pulse Width		t_{PWTCK}		200	–	–	200	–	–	200	–	–	ns
Timer 1·2, SCI Input Clock Rise Time		t_{CKr}		–	–	100	–	–	100	–	–	100	ns
Timer 1·2, SCI Input Clock Fall Time		t_{CKf}		–	–	100	–	–	100	–	–	100	ns

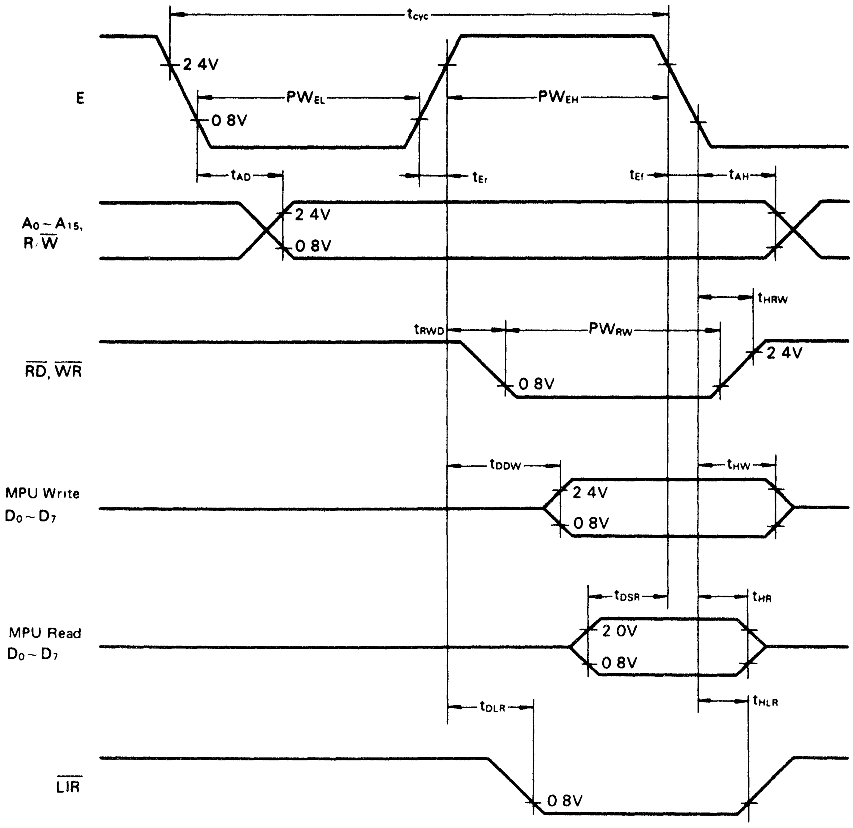


Figure 1 Bus Timing

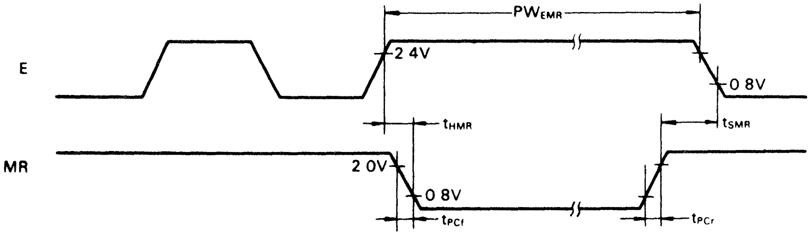


Figure 2 Memory Ready and E Clock Timing

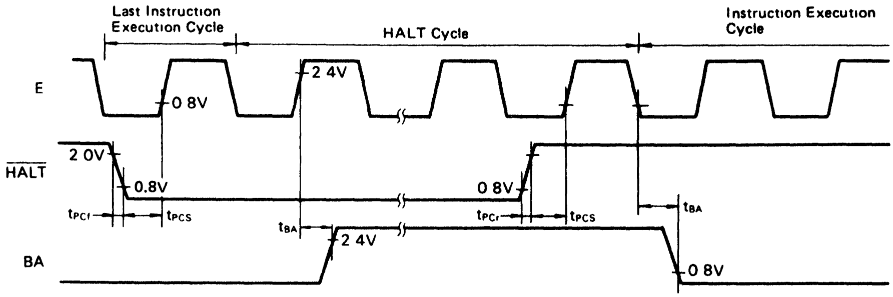


Figure 3 HALT and BA Timing

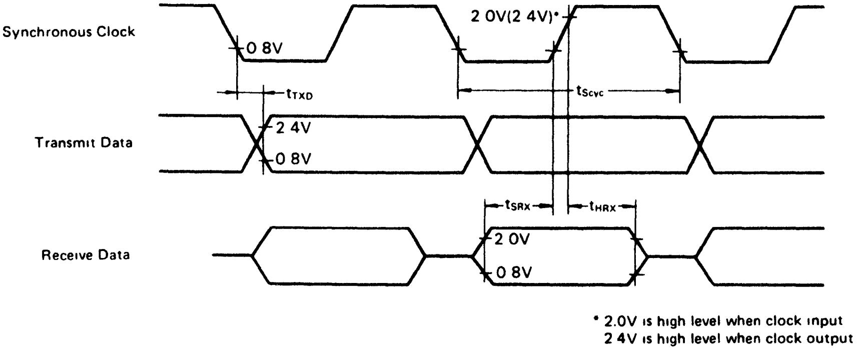


Figure 4 SCI Clocked Synchronous Timing

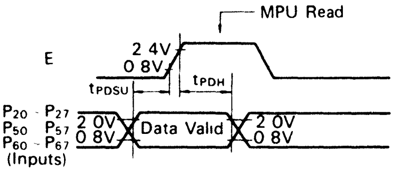


Figure 5 Port Data Set-up and Hold Times (MPU Read)

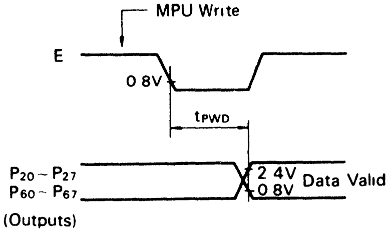
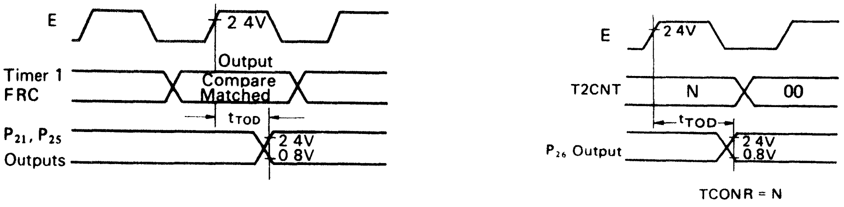


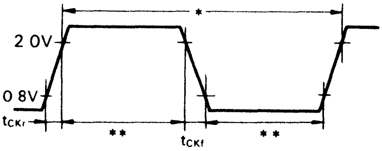
Figure 6 Port Data Delay Times (MPU Write)



(a) Tımer 1 Output Tımıng

(b) Tımer 2 Output Timing

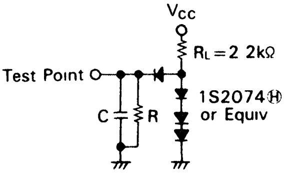
Fıgure 7 Tımer Output Tımıng



* Tımer 2, tтcyc
SCI , tScyc

** Tımer 1, tPWT
Tımer 2, tPWTCK
SCI , tPWSCK

Fıgure 8 Tımer 1·2, SCI Input Clock Tımıng



C = 90pF for D_0~D_7, A_0~A_{15}, E
=30pF for Port 2, Port 6, $\overline{RD}$, $\overline{WR}$, R/$\overline{W}$, BA, $\overline{LIR}$
R = 12kΩ

Fıgure 9 Bus Tımıng Test Loads (TTL Load)

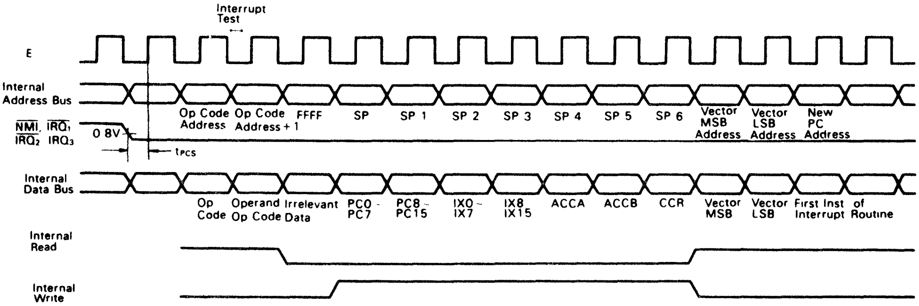


Figure 10 Interrupt Sequence

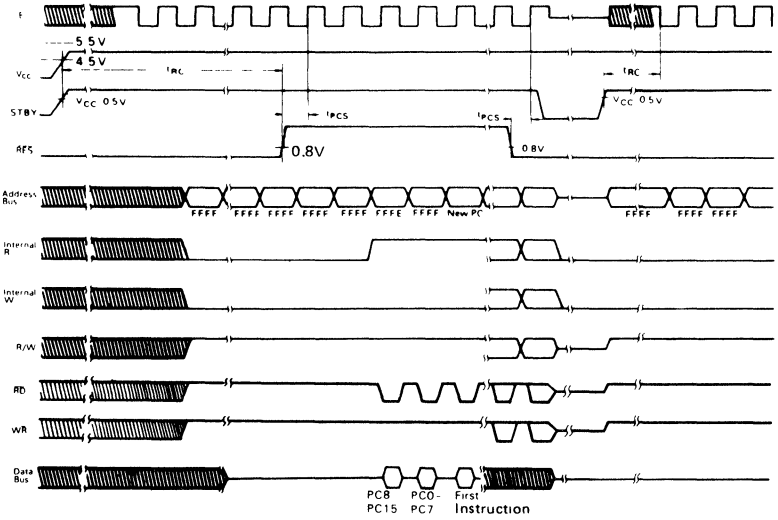


Figure 11 Reset Timing

■ **FUNCTIONAL PIN DESCRIPTION**

● **V_{CC}, V_{SS}**

V_{CC} and V_{SS} provide power to the MPU with 5V±10% supply. In the case of low speed operation (fmax = 500kHz), the MPU can operate with three through six volts. Two V_{SS} pins should be tied to ground.

● **XTAL, EXTAL**

These two pins interface with an AT-cut parallel resonant crystal Divide-by-four circuit is on chip, so if 4MHz crystal oscillator is used, the system clock is 1MHz for example.

AT Cut Parallel Resonant Crystal Oscillator

C_0=7pF max
R_S=60Ω max

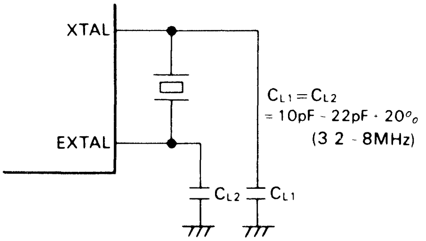


Figure 12 Crystal Interface

EXTAL pin can be drived by the external clock of 45 to 55% duty, and one fourth frequency of the external clock is produced in the LSI. The external clock frequency should be less than four times of the maximum operable frequency When using the external clock, XTAL pin should be open Fig. 12 shows an example of the crystal interface The crystal and C_{L1}, C_{L2} should be mounted as close as possible to XTAL and EXTAL pins. Any line must not cross the line between the crystal oscillator and XTAL, EXTAL.

● **STBY**

This pin makes the MPU standby mode. In "Low" level, the oscillation stops and the internal clock is stabilized to make reset condition To retain the contents of RAM at standby mode, "0" should be written into RAM enable bit (RAME). RAME is the bit 6 of the RAM/port 5 control register at $0014. RAM is disabled by this operation and its contents is sustained

Refer to "LOW POWER DISSIPATION MODE" for the standby mode.

● **Reset (RES)**

This pin resets the MPU from power OFF state and provides a startup procedure During power-on, RES pin must be held "Low" level for at least 20ms.

The CPU registers (accumulator, index register, stack pointer, condition code register except for interrupt mask bit), RAM and the data register of a port are not initialized during reset, so their contents are unknown in this procedure.

To reset the MPU during operation, RES should be held "Low" for at least 3 system-clock cycles. At the 3rd cycle during "Low" level, all the address buses become "High". When RES remains "Low", the address buses keep "High". If RES becomes "High", the MPU starts the next operation.

(1) Latch the value of the mode program pins; MP_0 and MP_1.
(2) Initialize each internal register (Refer to Table 3)
(3) Set the interrupt mask bit. For the CPU to recognize the maskable interrupts $\overline{IRQ_1}$, $\overline{IRQ_2}$ and IRQ_3, this bit should be cleared in advance.
(4) Put the contents (=start address) of the last two addresses ($FFFE, $FFFF) into the program counter and start the program from this address (Refer to Table 1).

*The MPU is usable to accept a reset input until the clock becomes normal oscillation after power on (max. 20ms). During this transient time, the MPU and I/O pins are undefined. Please be aware of this for system designing.

- **Enable (E)**

This pin provides a TTL-compatible system clock to external circuits. Its frequency is one fourth that of the crystal oscillator or external clock. This pin can drive one TTL load and 90pF capacitance.

- **Non-Maskable Interrupt ($\overline{\text{NMI}}$)**

When the falling edge of the input signal is detected at this pin, the CPU begins non-maskable interrupt sequence internally. As well as the IRQ mentioned below, the instruction being executed at $\overline{\text{NMI}}$ signal detection will proceed to its completion. The interrupt mask bit of the condition code register doesn't affect non-maskable interrupt at all.

When starting the acknowledge to the $\overline{\text{NMI}}$, the contents of the program counter, index register, accumulators and condition code register will be saved onto the stack. Upon completion of this sequence, a vector is fetched from $FFFC and $FFFD to transfer their contents into the program counter and branch to the non-maskable interrupt service routine

(Note) After reset start, the stack pointer should be initialized on an appropreate memory area and then the falling edge should be input to $\overline{\text{NMI}}$ pin.

- **Interrupt Request ($\overline{\text{IRQ}}_1$, $\overline{\text{IRQ}}_2$)**

These are level-sensitive pins which request an internal interrupt sequence to the CPU. At interrupt request, the CPU will complete the current instruction before its request acknowledgement. Unless the interrupt mask in the condition code register is set, the CPU starts an interrupt sequence; if set, the interrupt request will be ignored. When the sequence starts, the contents of the program counter, index register, accumulators and condition code register will be saved onto the stack, then the CPU sets the interrupt mask bit and will not acknowledge the maskable request. During the last cycle, the CPU fetches vectors depicted in Table 1 and transfers their contents to the program counter and branches to the service routine.

The CPU uses the external interrupt pins, $\overline{\text{IRQ}}_1$ and $\overline{\text{IRQ}}_2$, also as port pins P_{50} and P_{51}, so it provides an enable bit to Bit 0 and 1 of the RAM port 5 control register at $0014 Refer to "RAM/PORT 5 CONTROL REGISTER" for the details.

When one of the internal interrupts, ICI, OCI, TOI, CMI or SIO is generated, the CPU produces internal interrupt signal (IRQ_3) IRQ_3 functions just the same as $\overline{\text{IRQ}}_1$ or $\overline{\text{IRQ}}_2$ except for its vector address. Fig 13 shows the block diagram of the interrupt circuit.

Table 1 Interrupt Vector Memory Map

Priority	Vector		Interrupt
	MSB	LSB	
Highest	FFFE	FFFF	$\overline{\text{RES}}$
↑	FFEE	FFEF	TRAP
	FFFC	FFFD	$\overline{\text{NMI}}$
	FFFA	FFFB	SWI (Software Interrupt)
	FFF8	FFF9	$\overline{\text{IRQ}}_1$
	FFF6	FFF7	ICI (Timer 1 Input Capture)
	FFF4	FFF5	OCI (Timer 1 Output Compare 1, 2)
	FFF2	FFF3	TOI (Timer 1 Overflow)
	FFEC	FFED	CMI (Timer 2 Counter Match)
↓	FFEA	FFEB	$\overline{\text{IRQ}}_2$
Lowest	FFF0	FFF1	SIO (RDRF+ORFE+TDRE)

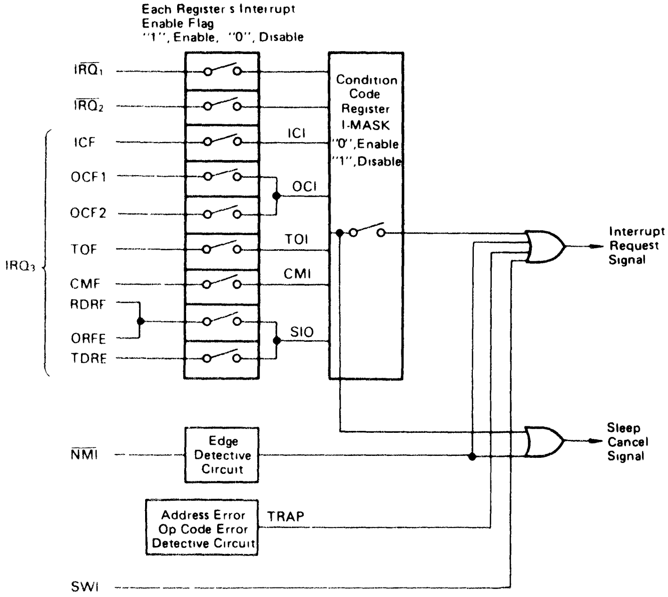


Figure 13 Interrupt Circuit Block Diagram

- **Mode Program (MP_0, MP_1)**

To operate MPU, MP_0 pin should be connected to "High" level and MP_1 should be connected to "Low" level (refer to Fig 15)

- **Read/Write (R/$\overline{W}$)**

This signal, usually be in read state ("High"), shows whether the CPU is in read ("High") or write ("Low") state to the peripheral or memory devices This can drive one TTL load and 30pF capacitance

- **$\overline{RD}$, $\overline{WR}$**

These signals show active low outputs when the CPU is reading/writing to the peripherals or memories This enables the CPU easy to access the peripheral LSI with $\overline{RD}$ and WR input pins These pins can drive one TTL load and 30pF capacitance

- **Load Instruction Register ($\overline{LIR}$)**

This signal shows the instruction opecode being on data bus (active low) This pin can drive one TTL load and 30pF capacitance

- **Memory Ready (MR, P_{52})**

This is the input control signal which stretches the system clock's "High" period to access low-speed memories. During this signal is in "High", the system clock operates in normal sequence But this signal in "Low", the "High" period of the system clock will be stretched depending on its "Low" level duration in integral multiples of the cycle time This allows the CPU to interface with low-speed memories (see Fig 2) Up to 9 μs can be stretched

During internal address space access or nonvalid memory access, MR is prohibited internally to prevent decrease of operation speed Even in the halt state, MR can also stretch "High" period of system clock to allow peripheral devices to access low-speed memories. As this signal is used also as P_{52}, an enable bit is provided at bit 2 of the RAM/port 5 control register at $0014. Refer to "RAM/PORT 5 CONTROL REGISTER" for more details

- **Halt ($\overline{HALT}$; P_{53})**

This is an input control signal to stop instruction execution and to release buses When this signal switches to "Low", the CPU stops to enter into the halt state after having executed the present instruction When entering into the halt state, it makes BA (P_{74}) "High" and also an address bus, data bus, $\overline{RD}$, $\overline{WR}$, R/$\overline{W}$ high impedance When an interrupt is generated in the halt state, the CPU uses the interrupt handler after the halt is cancelled

(Note) 1 Please don't switch the $\overline{HALT}$ signal to "Low" when the CPU executes the WAI instruction and is in the interrupt wait state to avoid the trouble of the CPU's operation after the halt is cancelled.

2 When power is supplied with the condition that $\overline{HALT}$ is "low", MCU cannot sometimes release the reset condition, even if $\overline{RESET}$ becomes "High". $\overline{HALT}$ should be low before $\overline{RESET}$ rises up.

- **Bus Available (BA)**

This is an output control signal which is normally "Low" but "High" when the CPU accepts $\overline{HALT}$ and releases the buses. The HD6800 and HD6802 make BA "High" and release the buses at WAI execution, while the HD6303X doesn't make BA "High" under the same condition. But if the $\overline{HALT}$ becomes

"Low" when the CPU is in the interrupt wait state after having executed the WAI, the CPU makes BA "High" and releases the buses. And when the $\overline{\text{HALT}}$ becomes "High", the CPU returns to the interrupt wait state.

■ PORT

The HD6303X provides three I/O ports. Table 2 gives the address of ports and the data direction register and Fig. 14 the block diagrams of each port.

Table 2 Port and Data Direction Register Address

Port	Port Address	Data Direction Register
Port 2	$0003	$0001
Port 5	$0015	–
Port 6	$0017	$0016

● Port 2

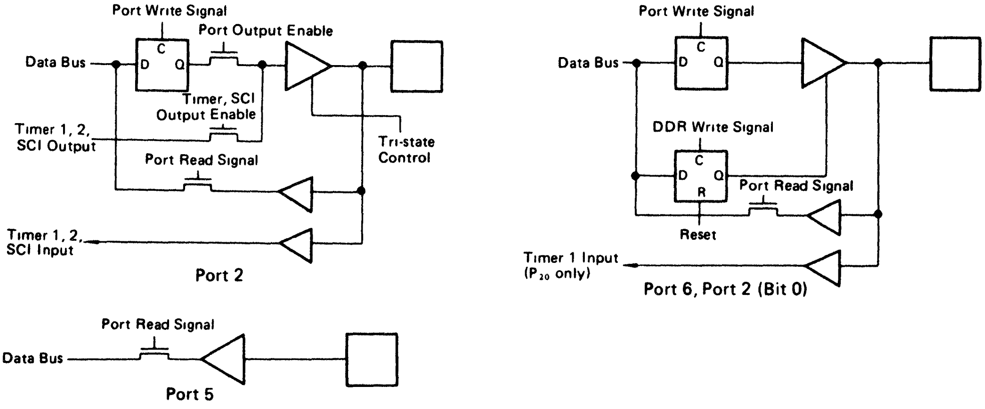
An 8-bit input/output port. The data direction register (DDR) of port 2 controls the I/O state. It provides two bits; bit 0 decides the I/O direction of P_{20} and bit 1 the I/O direction of P_{21} to P_{27} ("0" for input, "1" for output).

Port 2 is also used as an I/O pin for the timers and the SCI. When used as an I/O pin for the timers and the SCI, port 2 except P_{20} automatically becomes an input or an output depending on their functions regardless of the data direction register's value.

Port 2 Data Direction Register

7	6	5	4	3	2	1	0	
–	–	–	–	–	–	DDR 1~7	DDR 0	$0001

A reset clears the DDR of port 2 and configures port 2 as an input port. This port can drive one TTL and 30pF capacitance. In addition, it can produce 1mA current when V_{out} = 1.5V to drive directly the base of Darlington transistors.



Port 2

Port 6, Port 2 (Bit 0)

Port 5

Figure 14 Port Block Diagram

● Port 5

An 8-bit port for input only. The lower four bits are also usable as input pins for interrupt, MR and $\overline{\text{HALT}}$.

● Port 6

An 8-bit I/O port. This port provides an 8-bit DDR corresponding to each bit and can specify input or output by the bit ("0" for input, "1" for output). This port can drive one TTL load and 30pF capacitance. A reset clears the DDR of port 6. In addition, it can produce 1mA current when V_{out} = 1.5V to drive directly the base of Darlington transistors.

■ BUS

● D_0~D_7

These pins are data bus and can drive one TTL load and 90pF capacitance respectively.

● A_0~A_{15}

These pins are address bus and can drive one TTL load and 90pF capacitance respectively.

■ RAM/PORT 5 CONTROL REGISTER

The control register located at $0014 controls on-chip RAM and port 5.

RAM/Port 5 Control Register

7	6	5	4	3	2	1	0	
STBY PWR	RAME	–	–	HLTE	MRE	IRQ$_2$ E	IRQ$_1$ E	$0014

Bit 0, Bit 1 $\overline{\text{IRQ}_1}$, $\overline{\text{IRQ}_2}$ Enable Bit (IRQ_1E, IRQ_2E)

When using P_{50} and P_{51} as interrupt pins, write "1" in these bits. When "0", the CPU doesn't accept an external interrupt or a sleep cancellation by the external interrupt. These bits become "0" during reset.

Bit 2 Memory Ready Enable Bit (MRE)

When using P_{52} as an input for Memory Ready signal, write "1" in this bit. When "0". the memory ready function is prohibited and P_{52} can be used as I/O port This bit becomes "1" during reset.

Bit 3 Halt Enable bit (HLTE)

When using P_{53} as an input for Halt signal, write "1" in this bit. When "0", the halt function is prohibited and P_{53} can be used as I/O port. This bit becomes "1" during reset.

(Note) When using P_{52} and P_{53} as the input ports in mode 1 and 2, MRE and HLTE bit should be cleared just after the reset.

Notice that memory ready and halt function is enable till MRE and HLTE bit is cleared

Bit 4, Bit 5 Not Used.

Bit 6 RAM Enable (RAME)

On-chip RAM can be disabled by this control bit By resetting the MPU, "1" is set to this bit, and on-chip RAM is enabled. This bit can be written "1" or "0" by software. When RAM is in disable condition (= logic "0"), on-chip RAM is invalid and the CPU can read data from external memory. This bit should be "0" before getting into the standby mode to protect on-chip RAM data.

Bit 7 Standby Power Bit (STBY PWR)

When V_{CC} is not provided in standby mode, this bit is cleared. This is a flag for both read/write by software. If this bit is set before standby mode, and remains set even after returning from standby mode, V_{CC} voltage is provided during standby mode and the on-chip RAM data is valid

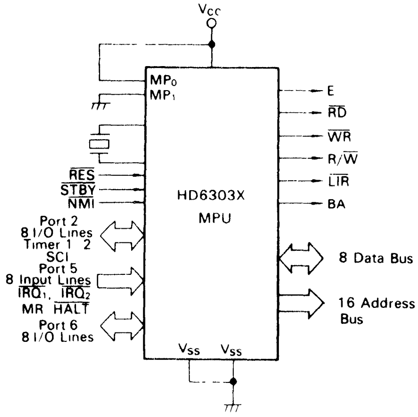


Figure 15 Operation Mode

■ MEMORY MAP

The MPU can address up to 65k bytes Fig 16 gives memory map of HD6303X. 32 internal registers use addresses from "00" as shown in Table 3

Table 3 Internal Register

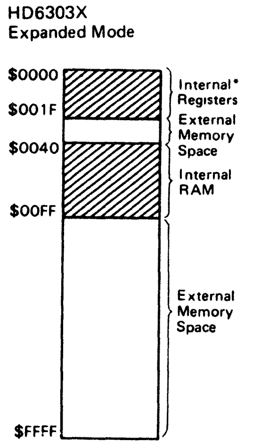
Address	Registers	R/W***	Initialize at RESET
00	–	–	–
01	Port 2 Data Direction Register	W	$FC
02*	–	–	–
03	Port 2	R/W	Undefined
04*	–	–	–
05	–	–	–
06*	–	–	–
07*	–	–	–
08	Timer Control/Status Register 1	R/W	$00
09	Free Running Counter ("High")	R/W	$00
0A	Free Running Counter ("Low")	R/W	$00
0B	Output Compare Register 1 ("High")	R/W	$FF
0C	Output Compare Register 1 ("Low")	R/W	$FF
0D	Input Capture Register ("High")	R	$00
0E	Input Capture Register ("Low")	R	$00
0F	Timer Control/Status Register 2	R/W	$10
10	Rate, Mode Control Register	R/W	$00
11	Tx/Rx Control Status Register	R/W	$20
12	Receive Data Register	R	$00
13	Transmit Data Register	W	$00
14	RAM/Port 5 Control Register	R/W	$7C or $FC
15	Port 5	R	–
16	Port 6 Data Direction Register	W	$00

(continued)

Table 3 Internal Register

Address	Registers	R/W***	Initialize at RESET
17	Port 6	R/W	Undefined
18*	–	–	–
19	Output Compare Register 2 ("High")	R/W	$FF
1A	Output Compare Register 2 ("Low")	R/W	$FF
1B	Timer Control/Status Register 3	R/W	$20
1C	Time Constant Register	W	$FF
1D	Timer 2 Up Counter	R/W	$00
1E	–	–	–
1F**	Test Register	–	–

* External Address
** Test Register Do not access to this register.
*** R : Read Only Register
W : Write Only Register
R/W : Read/Write Register



* Excludes the following addresses which may be used externally:
$02, $04, $06, $07, $18

Figure 16 HD6303X Memory Map

■ TIMER 1

The HD6303X provides a 16-bit programmable timer which can simultaneously measure an input waveform and generate two independent output waveforms. The pulse widths of both input/output waveforms vary from microseconds to seconds.

Timer 1 is configurated as follows (refer to Fig. 18)

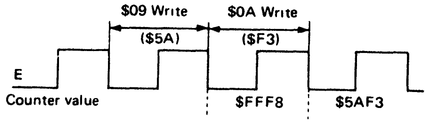
- Control/Status Register 1 (8 bit)
- Control/Status Register 2 (7 bit)
- Free Running Counter (16 bit)
- Output Compare Register 1 (16 bit)
- Output Compare Register 2 (16 bit)
- Input Capture Register (16 bit)

● Free-Running Counter (FRC) ($0009 : 000A)

The key timer element is a 16-bit free-running counter driven and incremented by system clock. The counter value is readable by software without affecting the counter. The counter is cleared by reset.

When writing to the upper byte ($09), the CPU writes the preset value ($FFF8) into the counter (address $09, $0A) regardless of the write data value. But when writing to the lower byte ($0A) after the upper byte writing, the CPU writes not only the lower byte data into lower 8 bit, but also the upper byte data into higher 8 bit of the FRC.

The counter will be as follows when the CPU writes to it by double store instructions (STD, STX etc.).



In the case of the CPU write ($5AF3) to the FRC

Figure 17 Counter Write Timing

● Output Compare Register (OCR) ($000B, $000C; OCR1) ($0019, $001A ; OCR2)

The output compare register is a 16-bit read/write register which can control an output waveform. The data of OCR is always compared with the FRC.

When the data matches, output compare flag (OCF) in the timer control/status register (TCSR) is set. If an output enable bit (OE) in the TCSR2 is "1", an output level bit (OLVL) in the TCSR will be output to bit 1 (Tout 1) and bit 5 (Tout 2) of port 2. To control the output level again by the next compare, the value of OCR and OLVL should be changed. The OCR is set to $FFFF at reset. The compare function is inhibited for a cycle just after a write to the OCR or to the upper byte of the FRC. This is to begin the comparison after setting the 16-bit value valid in the register and to inhibit the compare function at this cycle, because the CPU writes the upper byte to the FRC, and at the next cycle the counter is set to $FFF8.

* For data write to the FRC or the OCR, 2-byte transfer instruction (such as STX etc) should be used.

● Input Capture Register (ICR) ($000D : 000E)

The input capture register is a 16-bit read only register which stores the FRC's value when external input signal transition generates an input capture pulse. Such transition is controlled by input edge bit (IEDG) in the TCSR1.

In order to input the external input signal to the edge detecter, a bit of the DDR corresponding to bit 0 of port 2 should be cleared ("0"). When an input capture pulse occurs by the external input signal transition at the next cycle of CPU's high-byte read of the ICR, the input capture pulse will be delayed by one cycle. In order to ensure the input capture operation, a CPU read of the ICR needs 2-byte transfer instruction. The input pulse width should be at least 2 system cycles. This register is cleared ($0000) during reset

- **Timer Control/Status Register 1 (TCSR1) ($0008)**

The timer control/status register 1 is an 8-bit register. All bits are readable and the lower 5 bits are also writable. The upper 3 bits are read only which indicate the following timer status.

Bit 5 The counter value reached to $0000 as a result of counting-up (TOF)

Bit 6 A match has occured between the FRC and the OCR 1 (OCF1)

Bit 7 Defined transition of the timer input signal causes the counter to transfer its data to the ICR (ICF).

The followings are each bit descriptions.

Timer Control/Status Register 1

7	6	5	4	3	2	1	0	
ICF	OCF1	TOF	EICI	EOCI1	ETOI	IEDG	OLVL1	$0008

Bit 0 OLVL1 Output Level 1

OLVL1 is transferred to port 2, bit 1 when a match occurs between the counter and the OCR1. If bit 0 of the TCSR2 (OE1) is set to "1", OLVL1 will appear at bit 1 of port 2

Bit 1 IEDG Input Edge

This bit determines which edge, rising or falling, of input signal of port 2, bit 0 will trigger data transfer from the counter to the ICR. For this function, the DDR corresponding to port 2, bit 0 should be cleared beforehand

IEDG=0, triggered on a falling edge ("High" to "Low")

IEDG=1, triggered on a rising edge ("Low" to "High")

Bit 2 ETOI Enable Timer Overflow Interrupt

When this bit is set, an internal interrupt (IRQ_3) by TOI interrupt is enabled When cleared, the interrupt is inhibited

Bit 3 EOCI1 Enable Output Compare Interrupt 1

When this bit is set, an internal interrupt (IRQ_3) by OCI1 interrupt is enabled. When cleared, the interrupt is inhibited.

Bit 4 EICI Enable Input Capture Interrupt

When this bit is set, an internal interrupt (IRQ_3) by ICI interrupt is enabled. When cleared, the interrupt is inhibited.

Bit 5 TOF Timer Overflow Flag

This read-only bit is set when the counter increments from $FFFF by 1. Cleared when the counter's upper byte ($0009) is ready by the CPU after the TCSR1 read

Bit 6 OCF1 Output Compare Flag 1

This read-only bit is set when a match occurs between the OCR1 and the FRC. Cleared when writing to the OCR1 ($000B or $000C) after the TCSR1 or TCSR2 read.

Bit 7 ICF Input Capture Flag

This read-only bit is set when an input signal of port 2, bit 0 makes a transition as defined by IEDG and the FRC is transferred to the ICR. Cleared when reading the upper byte ($000D) of the ICR following the TCSR1 or TCSR2 read.

- **Timer Control/Status Register 2 (TCSR2) ($000F)**

The timer control/status register 2 is a 7-bit register. All bits are readable and the lower 4 bits are also writable. But the upper 3 bits are read-only which indicate the following timer status.

Bit 5 A match has occured between the FRC and the OCR2 (OCF2).

Bit 6 The same status flag as the OCF1 flag of the TCSR1, bit 6.

Bit 7 The same status flag as the ICF flag of the TCSR1, bit 7.

The followings are the each bit descriptions.

Timer Control/Status Register 2

7	6	5	4	3	2	1	0	
ICF	OCF1	OCF2	–	EOCI2	OLVL2	OE2	OE1	$000F

Bit 0 OE1 Output Enable 1

This bit enables the OLVL1 to appear at port 2, bit 1 when a match has occurred between the counter and the output compare register 1 When this bit is cleared, bit 1 of port 2 will be an I/O port. When set, it will be an output of OLVL1 automatically.

Bit 1 OE2 Output Enable 2

This bit enables the OLVL2 to appear at port 2, bit 5 when a match has occurred between the counter and the output compare register 2. When this bit is cleared, port 2, bit 5 will be an I/O port. When set, it will be an output of OLVL2 automatically.

Bit 2 OLVL2 Output Level 2

OLVL2 is transferred to port 2, bit 5 when a match has occurred between the counter and the OCR2. If bit 5 of the TCSR2 (OE2) is set to "1", OLVL2 will appear at port 2, bit 5.

Bit 3 EOCI2 Enable Output Compare Interrupt 2

When this bit is set, an internal interrupt (IRQ_3) by OCI2 interrupt is enabled. When cleared, the interrupt is inhibited.

Bit 4 Not Used

Bit 5 OCF2 Output Compare Flag 2

This read-only bit is set when a match has occurred between the counter and the OCR2. Cleared when writing to the OCR2 ($0019 or $001A) after the TCSR2 read.

Bit 6 OCF1 Output Compare Flag 1

Bit 7 ICF Input Capture Flag

OCF1 and ICF addresses are partially decoded. The CPU read of the TCSR1/TCSR2 makes it possible to read OCF1 and ICF into bit 6 and bit 7.

Both the TCSR1 and TCSR2 will be cleared during reset.

(Note) If OE1 or OE2 is set to "1" before the first output compare match occurs after reset restart, bit 1 or bit 5 of port 2 will produce "0" respectively.

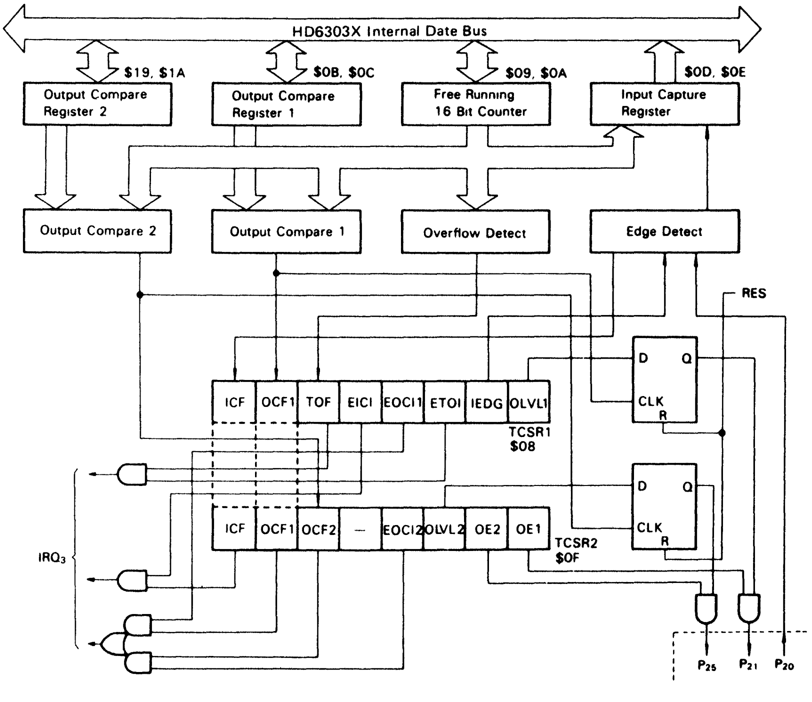


Figure 18 Timer 1 Block Diagram

■ **TIMER 2**

In addition to the timer 1, the HD6303X provides an 8-bit reloadable timer, which is capable of counting the external event. This timer 2 contains a timer output, so the MPU can generate three independent waveforms (refer to Fig. 19).

The timer 2 is configured as follows.

Control/Status Register 3 (7 bit)
8-bit Up Counter
Time Constant Register (8 bit)

● **Timer 2 Up Counter (T2CNT) ($001D)**

This is an 8-bit up counter which operates with the clock decided by CKS0 and CKS1 of the TCSR3. The CPU can read the value of the counter without affecting the counter. In addition, any value can be written to the counter by software even during counting.

The counter is cleared when a match occurs between the counter and the TCONR or during reset.

If a write operation is made by software to the counter at the cycle of counter clear, it does not reset the counter but put the write data to the counter.

● **Time Constant Register (TCONR) ($001C)**

The time constant register is an 8-bit write only register. It is always compared with the counter.

When a match has occurred, counter match flag (CMF) of the timer control status register 3 (TCSR3) is set and the value selected by TOS0 and TOS1 of the TCSR3 will appear at port 2, bit 6. When CMF is set, the counter will be cleared simultaneously and then start counting from $00. This enables regular interrupts and waveform outputs without any software support. The TCONR is set to "$FF" during reset.

● **Timer Control/Status Register 3 (TCSR3) ($001B)**

The timer control/status register 3 is a 7-bit register. All bits are readable and 6 bits except for CMF can be written.

The followings are each pin descriptions.

Timer Control/Status Register 3

7	6	5	4	3	2	1	0	
CMF	ECMI	—	T2E	TOS1	TOS0	CKS1	CKS0	$001B

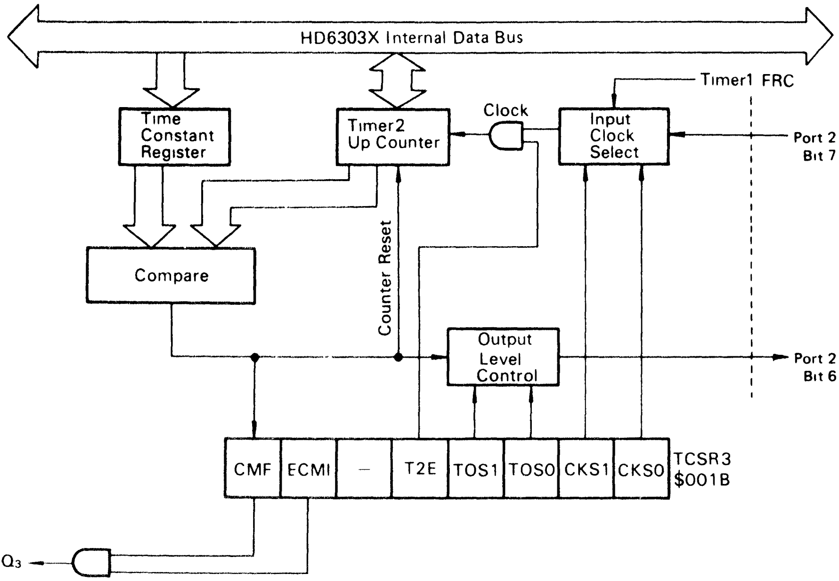


Figure 19 Timer 2 Block Diagram

Bit 0 CKSO Input Clock Select 0
Bit 1 CKS1 Input Clock Select 1

Input clock to the counter is selected as shown in Table 4 depending on these two bits. When an external clock is selected, bit 7 of port 2 will be a clock input automatically. Timer 2 detects the rising edge of the external clock and increments the counter. The external clock is countable up to half the frequency of the system clock

Table 4 Input Clock Select

CKS1	CKSO	Input Clock to the Counter
0	0	E clock
0	1	E clock/8*
1	0	E clock/128*
1	1	External clock

* These clocks come from the FRC of the timer 1 If one of these clocks is selected as an input clock to the up counter, the CPU should not write to the FRC of the timer 1

Bit 2 TOSO Timer Output Select 0
Bit 3 TOS1 Timer Output Select 1

When a match occurs between the counter and the TCONR timer 2 outputs shown in Table 5 will appear at port 2, bit 6 depending on these two bits When both TOSO and TOS1 are "0", bit 6 of port 2 will be an I/O port.

Table 5 Timer 2 Output Select

TOS1	TOSO	Timer Output
0	0	Timer Output Inhibited
0	1	Toggle Output*
1	0	Output "0"
1	1	Output "1"

* When a match occurs between the counter and the TCONR, timer 2 output level is reversed This leads to production of a square wave with 50% duty to the external without any software support

Bit 4 T2E Timer 2 Enable Bit

When this bit is cleared, a clock input to the up counter is prohibited and the up counter stops. When set to "1", a clock selected by CKS1 and CKSO (Table 4) is input to the up counter

(Note) P_{26} outputs "0" when T2E bit cleared and timer 2 set in output enable condition by TOS1 or TOSO. It also outputs "0" when T2E bit set "1" and timer 2 set in output enable condition before the first counter match occurs

Bit 5 Not Used

Bit 6 ECMI Enable Counter Match Interrupt

When this bit is set, an internal interrupt (IRQ3) by CMI is enabled When cleared, the interrupt is inhibited.

Bit 7 CMF Counter Match Flag

This read-only bit is set when a match occurs between the up counter and the TCONR. Cleared by writing "0" by software write (unable to write "1" by software).

Each bit of the TCSR3 is cleared during reset.

■ SERIAL COMMUNICATION INTERFACE (SCI)

The HD6303X SCI contains two operation modes; one is an asynchronous mode by the NRZ format and the other is a clocked synchronous mode which transfers data synchronizing with the serial clock.

The SCI consists of the following registers as shown in Fig. 20 Block Diagram.

- Control/Status Register (TRCSR)
- Rate/Mode Control Register (RMCR)
- Receive Data Register (RDR)
- Receive Data Shift Register (RDSR)
- Transmit Data Register (TDR)
- Transmit Data Shift Register (TDSR)

The serial I/O hardware requires an initialization by software for operation. The procedure is usually as follows.

1) Write a desirable operation mode into each corresponding control bit of the RMCR.
2) Write a desirable operation mode into each corresponding control bit of the TRCSR.

When using bit 3 and 4 of port 2 for serial I/O only, there is no problem even if TE and RE bit are set. But when setting the baud rate and operation mode, TE and RE should be "0". When clearing TE and RE bit and setting them again, more than 1 bit cycle of the current baud rate is necessary. If set in less than 1 bit cycle, there may be a case that the internal transmit/receive initialization fails.

● Asynchronous Mode

An asynchronous mode contains the following two data formats:

1 Start Bit + 8 Bit Data + 1 Stop Bit
1 Start Bit + 9 Bit Data + 1 Stop Bit

In addition, if the 9th bit is set to "1" when making 9 bit data format, the format of

1 Start bit + 8 Bit Data + 2 Stop Bit

is also transferred.

Data transmission is enabled by setting TE bit of the TRCSR, then port 2, bit 4 will become a serial output independently of the corresponding DDR.

For data transmit, both the RMCR and TRCSR should be set under the desirable operating conditions. When TE bit is set during this process, 10 bit preamble will be sent in 8-bit data format and 11 bit in 9-bit data format. When the preamble is produced, the internal synchronization will become stable and the transmitter is ready to act.

The conditions at this stage are as follows.

1) If the TDR is empty (TDRE=1), consecutive 1's are produced to indicate the idle state.
2) If the TDR contains data (TDRE=0), data is sent to the transmit data shift register and data transmit starts.

During data transmit, a start bit of "0" is transmitted first. Then 8-bit or 9-bit data (starts from bit 0) and a stop bit "1" are transmitted.

When the TDR is "empty", hardware sets TDRE flag bit. If the CPU doesn't respond to the flag in proper timing (the TDRE is in set condition till the next normal data transfer starts from the transmit data register to the transmit sift register), "1" is transferred instead of the start bit "0" and continues to be transferred till data is provided to the data register. While the TDRE is "1", "0" is not transferred.

Data receive is possible by setting RE bit. This makes port 2, bit 3 be a serial input. The operation mode of data receive is decided by the contents of the TRCSR and RMCR. The first "0" (space) synchronizes the receive bit flow. Each bit of the following data will be strobed in the middle. If a stop bit is not "1", a framing error assumed and ORFE is set

When a framing error occurs, receive data is transferred to the receive data register and the CPU can read error-generating data. This makes it possible to detect a line break.

If the stop bit is "1", data is transferred to the receive data register and an interrupt flag RDRF is set. If RDRF is still set when receiving the stop bit of the next data, ORFE is set to indicate overrun generation.

When the CPU read the receive data register as a response to RDRF flag or ORFE flag after having read TRCS, RDRF or ORFE is cleared.

(Note) Clock Source in Asynchronous Mode

If CC1 . CC0 = 10, the internal bit rate clock is provided at P_{22} regardless of the values for TE or RE. Maximum clock rate is E ÷ 16.

If both CC1 and CC0 are set, an external TTL compatible clock must be connected to P_{22} at sixteen times (16×) the desired bit rate, but not greater than E.

● Clocked Synchronous Mode

In the clocked synchronous mode, data transmit is synchronized with the clock pulse. The HD6303X SCI provides functionally independent transmitter and receiver which makes full duplex operation possible in the asynchronous mode. But in the clocked synchronous mode an SCI clock I/O pin is only P_{22}, so the simultaneous receive and transmit operation is not available. In this mode, TE and RE should not be in set condition ("1") simultaneously. Fig. 21 gives a synchronous clock and a data format in the clocked synchronous mode.

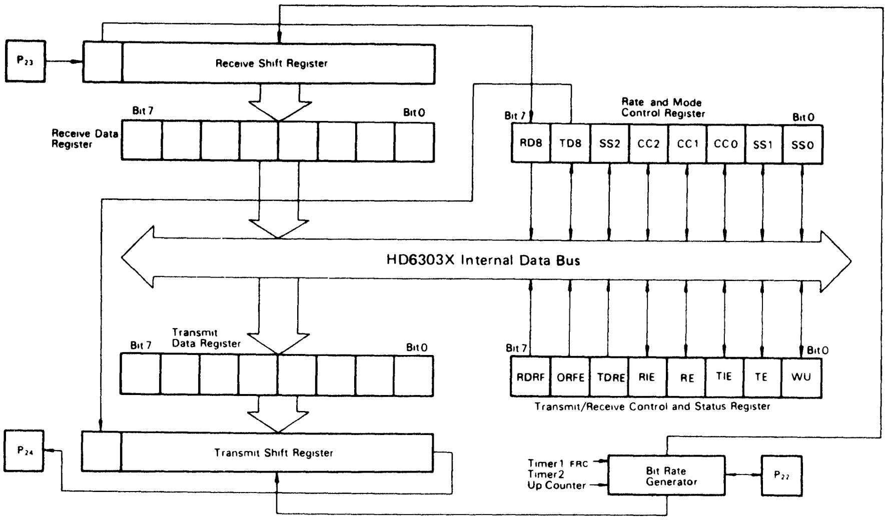


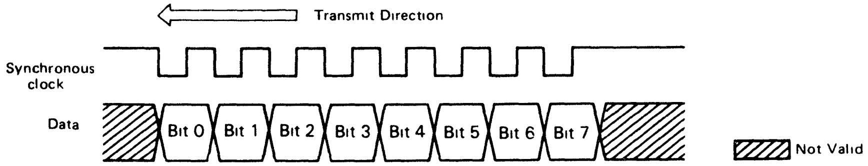
Figure 20 Serial Communication Interface Block Diagram

Data transmit is realized by setting TE bit in the TRCSR. Port 2, bit 4 becomes an output unconditionally independent of the value of the corresponding DDR.

Both the RMCR and TRCSR should be set in the desirable operating condition for data transmit.

When an external clock input is selected, data transmit is performed under the TDRE flag "0" from port 2, bit 4, synchronizing with 8 clock pulses input from external to port 2, bit 2.

Data is transmitted from bit 0 and the TDRE is set when the transmit data shift register is "empty". More than 9th clock pulse of external are ignored.



- Transmit data is output from a falling edge of a synchronous clock to the next falling edge
- Receive data is latched at the rising edge

Figure 21 Clocked Synchronous Mode Format

When data transmit is selected to the clock output, the MPU produces transmit data and synchronous clock at TDRE flag clear.

Data receive is enabled by setting RE bit. Port 2, bit 3 will be a serial input. The operating mode of data receive is decided by the TRCSR and the RMCR.

If the external clock input is selected, RE bit should be set when P_{22} is "High". Then 8 external clock pulses and the synchronized receive data are input to port 2, bit 2 and bit 3 respectively The MPU put receive data into the receive data shift register by this clock and set the RDRF flag at the termination of 8 bit data receive More than 9th clock pulse of external input are ignored When RDRF is cleared by reading the receive data register, the MPU starts receiving the next data. So RDRF should be cleared with P_{22} "High"

When data receive is selected to the clock output, 8 synchronous clocks are output to the external by setting RE bit. So receive data should be input from external, synchronously with this clock. When the first byte data is received, the RDRF flag is set. After the second byte, receive operation is performed and output the synchronous clock to the external by clearing the RDRF bit.

- **Transmit/Receive Control Status Register (TRCSR) ($0011)**

The TRCSR is composed of 8 bits which are all readable. Bits 0 to 4 are also writable. This register is initialized to $20 during reset. Each bit functions as follows.

Transmit/Receive Control Status Register

7	6	5	4	3	2	1	0	
RDRF	ORFE	TDRE	RIE	RE	TIE	TE	WU	$0011

Bit 0 WU Wake-up

In a typical multi-processor configuration, the software protocol provides the destination address at the first byte of the message In order to make uninterested MPU ignore the remaining message, a wake-up function is available. By this, uninterested MPU can inhibit all further receive processing till the next message starts.

Then wake-up function is triggered by consecutive 1's with 1 frame length (10 bits for 8-bit data, 11 for 9-bit). The software protocol should provide the idle time between messages.

By setting this bit, the MPU stops data receive till the next message. The receive of consecutive "1" with one frame length wakes up and clears this bit and then the MPU restarts receive operation. However, the RE flag should be already set before setting this bit. In the clocked synchronous mode WU is not available, so this bit should not be set.

Bit 1 TE Transmit Enable

When this bit is set, transmit data will appear at port 2, bit 4 after one frame preamble in asynchronous mode, while in clocked synchronous mode it appears immediately. This is executed regardless of the value of the corresponding DDR. When TE is cleared, the serial I/O doesn't affect port 2, bit 4.

Bit 2 TIE Transmit Interrupt Enable

When this bit is set, an internal interrupt (IRQ_3) is enabled when TDRE (bit 5) is set. When cleared, the interrupt is inhibited.

Bit 3 RE Receive Enable

When set, a signal is input to the receiver from port 2, bit 3 regardless of the value of the DDR. When RE is cleared, the serial I/O doesn't affect port 2, bit 3.

Bit 4 RIE Receive Interrupt Enable

When this bit is set, an internal interrupt, IRQ_3 is enabled when RDRF (bit 7) or ORFE (bit 6) is set. When cleared, the interrupt is inhibited.

Bit 5 TDRE Transmit Data Register Empty

TDRE is set when the TDR is transferred to the transmit data shift register in the asynchronous mode, while in clocked synchronous mode when the TDSR is "empty". This bit is reset by reading the TRCSR and writing new transmit data to the transmit data register. TDRE is set to "1" during reset.

(Note) TDRE should be cleared in the transmittable state after the TE set.

Bit 6 ORFE Overrun Framing Error

ORFE is set by hardware when an overrun or a framing error is generated (during data receive only). An overrun error occurs when new receive data is ready to be transferred to the RDR during RDRF still being set. A framing error occurs when a stop bit is "0". But in clocked synchronous mode, this bit is not affected. This bit is cleared when reading the TRCSR, then the RDR, or during reset.

Bit 7 RDRF Receive Data Register Full

RDRF is set by hardware when the RDSR is transferred to the RDR. Cleared when reading the TRCSR, then the RDR, or during reset.

(Note) When a few bits are set between bit 5 to bit 7 in the TRCSR, a read of the TRCSR is sufficient for clearing those bits. It is not necessary to read the TRCSR everytime to clear each bit.

- **Transmit Rate/Mode Control Register (RMCR)**

The RMCR controls the following serial I/O:

• Baud Rate	• Data Format
• Clock Source	• Port 2, Bit 2 Function

In addition, if 9-bit data format is set in the asynchronous mode, the 9th bit is put in this register. All bits are readable and writable except bit 7 (read only). This register is set to $00 during reset.

Transfer Rate/Mode Control Register

7	6	5	4	3	2	1	0	
RD8	TD8	SS2	CC2	CC1	CC0	SS1	SS0	$0010

Bit 0 SS0 }
Bit 1 SS1 } Speed Select
Bit 5 SS2 }

These bits control the baud rate used for the SCI. Table 6 lists the available baud rates. The timer 1 FRC (SS2=0) and the timer 2 up counter (SS2=1) provide the internal clock to the SCI. When selecting the timer 2 as a baud rate source, it functions as a baud rate generator. The timer 2 generates the baud rate listed in Table 7 depending on the value of the TCONR.

(Note) When operating the SCI with internal clock, do not perform write operation to the timer/counter which is the clock source of the SCI.

Bit 2 CC0 }
Bit 3 CC1 } Clock Control/Format Select*
Bit 4 CC2 }

These bits control the data format and the clock source (refer to Table 8)

* CC0, CC1 and CC2 are cleared during reset and the MPU goes to the clocked synchronous mode of the external clock operation. Then the MPU sets port 2, bit 2 into the clock input state. When using port 2, bit 2 as an output port, the DDR of port 2 should be set to "1" and CC1 and CC0 to "0" and "1" respectively.

Table 6 SCI Bit Times and Transfer Rates

(1) Asynchronous Mode

			XTAL	2 4576MHz	4 0MHz	4 9152MHz
SS2	SS1	SS0	E	614 4kHz	1 0MHz	1 2288MHz
0	0	0	E÷16	26μs/38400Baud	16μs/62500Baud	13μs/76800Baud
0	0	1	E÷128	208μs/4800Baud	128μs/7812 5Baud	104 2μs/9600Baud
0	1	0	E÷1024	1 67ms/600Baud	1 024ms 976 6Baud	833 3μs/1200Baud
0	1	1	E÷4096	6 67ms/150Baud	4 096ms/244 1Baud	3 333ms/300Baud
1	—	—	—	*	*	*

* When SS2 is "1", Timer 2 provides SCI clocks. The baud rate is shown as follows with the TCONR as N

$$\text{Baud Rate} = \frac{f}{32\,(N+1)} \qquad \left(\begin{array}{l} f \ \text{input clock frequency to the} \\ \quad \text{timer 2 counter} \\ N = 0 \sim 255 \end{array}\right)$$

(2) Clocked Synchronous Mode *

			XTAL	4 0MHz	6 0MHz	8 0MHz
SS2	SS1	SS0	E	1 0MHz	1 5MHz	2 0MHz
0	0	0	E÷2	2μs/bit	1 33μs/bit	1μs/bit
0	0	1	E÷16	16μs/bit	10 7μs/bit	8μs/bit
0	1	0	E÷128	128μs/bit	85 3μs/bit	64μs/bit
0	1	1	E÷512	512μs/bit	341μs/bit	256μs bit
1	—	—	—	**	**	**

* Bit rates in the case of internal clock operation. In the case of external clock operation, the external clock is operatable up to DC ~ 1/2 system clock.

** The bit rate is shown as follows with the TCONR as N.

$$\text{Bit Rate } (\mu\text{s/bit}) = \frac{4\,(N+1)}{f} \qquad \left(\begin{array}{l} f: \ \text{input clock frequency to the} \\ \quad \text{timer 2 counter} \\ N = 0 \sim 255 \end{array}\right)$$

Table 7 Baud Rate and Time Constant Register Example

XTAL Baud Rate (Baud)	2 4576MHz	3 6864MHz	4 0MHz	4 9152MHz	8 0MHz
110	21*	32*	35*	43*	70*
150	127	191	207	255	51*
300	63	95	103	127	207
600	31	47	51	63	103
1200	15	23	25	31	51
2400	7	11	12	15	25
4800	3	5	—	7	12
9600	1	2	—	3	-
19200	0	—	—	1	—
38400	—	-	—	0	—

* E/8 clock is input to the timer 2 up counter and E clock otherwise.

Table 8 SCI Format and Clock Source Control

CC2	CC1	CC0	Format	Mode	Clock Source	Port 2, Bit 2	Port 2, Bit 3	Port 2, Bit 4
0	0	0	8-bit data	Clocked Synchronous	External	Input	When the TRCSR, RE bit is "1", bit 3 is used as a serial input.	When the TRCSR, TE bit is "1", bit 4 is used as a serial output.
0	0	1	8-bit data	Asynchronous	Internal	Not Used**		
0	1	0	8-bit data	Asynchronous	Internal	Output*		
0	1	1	8-bit data	Asynchronous	External	Input		
1	0	0	8-bit data	Clocked Synchronous	Internal	Output		
1	0	1	9-bit data	Asynchronous	Internal	Not Used**		
1	1	0	9-bit data	Asynchronous	Internal	Output*		
1	1	1	9-bit data	Asynchronous	External	Input		

* Clock output regardless of the TRCSR, bit RE and TE. ** Not used for the SCI.

Bit 6 TD8 Transmit Data Bit 8

When selecting 9-bit data format in the asynchronous mode, this bit is transmitted as the 9th data. In transmitting 9-bit data, write the 9th data into this bit then write data to the receive data register.

Bit 7 RD8 Receive Data Bit 8

When selecting 9-bit data format in the asynchronous mode, this bit stores the 9th bit data. In receiving 9-bit data, read this bit then the receive data register.

■ PRECAUTION 1

In the synchronous clocked receive operation with clock-output, there are three cases for clock pulse timing after RDRF clear as shown below.

Please consider above in designing system, since transmitting receiving time is not uniform.

The clock-output of case 1 or case 2 is determined by "1" or "0" of SCI internal operation clock of RDRF clearing cycle. In addition, in the case of low voltage operation ($V_{CC} < 4.5V$), the clock-output of case 1 may transfer to case 3.

■ PRECAUTION 2

When transmitting through clock-synchronous serial communication interface, TE bit should not be cleared with TDRE of TRCSR ($11) is "0".

The TDRE set and clear conditions of SCI are shown as follows.

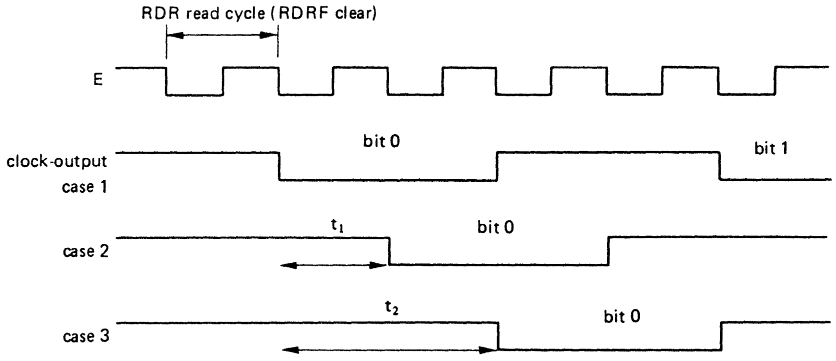
	Set condition	Clear condition
TDRE	1. TDR → transmit shift register (asynchronous) 2. Transmit shift register is empty. (clock-synchronous) 3. $\overline{RES}$ = 0	When writing to TDR after TRSCR read, with TDRE = 1, TDRE is cleared.

If transmit data is written to TDR, and then TE bit is cleared with TDRE = 0 to stop transmitting, TDRE remains "0".

In this case, even if TE bit is set and transmit data is written again, the TDR data is not transmitted.

Please note that TE bit must be cleared after the last data has been transmitted.

(This caution is not applied to asynchronous serial communication interface.)



(note) When bit rate is E/2, t_1 = E, and t_2 = 2E. E/128, t_1 = 64E, t_2 = 128E.
E/16, t_1 = 8E, t_2 = 16E. E/512, t_1 = 256E, t_2 = 512E.

Precaution 1 Diagram

■ TIMER, SCI STATUS FLAG

Table 9 shows the set and reset conditions of each status flag in the timer 1, timer 2 and SCI.

As for Timer 1 and Timer 2 status flag, if the set and reset condition occur simultaneously, the set condition is prior to the reset condition. But in case of SCI control status flag, the reset condition has priority. Especially as for OCF1 and OCF2 of Timer 1, the set signal is generated periodically whenever FRC matches OCR after the set, and which can cause the unclear of the flag. To clear surely, the method is necessary to avoid the occurence of the set signal between TCSR Read and OCR write. For example, match the OCR value to FRC first, and next read TCSR, and then write OCR at once.

Table 9 Timer 1, Timer 2 and SCI Status Flag

		Set Condition	Reset Condition
Timer 1	ICF	FRC → ICR by edge input to P_{20}.	1 Read the TCSR1 or TCSR2 then ICRH, when ICF = 1 2. $\overline{RES}$=0
	OCF1	OCR1=FRC	1. Read the TCSR1 or TCSR2 then write to the OCR1H or OCR1L, when OCF1 = 1 2. $\overline{RES}$=0
	OCF2	OCR2=FRC	1 Read the TCSR2 then write to the OCR2H or OCR2L, when OCF2 = 1 2. $\overline{RES}$=0
	TOF	FRC=$FFFF+1 cycle	1. Read the TCSR1 then FRCH, when TOF = 1 2. $\overline{RES}$=0
Timer 2	CMF	T2CNT=TCONR	1. Write "0" to CMF, when CMF = 1 2. $\overline{RES}$=0
SCI	RDRF	Receive Shift Register → RDR	1. Read the TRCSR then RDR, when RDRF = 1 2. $\overline{RES}$=0
	ORFE	1. Framing Error (Asynchronous Mode) Stop Bit = 0 2. Overrun Error (Asynchronous Mode) Receive Shift Register → RDR when RDRF=1	1. Read the TRCSR then RDR, when ORFE = 1 2. $\overline{RES}$=0
	TDRE	1. Asynchronous Mode TDR → Transmit Shift Register 2. Clocked Synchronous Mode Transmit Shift Register is "empty" 3. $\overline{RES}$=0	Read the TRCSR then write to the TDR, when TDRE = 1 (Note) TDRE should be reset after the TE set.

(Note) 1. →, transfer
2 For example, "ICRH" means High byte of ICR

■ LOW POWER DISSIPATION MODE

The HD6303X provides two low power dissipation modes, sleep and standby.

● Sleep Mode

The MPU goes to the sleep mode by SLP instruction execution. In the sleep mode, the CPU stops its operation, while the registers' contents are retained. In this mode, the peripherals except the CPU such as timers, SCI etc. continue their functions. The power dissipation of sleep-condition is one fifth that of operating condition.

The MPU returns from this mode by an interrupt, $\overline{RES}$ or $\overline{STBY}$, it goes to the reset state by $\overline{RES}$ and the standby mode by $\overline{STBY}$. When the CPU acknowledges an interrupt request, it cancels the sleep mode, returns to the operation mode and branches to the interrupt routine. When the CPU masks this interrupt, it cancels the sleep mode and executes the next instruction. However, for example if the timer 1 or 2 prohibits a timer interrupt, the CPU doesn't cancel the sleep mode because of no interrupt request.

This sleep mode is effective to reduce the power dissipation for a system with no need of the HD6303X's consecutive operation.

● Standby Mode

The HD6303X stops all the clocks and goes to the reset state with $\overline{STBY}$ "Low". In this mode, the power dissipation is reduced conspicuously. All pins except for the power supply, the $\overline{STBY}$ and XTAL are detached from the MPU internally and go to the high impedance state.

In this mode the power is supplied to the HD6303X, so the contents of RAM is retained. The MPU returns from this mode during reset. The followings are typical usage of this mode.

Save the CPU information and SP contents on RAM by $\overline{NMI}$. Then disable the RAME bit of the RAM control register and set the STBY PWR bit to go to the standby mode. If the STBY PWR bit is still set at reset start, that indicates the power is supplied to the MPU and RAM contents are retained properly. So system can restore itself by returning their pre-standby informations to the SP and the CPU. Fig. 22 depicts the timing at each pin with this example.

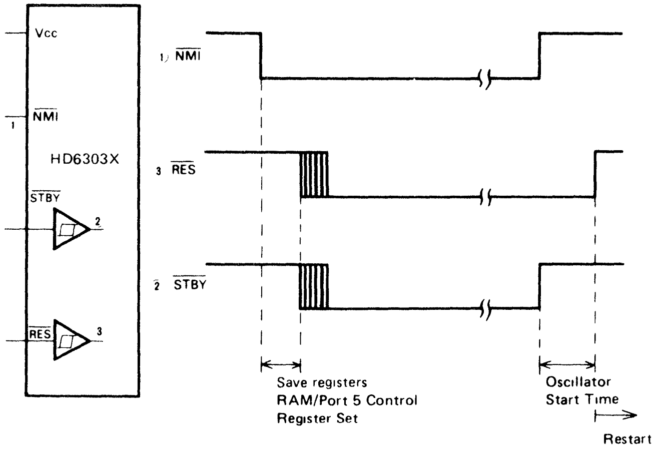


Figure 22 Standby Mode Timing

■ **TRAP FUNCTION**

The CPU generates an interrupt with the highest priority (TRAP) when fetching an undefined instruction or an instruction from non-memory space The TRAP prevents the system-burst caused by noise or a program error

● **Op Code Error**

When fetching an undefined op code, the CPU saves CPU registers as well as a normal interrupt and branches to the TRAP ($FFEE, $FFEF) This has the priority next to reset.

● **Address Error**

When an instruction fetch is made from internal register ($0000~$001F), the MPU generates an interrupt as well as an op code error But on the system with no memory in its external memory area, this function is not applicable if an instruction fetch is made from the external non-memory area.

This function is available only for an instruction fetch and is not applicable to the access of normal data read/write

(Note) The TRAP interrupt provides a retry function differently from other interrupts. This is a program flow return to the address where the TRAP occurs when a sequence returns to a main routine from the TRAP interrupt routine by RTI. The retry can prevent the system burst caused by noise etc.
However, if another TRAP occurs, the program repeats the TRAP interrupt forever, so the consideration is necessary in programming.

■ **INSTRUCTION SET**

The HD6303X provides object code upward compatible with the HD6801 to utilize all instruction' set of the HMCS6800. It also reduces the execution times of key instructions for throughput improvement.

Bit manipulation instruction, change instruction of the index register and accumulator and sleep instruction are also added.

The followings are explained here.

- CPU Programming Model (refer to Fig 23)
- Addressing Mode
- Accumulator and Memory Manipulation Instruction (refer to Table 10)
- New Instruction
- Index Register and Stack Manipulation Instruction (refer to Table 11)
- Jump and Branch Instruction (refer to Table 12)
- Condition Code Register Manipulation (refer to Table 13)
- Op Code Map (refer to Table 14)

● **Programming Model**

Fig. 23 depicts the HD6303X programming model The double accumulator D consists of accumulator A and B, so when using the accumulator D, the contents of A and B are destroyed.

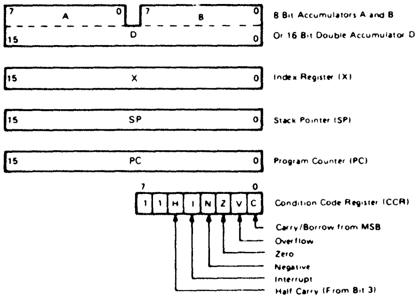


Figure 23 CPU Programming Model

● **CPU Addressing Mode**

The HD6303X provides 7 addressing modes The addressing mode is decided by an instruction type and code Table 10 through 14 show addressing modes of each instruction with the execution times counted by the machine cycle.

When the clock frequency is 4 MHz, the machine cycle time becomes microseconds directly.

Accumulator (ACCX) Addressing

Only an accumulator is addressed and the accumulator A or B is selected. This is a one-byte instruction.

Immediate Addressing

This addressing locates a data in the second byte of an instruction However, LDS and LDX locate a data in the second and third byte exceptionally This addressing is a 2 or 3-byte instruction

Direct Addressing

In this addressing mode, the second byte of an instruction shows the address where a data is stored 256 bytes ($0 through $255) can be addressed directly. Execution times can be reduced by storing data in this area so it is recommended to make it RAM for users' data area in configurating a system. This is a 2-byte instruction, while 3-byte with regard to AIM, OIM, EIM and TIM.

Extended Addressing

In this mode, the second byte shows the upper 8 bit of the data stored address and the third byte the lower 8 bit This indicates the absolute address of 3-byte instruction in the memory.

Indexed Addressing

The second byte of an instruction and the lower 8 bit of the index register are added in this mode. As for AIM, OIM, EIM and TIM, the third byte of an instruction and the lower 8 bits of the index register are added.

This carry is added to the upper 8 bit of the index register and the result is used for addressing the memory. The modified address is retained in the temporary address register, so the contents of the index register doesn't change. This is a 2-byte instruction except AIM, OIM, EIM and TIM (3-byte instruction).

Implied Addressing

An instruction itself specifies the address. That is, the instruction addresses a stack pointer, index register etc. This is a one-byte instruction.

Relative Addressing

The second byte of an instruction and the lower 8 bits of the program counter are added. The carry or borrow is added to the upper 8 bit. So addressing from −126 to +129 byte of the current instruction is enabled. This is a 2-byte instruction.

(Note) CLI, SEI Instructions and Interrupt Operation

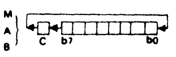
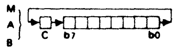
When accepting the IRQ at a preset timing with CLI and SEI instructions, more than 2 cycles are necessary between the CLI and SEI instructions. For example, the following program (a) (b) don't accept the IRQ but (c) accepts it.

```
   .       .       .
   .       .       .
   .       .       .
   .       .      CLI
  CLI     CLI     NOP
  SEI     NOP     NOP
   .      SEI     SEI
   .       .       .
   .       .       .
   .       .       .
   .       .       .

  (a)     (b)     (c)
```

The same thing can be said to the TAP instruction instead of the CLI and SEI instructions.

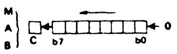
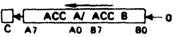
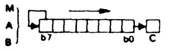
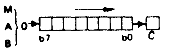
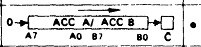
Table 10 Accumulator, Memory Manipulation Instructions

Operations	Mnemonic	Addressing Modes															Boolean/ Arithmetic Operation	Condition Code Register					
		IMMED			DIRECT			INDEX			EXTEND			IMPLIED				5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#		H	I	N	Z	V	C
Add	ADDA	8B	2	2	9B	3	2	AB	4	2	BB	4	3				A + M → A	↕	●	↕	↕	↕	↕
	ADDB	CB	2	2	DB	3	2	EB	4	2	FB	4	3				B + M → B	↕	●	↕	↕	↕	↕
Add Double	ADDD	C3	3	3	D3	4	2	E3	5	2	F3	5	3				A B + M M + 1 → A B	●	●	↕	↕	↕	↕
Add Accumulators	ABA													1B	1	1	A + B → A	↕	●	↕	↕	↕	↕
Add With Carry	ADCA	89	2	2	99	3	2	A9	4	2	B9	4	3				A + M + C → A	↕	●	↕	↕	↕	↕
	ADCB	C9	2	2	D9	3	2	E9	4	2	F9	4	3				B + M + C → B	↕	●	↕	↕	↕	↕
AND	ANDA	84	2	2	94	3	2	A4	4	2	B4	4	3				A·M → A	●	●	↕	↕	R	●
	ANDB	C4	2	2	D4	3	2	E4	4	2	F4	4	3				B·M → B	●	●	↕	↕	R	●
Bit Test	BIT A	85	2	2	95	3	2	A5	4	2	B5	4	3				A·M	●	●	↕	↕	R	●
	BIT B	C5	2	2	D5	3	2	E5	4	2	F5	4	3				B·M	●	●	↕	↕	R	●
Clear	CLR							6F	5	2	7F	5	3				00 → M	●	●	R	S	R	R
	CLRA													4F	1	1	00 → A	●	●	R	S	R	R
	CLRB													5F	1	1	00 → B	●	●	R	S	R	R
Compare	CMPA	81	2	2	91	3	2	A1	4	2	B1	4	3				A − M	●	●	↕	↕	↕	↕
	CMPB	C1	2	2	D1	3	2	E1	4	2	F1	4	3				B − M	●	●	↕	↕	↕	↕
Compare Accumulators	CBA													11	1	1	A − B	●	●	↕	↕	↕	↕
Complement, 1's	COM							63	6	2	73	6	3				$\overline{M}$ → M	●	●	↕	↕	R	S
	COMA													43	1	1	$\overline{A}$ → A	●	●	↕	↕	R	S
	COMB													53	1	1	$\overline{B}$ → B	●	●	↕	↕	R	S
Complement, 2's (Negate)	NEG							60	6	2	70	6	3				00 − M → M	●	●	↕	↕	①	②
	NEGA													40	1	1	00 − A → A	●	●	↕	↕	①	②
	NEGB													50	1	1	00 − B → B	●	●	↕	↕	①	②
Decimal Adjust, A	DAA													19	2	1	Converts binary add of BCD characters into BCD format	●	●	↕	↕	↕	③
Decrement	DEC							6A	6	2	7A	6	3				M − 1 → M	●	●	↕	↕	④	●
	DECA													4A	1	1	A − 1 → A	●	●	↕	↕	④	●
	DECB													5A	1	1	B − 1 → B	●	●	↕	↕	④	●
Exclusive OR	EORA	88	2	2	98	3	2	A8	4	2	88	4	3				A ⊕ M → A	●	●	↕	↕	R	●
	EORB	C8	2	2	D8	3	2	E8	4	2	F8	4	3				B ⊕ M → B	●	●	↕	↕	R	●
Increment	INC							6C	6	2	7C	6	3				M + 1 → M	●	●	↕	↕	⑤	●
	INCA													4C	1	1	A + 1 → A	●	●	↕	↕	⑤	●
	INCB													5C	1	1	B + 1 → B	●	●	↕	↕	⑤	●
Load Accumulator	LDAA	86	2	2	96	3	2	A6	4	2	B6	4	3				M → A	●	●	↕	↕	R	●
	LDAB	C6	2	2	D6	3	2	E6	4	2	F6	4	3				M → B	●	●	↕	↕	R	●
Load Double Accumulator	LDD	CC	3	3	DC	4	2	EC	5	2	FC	5	3				M + 1 → B, M → A	●	●	↕	↕	R	●
Multiply Unsigned	MUL													3D	7	1	A x B → A B	●	●	●	●	●	⑪
OR, Inclusive	ORAA	8A	2	2	9A	3	2	AA	4	2	BA	4	3				A + M → A	●	●	↕	↕	R	●
	ORAB	CA	2	2	DA	3	2	EA	4	2	FA	4	3				B + M → B	●	●	↕	↕	R	●
Push Data	PSHA													36	4	1	A → Msp, SP − 1 → SP	●	●	●	●	●	●
	PSHB													37	4	1	B → Msp, SP − 1 → SP	●	●	●	●	●	●
Pull Data	PULA													32	3	1	SP + 1 → SP, Msp → A	●	●	●	●	●	●
	PULB													33	3	1	SP + 1 → SP, Msp → B	●	●	●	●	●	●
Rotate Left	ROL							69	6	2	79	6	3					●	●	↕	↕	⑥	↕
	ROLA													49	1	1		●	●	↕	↕	⑥	↕
	ROLB													59	1	1		●	●	↕	↕	⑥	↕
Rotate Right	ROR							66	6	2	76	6	3					●	●	↕	↕	⑥	↕
	RORA													46	1	1		●	●	↕	↕	⑥	↕
	RORB													56	1	1		●	●	↕	↕	⑥	↕

(Note) Condition Code Register will be explained in Note of Table 13

(continued)

Table 10 Accumulator, Memory Manipulation Instructions

Operations	Mnemonic	Addressing Modes															Boolean/ Arithmetic Operation	Condition Code Register					
		IMMED			DIRECT			INDEX			EXTEND			IMPLIED				5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#		H	I	N	Z	V	C
Shift Left Arithmetic	ASL							68	6	2	78	6	3					●	●	↕	↕	⑥	↕
	ASLA													48	1	1		●	●	↕	↕	⑥	↕
	ASLB													58	1	1		●	●	↕	↕	⑥	↕
Double Shift Left, Arithmetic	ASLD													05	1	1		●	●	↕	↕	⑥	↕
Shift Right Arithmetic	ASR							67	6	2	77	6	3					●	●	↕	↕	⑥	↕
	ASRA													47	1	1		●	●	↕	↕	⑥	↕
	ASRB													57	1	1		●	●	↕	↕	⑥	↕
Shift Right Logical	LSR							64	6	2	74	6	3					●	●	R	↕	⑥	↕
	LSRA													44	1	1		●	●	R	↕	⑥	↕
	LSRB													54	1	1		●	●	R	↕	⑥	↕
Double Shift Right Logical	LSRD													04	1	1		●	●	R	↕	⑥	↕
Store Accumulator	STAA				97	3	2	A7	4	2	B7	4	3				A → M	●	●	↕	↕	R	●
	STAB				D7	3	2	E7	4	2	F7	4	3				B → M	●	●	↕	↕	R	●
Store Double Accumulator	STD				DD	4	2	ED	5	2	FD	5	3				A → M B → M + 1	●	●	↕	↕	R	●
Subtract	SUBA	80	2	2	90	3	2	A0	4	2	B0	4	3				A - M → A	●	●	↕	↕	↕	↕
	SUBB	C0	2	2	D0	3	2	E0	4	2	F0	4	3				B - M → B	●	●	↕	↕	↕	↕
Double Subtract	SUBD	83	3	3	93	4	2	A3	5	2	B3	5	3				A B - M M + 1 → A B	●	●	↕	↕	↕	↕
Subtract Accumulators	SBA													10	1	1	A - B → A	●	●	↕	↕	↕	↕
Subtract With Carry	SBCA	82	2	2	92	3	2	A2	4	2	B2	4	3				A - M - C → A	●	●	↕	↕	↕	↕
	SBCB	C2	2	2	D2	3	2	E2	4	2	F2	4	3				B - M - C → B	●	●	↕	↕	↕	↕
Transfer Accumulators	TAB													16	1	1	A → B	●	●	↕	↕	R	●
	TBA													17	1	1	B → A	●	●	↕	↕	R	●
Test Zero or Minus	TST							6D	4	2	7D	4	3				M - 00	●	●	↕	↕	R	R
	TSTA													4D	1	1	A - 00	●	●	↕	↕	R	R
	TSTB													5D	1	1	B - 00	●	●	↕	↕	R	R
And Immediate	AIM				71	6	3	61	7	3							M IMM→M	●	●	↕	↕	R	●
OR Immediate	OIM				72	6	3	62	7	3							M+IMM →M	●	●	↕	↕	R	●
EOR Immediate	EIM				75	6	3	65	7	3							M⊕IMM →M	●	●	↕	↕	R	●
Test Immediate	TIM				7B	4	3	6B	5	3							M IMM	●	●	↕	↕	R	●

(Note) Condition Code Register will be explained in Note of Table 13

2

HITACHI

- **Additional Instruction**

In addition to the HD6801 instruction set, the HD6303X prepares the following new instructions.

AIM . . . . (M)·(IMM) → (M)

Executes "AND" operation to immediate data and the memory contents and stores its result in the memory.

OIM . . . . (M) + (IMM) → (M)

Executes "OR" operation to immediate data and the memory contents and stores its result in the memory.

EIM . . . . (M) ⊕ (IMM) → (M)

Executes "EOR" operation to immediate data and the memory contents and stores its result in the memory.

TIM . . . . (M) · (IMM)

Executes "AND" operation to immediate data and changes the relative flag of the condition code register.

These area 3-byte instructions; the first byte is op code, the second immediate data and the third address modifier.

XGDX . . . . . (ACCD) ↔ (IX)

Exchanges the contents of accumulator and the index register.

SLP

Goes to the sleep mode. Refer to "LOW POWER DISSIPATION MODE" for more details of the sleep mode.

Table 11 Index Register, Stack Manipulation Instructions

Pointer Operations	Mnemonic	Addressing Modes															Boolean/ Arithmetic Operation	Condition Code Register					
		IMMED			DIRECT			INDEX			EXTEND			IMPLIED				5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#		H	I	N	Z	V	C
Compare Index Reg	CPX	8C	3	3	9C	4	2	AC	5	2	BC	5	3				X − M M + 1	●	●	:	↕	:	↕
Decrement Index Reg	DEX													09	1	1	X − 1 → X	●	●	●	↕	●	●
Decrement Stack Pntr	DES													34	1	1	SP − 1 → SP	●	●	●	●	●	●
Increment Index Reg	INX													08	1	1	X + 1 → X	●	●	●	↕	●	●
Increment Stack Pntr	INS													31	1	1	SP + 1 → SP	●	●	●	●	●	●
Load Index Reg	LDX	CE	3	3	DE	4	2	EE	5	2	FE	5	3				M → X_H, (M + 1) → X_L	●	●	⑦	↕	R	●
Load Stack Pntr	LDS	8E	3	3	9E	4	2	AE	5	2	BE	5	3				M → SP_H, (M+1) → SP_L	●	●	⑦	↕	R	●
Store Index Reg	STX				DF	4	2	EF	5	2	FF	5	3				X_H → M, X_L → (M + 1)	●	●	⑦	↕	R	●
Store Stack Pntr	STS				9F	4	2	AF	5	2	BF	5	3				SP_H → M, SP_L → (M + 1)	●	●	⑦	↕	R	●
Index Reg → Stack Pntr	TXS													35	1	1	X − 1 → SP	●	●	●	●	●	●
Stack Pntr → Index Reg	TSX													30	1	1	SP + 1 → X	●	●	●	●	●	●
Add	ABX													3A	1	1	B + X → X	●	●	●	●	●	●
Push Data	PSHX													3C	5	1	X_L → M_{sp}, SP − 1 → SP X_H → M_{sp}, SP − 1 → SP	●	●	●	●	●	●
Pull Data	PULX													38	4	1	SP + 1 → SP, M_{sp} → X_H SP + 1 → SP, M_{sp} → X_L	●	●	●	●	●	●
Exchange	XGDX													18	2	1	ACCD· ·IX	●	●	●	●	●	●

(Note) Condition Code Register will be explained in Note of Table 13

Table 12 Jump, Branch Instructions

Operations	Mnemonic	Addressing Modes															Branch Test	Condition Code Register					
		RELATIVE			DIRECT			INDEX			EXTEND			IMPLIED				5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#		H	I	N	Z	V	C
Branch Always	BRA	20	3	2													None	●	●	●	●	●	●
Branch Never	BRN	21	3	2													None	●	●	●	●	●	●
Branch If Carry Clear	BCC	24	3	2													C = 0	●	●	●	●	●	●
Branch If Carry Set	BCS	25	3	2													C = 1	●	●	●	●	●	●
Branch If = Zero	BEQ	27	3	2													Z = 1	●	●	●	●	●	●
Branch If ≥ Zero	BGE	2C	3	2													N ⊕ V = 0	●	●	●	●	●	●
Branch If > Zero	BGT	2E	3	2													Z + (N ⊕ V) = 0	●	●	●	●	●	●
Branch If Higher	BHI	22	3	2													C + Z = 0	●	●	●	●	●	●
Branch If ≤ Zero	BLE	2F	3	2													Z + (N ⊕ V) = 1	●	●	●	●	●	●
Branch If Lower Or Same	BLS	23	3	2													C + Z = 1	●	●	●	●	●	●
Branch If < Zero	BLT	2D	3	2													N ⊕ V = 1	●	●	●	●	●	●
Branch If Minus	BMI	2B	3	2													N = 1	●	●	●	●	●	●
Branch If Not Equal Zero	BNE	26	3	2													Z = 0	●	●	●	●	●	●
Branch If Overflow Clear	BVC	28	3	2													V = 0	●	●	●	●	●	●
Branch If Overflow Set	BVS	29	3	2													V = 1	●	●	●	●	●	●
Branch If Plus	BPL	2A	3	2													N = 0	●	●	●	●	●	●
Branch To Subroutine	BSR	8D	5	2														●	●	●	●	●	●
Jump	JMP							6E	3	2	7E	3	3					●	●	●	●	●	●
Jump To Subroutine	JSR				9D	5	2	AD	5	2	BD	6	3					●	●	●	●	●	●
No Operation	NOP													01	1	1	Advances Prog Cntr Only	●	●	●	●	●	●
Return From Interrupt	RTI													3B	10	1		⑧					
Return From Subroutine	RTS													39	5	1		●	●	●	●	●	●
Software Interrupt	SWI													3F	12	1		●	S	●	●	●	●
Wait for Interrupt*	WAI													3E	9	1		●	⑨	●	●	●	●
Sleep	SLP													1A	4	1		●	●	●	●	●	●

(Note) * WAI puts R/$\overline{W}$ high, Address Bus goes to FFFF, Data Bus goes to the three state
Condition Code Register will be explained in Note of Table 13

Table 13 Condition Code Register Manipulation Instructions

Operations	Mnemonic	AddressingModes			Boolean Operation	Condition Code Register					
		IMPLIED				5	4	3	2	1	0
		OP	~	#		H	I	N	Z	V	C
Clear Carry	CLC	0C	1	1	0 → C	●	●	●	●	●	R
Clear Interrupt Mask	CLI	0E	1	1	0 → I	●	R	●	●	●	●
Clear Overflow	CLV	0A	1	1	0 → V	●	●	●	●	R	●
Set Carry	SEC	0D	1	1	1 → C	●	●	●	●	●	S
Set Interrupt Mask	SEI	0F	1	1	1 → I	●	S	●	●	●	●
Set Overflow	SEV	0B	1	1	1 → V	●	●	●	●	S	●
Accumulator A → CCR	TAP	06	1	1	A → CCR	⑩					
CCR → Accumulator A	TPA	07	1	1	CCR → A	●	●	●	●	●	●

LEGEND

OP	Operation Code (Hexadecimal)
~	Number of MCU Cycles
M_{SP}	Contents of memory location pointed to by Stack Pointer
#	Number of Program Bytes
+	Arithmetic Plus
–	Arithmetic Minus
●	Boolean AND
+	Boolean Inclusive OR
⊕	Boolean Exclusive OR
$\overline{M}$	Complement of M
→	Transfer into
0	Bit = Zero
00	Byte = Zero

CONDITION CODE SYMBOLS

H	Half-carry from bit 3 to bit 4
I	Interrupt mask
N	Negative (sign bit)
Z	Zero (byte)
V	Overflow, 2's complement
C	Carry/Borrow from/to bit 7
R	Reset Always
S	Set Always
↕	Set if true after test or clear
●	Not Affected

(Note) Condition Code Register Notes (Bit set if test is true and cleared otherwise)

①	(Bit V)	Test Result = 10000000?
②	(Bit C)	Test Result ≠ 00000000?
③	(Bit C)	Test BCD Character of high-order byte greater than 10? (Not cleared if previously set)
④	(Bit V)	Test Operand = 10000000 prior to execution?
⑤	(Bit V)	Test Operand = 01111111 prior to execution?
⑥	(Bit V)	Test Set equal to N⊕ C = 1 after the execution of instructions
⑦	(Bit N)	Test Result less than zero? (Bit 15=1)
⑧	(All Bit)	Load Condition Code Register from Stack
⑨	(Bit I)	Set when interrupt occurs If previously set, a Non-Maskable Interrupt is required to exit the wait state
⑩	(All Bit)	Set according to the contents of Accumulator A
⑪	(Bit C)	Result of Multiplication Bit 7=1? (ACCB)

Table 14 OP-Code Map

OP CODE						ACC A	ACC B	IND	EXT	ACCA or SP				ACCB or X				
									DIR*	IMM	DIR	IND	EXT	IMM	DIR	IND	EXT	
HI		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111	
LO		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0000	0	▨	SBA	BRA	TSX	NEG				SUB								0
0001	1	NOP	CBA	BRN	INS	▨		AIM		CMP								1
0010	2	▨	▨	BHI	PULA	▨		OIM		SBC								2
0011	3	▨	▨	BLS	PULB	COM				SUBD				ADDD				3
0100	4	LSRD	▨	BCC	DES	LSR				AND								4
0101	5	ASLD	▨	BCS	TXS	▨		EIM		BIT								5
0110	6	TAP	TAB	BNE	PSHA	ROR				LDA								6
0111	7	TPA	TBA	BEQ	PSHB	ASR				▨	STA			▨	STA			7
1000	8	INX	XGDX	BVC	PULX	ASL				EOR								8
1001	9	DEX	DAA	BVS	RTS	ROL				ADC								9
1010	A	CLV	SLP	BPL	ABX	DEC				ORA								A
1011	B	SEV	ABA	BMI	RTI	▨		TIM		ADD								B
1100	C	CLC	▨	BGE	PSHX	INC				CPX				LDD				C
1101	D	SEC	▨	BLT	MUL	TST				BSR	JSR			▨	STD			D
1110	E	CLI	▨	BGT	WAI	▨	▨	JMP		LDS				LDX				E
1111	F	SEI	▨	BLE	SWI	CLR				▨	STS			▨	STX			F
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	

UNDEFINED OP CODE ▨

* Only each instructions of AIM, OIM, EIM, TIM

■ CPU OPERATION

● CPU Instruction Flow

When operating, the CPU fetches an instruction from a memory and executes the required function. This sequence starts with $\overline{\text{RES}}$ cancel and repeats itself limitlessly if not affected by a special instruction or a control signal. SWI, RTI, WAI and SLP instructions change this operation, while $\overline{\text{NMI}}$, $\overline{\text{IRQ}_1}$, $\overline{\text{IRQ}_2}$, IRQ_3, $\overline{\text{HALT}}$ and $\overline{\text{STBY}}$ control it. Fig. 24 gives the CPU mode transition and Fig. 25 the CPU system flow chart. Table 15 shows CPU operating states and port states.

● Operation at Each Instruction Cycle

Table 16 shows the operation at each instruction cycle. By the pipeline control of the HD6303X, MULT, PUL, DAA and XGDX instructions etc. prefetch the next instruction. So attention is necessary to the counting of the instruction cycles because it is different from the usual one ------ op code fetch to the next instruction op code.

Table 15 CPU Operation State and Port State

Port	Reset	STBY***	HALT	Sleep
$A_0 \sim A_7$	H	T	T	H
Port 2	T	T	Keep	Keep
$D_0 \sim D_7$	T	T	T	T
$A_8 \sim A_{15}$	H	T	T	H
Port 5	T	T	T	T
Port 6	T	T	Keep	Keep
Control Signal	*	T	**	*

H ; High, L , Low, T , High Impedance

* $\overline{\text{RD}}$, $\overline{\text{WR}}$, R/$\overline{\text{W}}$, $\overline{\text{LIR}}$ = H, BA = L

** $\overline{\text{RD}}$, $\overline{\text{WR}}$, R/$\overline{\text{W}}$ = T, $\overline{\text{LIR}}$, BA = H

*** E pin goes to high impedance state

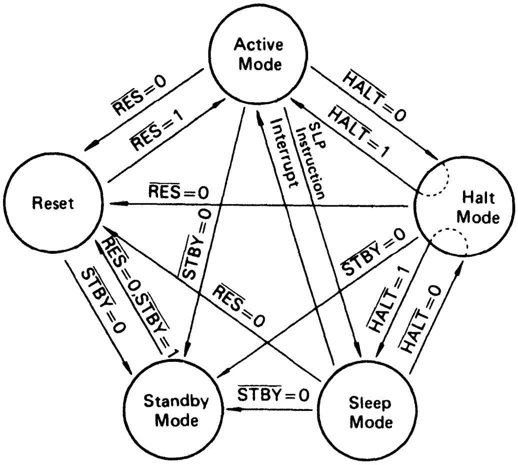
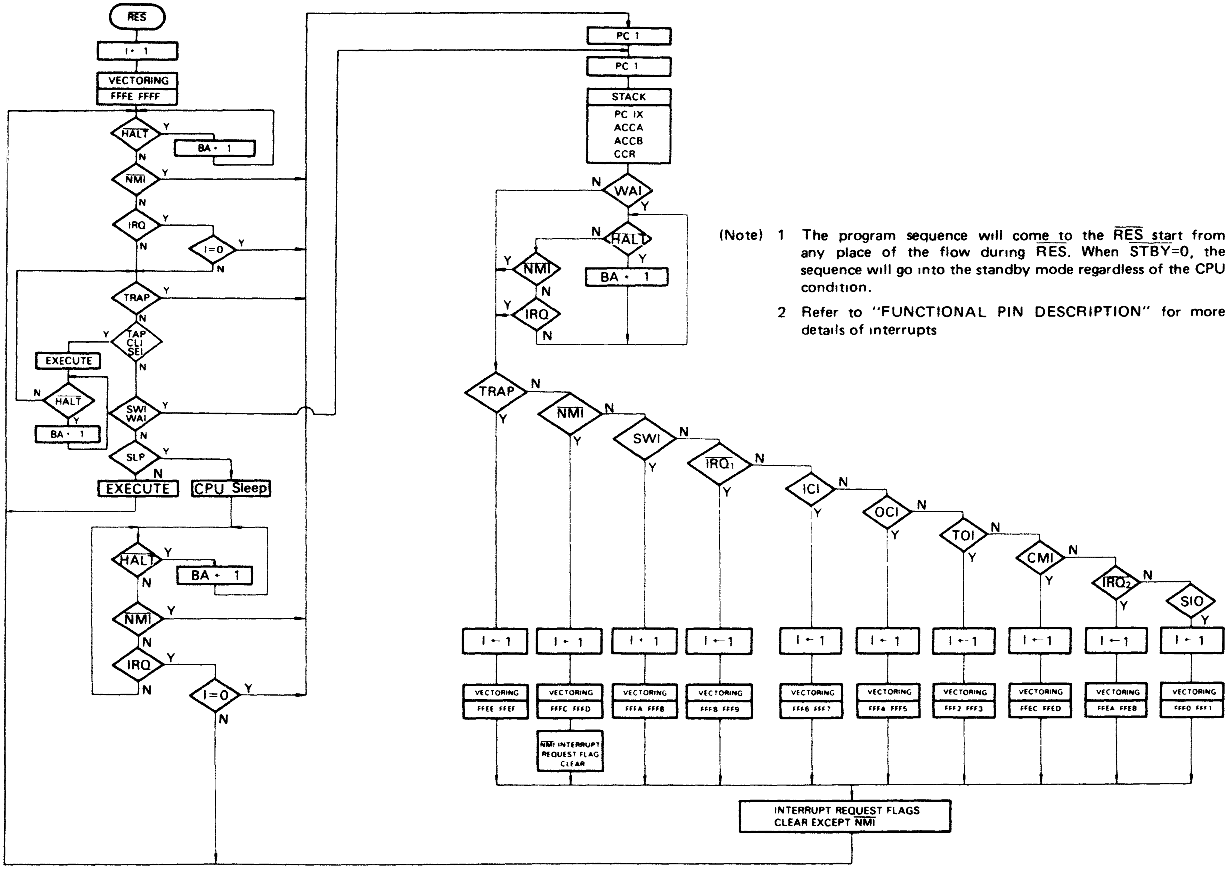


Figure 24 CPU Operation Mode Transition



(Note) 1 The program sequence will come to the RES start from any place of the flow during RES. When STBY=0, the sequence will go into the standby mode regardless of the CPU condition.

2 Refer to "FUNCTIONAL PIN DESCRIPTION" for more details of interrupts

Figure 25 HD6303X System Flow Chart

Table 16 Cycle-by-Cycle Operation

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/$\overline{W}$	$\overline{RD}$	$\overline{WR}$	$\overline{LIR}$	Data Bus
IMMEDIATE								
ADC ADD	2	1	Op Code Address + 1	1	0	1	1	Operand Data
AND BIT		2	Op Code Address + 2	1	0	1	0	Next Op Code
CMP EOR								
LDA ORA								
SBC SUB								
ADDD CPX	3	1	Op Code Address + 1	1	0	1	1	Operand Data (MSB)
LDD LDS		2	Op Code Address + 2	1	0	1	1	Operand Data (LSB)
LDX SUBD		3	Op Code Address + 3	1	0	1	0	Next Op Code
DIRECT								
ADC ADD	3	1	Op Code Address + 1	1	0	1	1	Address of Operand (LSB)
AND BIT		2	Address of Operand	1	0	1	1	Operand Data
CMP EOR		3	Op Code Address + 2	1	0	1	0	Next Op Code
LDA ORA								
SBC SUB								
STA	3	1	Op Code Address + 1	1	0	1	1	Destination Address
		2	Destination Address	0	1	0	1	Accumulator Data
		3	Op Code Address + 2	1	0	1	0	Next Op Code
ADDD CPX	4	1	Op Code Address + 1	1	0	1	1	Address of Operand (LSB)
LDD LDS		2	Address of Operand	1	0	1	1	Operand Data (MSB)
LDX SUBD		3	Address of Operand + 1	1	0	1	1	Operand Data (LSB)
		4	Op Code Address + 2	1	0	1	0	Next Op Code
STD STS	4	1	Op Code Address + 1	1	0	1	1	Destination Address (LSB)
STX		2	Destination Address	0	1	0	1	Register Data (MSB)
		3	Destination Address + 1	0	1	0	1	Register Data (LSB)
		4	Op Code Address + 2	1	0	1	0	Next Op Code
JSR	5	1	Op Code Address + 1	1	0	1	1	Jump Address (LSB)
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	Stack Pointer	0	1	0	1	Return Address (LSB)
		4	Stack Pointer − 1	0	1	0	1	Return Address (MSB)
		5	Jump Address	1	0	1	0	First Subroutine Op Code
TIM	4	1	Op Code Address + 1	1	0	1	1	Immediate Data
		2	Op Code Address + 2	1	0	1	1	Address of Operand (LSB)
		3	Address of Operand	1	0	1	1	Operand Data
		4	Op Code Address + 3	1	0	1	0	Next Op Code
AIM EIM	6	1	Op Code Address + 1	1	0	1	1	Immediate Data
OIM		2	Op Code Address + 2	1	0	1	1	Address of Operand (LSB)
		3	Address of Operand	1	0	1	1	Operand Data
		4	FFFF	1	1	1	1	Restart Address (LSB)
		5	Address of Operand	0	1	0	1	New Operand Data
		6	Op Code Address + 3	1	0	1	0	Next Op Code

(Continued)

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/$\overline{\text{W}}$	$\overline{\text{RD}}$	$\overline{\text{WR}}$	$\overline{\text{LIR}}$	Data Bus
INDEXED								
JMP	3	1	Op Code Address+1	1	0	1	1	Offset
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	Jump Address	1	0	1	0	First Op Code of Jump Routine
ADC ADD	4	1	Op Code Address+1	1	0	1	1	Offset
AND BIT		2	FFFF	1	1	1	1	Restart Address (LSB)
CMP EOR		3	IX+Offset	1	0	1	1	Operand Data
LDA ORA		4	Op Code Address+2	1	0	1	0	Next Op Code
SBC SUB								
TST								
STA	4	1	Op Code Address+1	1	0	1	1	Offset
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	IX+Offset	0	1	0	1	Accumulator Data
		4	Op Code Address+2	1	0	1	0	Next Op Code
ADDD	5	1	Op Code Address+1	1	0	1	1	Offset
CPX LDD		2	FFFF	1	1	1	1	Restart Address (LSB)
LDS LDX		3	IX+Offset	1	0	1	1	Operand Data (MSB)
SUBD		4	IX+Offset+1	1	0	1	1	Operand Data (LSB)
		5	Op Code Address+2	1	0	1	0	Next Op Code
STD STS	5	1	Op Code Address+1	1	0	1	1	Offset
STX		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	IX+Offset	0	1	0	1	Register Data (MSB)
		4	IX+Offset+1	0	1	0	1	Register Data (LSB)
		5	Op Code Address+2	1	0	1	0	Next Op Code
JSR	5	1	Op Code Address+1	1	0	1	1	Offset
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	Stack Pointer	0	1	0	1	Return Address (LSB)
		4	Stack Pointer − 1	0	1	0	1	Return Address (MSB)
		5	IX+Offset	1	0	1	0	First Subroutine Op Code
ASL ASR	6	1	Op Code Address+1	1	0	1	1	Offset
COM DEC		2	FFFF	1	1	1	1	Restart Address (LSB)
INC LSR		3	IX+Offset	1	0	1	1	Operand Data
NEG ROL		4	FFFF	1	1	1	1	Restart Address (LSB)
ROR		5	IX+Offset	0	1	0	1	New Operand Data
		6	Op Code Address+2	1	0	1	0	Next Op Code
TIM	5	1	Op Code Address+1	1	0	1	1	Immediate Data
		2	Op Code Address+2	1	0	1	1	Offset
		3	FFFF	1	1	1	1	Restart Address (LSB)
		4	IX+Offset	1	0	1	1	Operand Data
		5	Op Code Address+3	1	0	1	0	Next Op Code
CLR	5	1	Op Code Address+1	1	0	1	1	Offset
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	IX+Offset	1	0	1	1	Operand Data
		4	IX+Offset	0	1	0	1	00
		5	Op Code Address+2	1	0	1	0	Next Op Code
AIM EIM	7	1	Op Code Address+1	1	0	1	1	Immediate Data
OIM		2	Op Code Address+2	1	0	1	1	Offset
		3	FFFF	1	1	1	1	Restart Address (LSB)
		4	IX+Offset	1	0	1	1	Operand Data
		5	FFFF	1	1	1	1	Restart Address (LSB)
		6	IX+Offset	0	1	0	1	New Operand Data
		7	Op Code Address+3	1	0	1	0	Next Op Code

(Continued)

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/$\overline{\text{W}}$	$\overline{\text{RD}}$	$\overline{\text{WR}}$	$\overline{\text{LIR}}$	Data Bus
EXTEND								
JMP	3	1	Op Code Address + 1	1	0	1	1	Jump Address (MSB)
		2	Op Code Address + 2	1	0	1	1	Jump Address (LSB)
		3	Jump Address	1	0	1	0	Next Op Code
ADC ADD TST	4	1	Op Code Address + 1	1	0	1	1	Address of Operand (MSB)
AND BIT		2	Op Code Address + 2	1	0	1	1	Address of Operand (LSB)
CMP EOR		3	Address of Operand	1	0	1	1	Operand Data
LDA ORA		4	Op Code Address + 3	1	0	1	0	Next Op Code
SBC SUB								
STA	4	1	Op Code Address + 1	1	0	1	1	Destination Address (MSB)
		2	Op Code Address + 2	1	0	1	1	Destination Address (LSB)
		3	Destination Address	0	1	0	1	Accumulator Data
		4	Op Code Address + 3	1	0	1	0	Next Op Code
ADDD	5	1	Op Code Address + 1	1	0	1	1	Address of Operand (MSB)
CPX LDD		2	Op Code Address + 2	1	0	1	1	Address of Operand (LSB)
LDS LDX		3	Address of Operand	1	0	1	1	Operand Data (MSB)
SUBD		4	Address of Operand + 1	1	0	1	1	Operand Data (LSB)
		5	Op Code Address + 3	1	0	1	0	Next Op Code
STD STS	5	1	Op Code Address + 1	1	0	1	1	Destination Address (MSB)
STX		2	Op Code Address + 2	1	0	1	1	Destination Address (LSB)
		3	Destination Address	0	1	0	1	Register Data (MSB)
		4	Destination Address + 1	0	1	0	1	Register Data (LSB)
		5	Op Code Address + 3	1	0	1	0	Next Op Code
JSR	6	1	Op Code Address + 1	1	0	1	1	Jump Address (MSB)
		2	Op Code Address + 2	1	0	1	1	Jump Address (LSB)
		3	FFFF	1	1	1	1	Restart Address (LSB)
		4	Stack Pointer	0	1	0	1	Return Address (LSB)
		5	Stack Pointer − 1	0	1	0	1	Return Address (MSB)
		6	Jump Address	1	0	1	0	First Subroutine Op Code
ASL ASR	6	1	Op Code Address + 1	1	0	1	1	Address of Operand (MSB)
COM DEC		2	Op Code Address + 2	1	0	1	1	Address of Operand (LSB)
INC LSR		3	Address of Operand	1	0	1	1	Operand Data
NEG ROL		4	FFFF	1	1	1	1	Restart Address (LSB)
ROR		5	Address of Operand	0	1	0	1	New Operand Data
		6	Op Code Address + 3	1	0	1	0	Next Op Code
CLR	5	1	Op Code Address + 1	1	0	1	1	Address of Operand (MSB)
		2	Op Code Address + 2	1	0	1	1	Address of Operand (LSB)
		3	Address of Operand	1	0	1	1	Operand Data
		4	Address of Operand	0	1	0	1	00
		5	Op Code Address + 3	1	0	1	0	Next Op Code

(Continued)

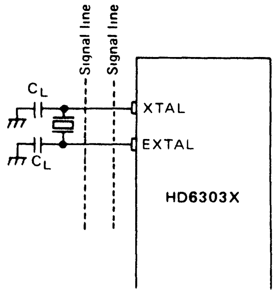
Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/$\overline{W}$	$\overline{RD}$	$\overline{WR}$	$\overline{LIR}$	Data Bus
IMPLIED								
ABA ABX	1	1	Op Code Address + 1	1	0	1	0	Next Op Code
ASL ASLD								
ASR CBA								
CLC CLI								
CLR CLV								
COM DEC								
DES DEX								
INC INS								
INX LSR								
LSRD ROL								
ROR NOP								
SBA SEC								
SEI SEV								
TAB TAP								
TBA TPA								
TST TSX								
TXS								
DAA XGDX	2	1	Op Code Address + 1	1	0	1	0	Next Op Code
		2	FFFF	1	1	1	1	Restart Address (LSB)
PULA PULB	3	1	Op Code Address + 1	1	0	1	0	Next Op Code
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	Stack Pointer + 1	1	0	1	1	Data from Stack
PSHA PSHB	4	1	Op Code Address + 1	1	0	1	1	Next Op Code
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	Stack Pointer	0	1	0	1	Accumulator Data
		4	Op Code Address + 1	1	0	1	0	Next Op Code
PULX	4	1	Op Code Address + 1	1	0	1	0	Next Op Code
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	Stack Pointer + 1	1	0	1	1	Data from Stack (MSB)
		4	Stack Pointer + 2	1	0	1	1	Data from Stack (LSB)
PSHX	5	1	Op Code Address + 1	1	0	1	1	Next Op Code
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	Stack Pointer	0	1	0	1	Index Register (LSB)
		4	Stack Pointer − 1	0	1	0	1	Index Register (MSB)
		5	Op Code Address + 1	1	0	1	0	Next Op Code
RTS	5	1	Op Code Address + 1	1	0	1	1	Next Op Code
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	Stack Pointer + 1	1	0	1	1	Return Address (MSB)
		4	Stack Pointer + 2	1	0	1	1	Return Address (LSB)
		5	Return Address	1	0	1	0	First Op Code of Return Routine
MUL	7	1	Op Code Address + 1	1	0	1	0	Next Op Code
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	FFFF	1	1	1	1	Restart Address (LSB)
		4	FFFF	1	1	1	1	Restart Address (LSB)
		5	FFFF	1	1	1	1	Restart Address (LSB)
		6	FFFF	1	1	1	1	Restart Address (LSB)
		7	FFFF	1	1	1	1	Restart Address (LSB)

(Continued)

Address Mode & Instructions	Cycles	Cycle #	Address Bus	$R/\overline{W}$	$\overline{RD}$	$\overline{WR}$	$\overline{LIR}$	Data Bus
IMPLIED								
WAI	9	1	Op Code Address + 1	1	0	1	1	Next Op Code
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	Stack Pointer	0	1	0	1	Return Address (LSB)
		4	Stack Pointer − 1	0	1	0	1	Return Address (MSB)
		5	Stack Pointer − 2	0	1	0	1	Index Register (LSB)
		6	Stack Pointer − 3	0	1	0	1	Index Register (MSB)
		7	Stack Pointer − 4	0	1	0	1	Accumulator A
		8	Stack Pointer − 5	0	1	0	1	Accumulator B
		9	Stack Pointer − 6	0	1	0	1	Conditional Code Register
RTI	10	1	Op Code Address + 1	1	0	1	1	Next Op Code
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	Stack Pointer + 1	1	0	1	1	Conditional Code Register
		4	Stack Pointer + 2	1	0	1	1	Accumulator B
		5	Stack Pointer + 3	1	0	1	1	Accumulator A
		6	Stack Pointer + 4	1	0	1	1	Index Register (MSB)
		7	Stack Pointer + 5	1	0	1	1	Index Register (LSB)
		8	Stack Pointer + 6	1	0	1	1	Return Address (MSB)
		9	Stack Pointer + 7	1	0	1	1	Return Address (LSB)
		10	Return Address	1	0	1	0	First Op Code of Return Routine
SWI	12	1	Op Code Address + 1	1	0	1	1	Next Op Code
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	Stack Pointer	0	1	0	1	Return Address (LSB)
		4	Stack Pointer − 1	0	1	0	1	Return Address (MSB)
		5	Stack Pointer − 2	0	1	0	1	Index Register (LSB)
		6	Stack Pointer − 3	0	1	0	1	Index Register (MSB)
		7	Stack Pointer − 4	0	1	0	1	Accumulator A
		8	Stack Pointer − 5	0	1	0	1	Accumulator B
		9	Stack Pointer − 6	0	1	0	1	Conditional Code Register
		10	Vector Address FFFA	1	0	1	1	Address of SWI Routine (MSB)
		11	Vector Address FFFB	1	0	1	1	Address of SWI Routine (LSB)
		12	Address of SWI Routine	1	0	1	0	First Op Code of SWI Routine
SLP	4	1	Op Code Address + 1	1	0	1	1	Next Op Code
		2	FFFF	1	1	1	1	Restart Address (LSB)
		↑ Sleep ↓		↓	↓	↓	↓	↓
		3	FFFF	1	1	1	1	Restart Address (LSB)
		4	Op Code Address + 1	1	0	1	0	Next Op Code
RELATIVE								
BCC BCS	3	1	Op Code Address + 1	1	0	1	1	Branch Offset
BEQ BGE		2	FFFF	1	1	1	1	Restart Address (LSB)
BGT BHI BLE BLS		3	Branch Address Test = 1 Op Code Address + 1 Test = ¨0	1	0	1	0	First Op Code of Branch Routine Next Op Code
BLT BMT								
BNE BPL								
BRA BRN								
BVC BVS								
BSR	5	1	Op Code Address + 1	1	0	1	1	Offset
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	Stack Pointer	0	1	0	1	Return Address (LSB)
		4	Stack Pointer − 1	0	1	0	1	Return Address (MSB)
		5	Branch Address	1	0	1	0	First Op Code of Subroutine

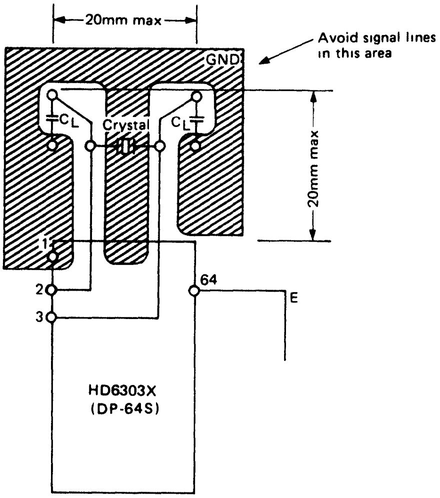
■ WARNING CONCERNING THE BOARD DESIGN OF OSCILLATION CIRCUIT

When designing a board, note that crosstalk may disturb the normal oscillation if signal lines are placed near the oscillation circuit as shown in Figure 26. Place the crystal and C_L as close to the HD6303X as possible.



Do not use this kind of printed-circuit board design.

Figure 26 Warning concerning board design of oscillation circuit



(Top View)

Figure 27 Example of Oscillation Circuits in Board Design

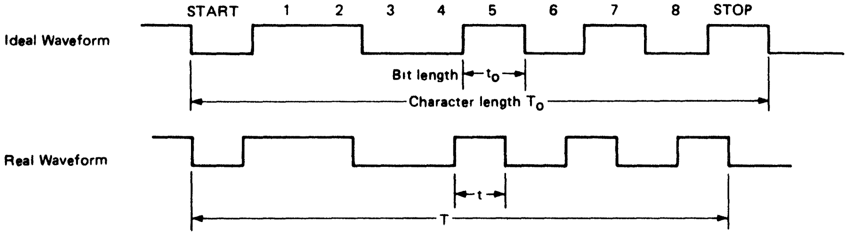
■ RECEIVE MARGIN OF THE SCI

Receive margin of the SCI contained in the HD6303X is shown in Table 17.

Note: SCI = Serial Communication Interface

Table 17

	Bit distortion tolerance (t−to) /to	Character distortion tolerance (T−To) /To
HD6303X	±43.7%	±4.37%



■ WARNING CONCERNING WAI INSTRUCTION

If the HALT signal is accepted by the MCU while the WAI instruction is executing, the CPU will not operate correctly after HALT mode is canceled.

WAI is a instruction which waits for an interrupt. The corresponding interrupt routine is executed after an interrupt occurs.

However, during the execution of the WAI instruction, $\overline{\text{HALT}}$ input makes the CPU malfunction and fetch an abnormal interrupt vectoring address.

In HALT mode, the CPU operates correctly without the WAI instruction, and WAI is executed correctly without $\overline{\text{HALT}}$ input. Therefore, if $\overline{\text{HALT}}$ input is necessary, make interrupts wait during the loop routine, as shown in Figure 28.

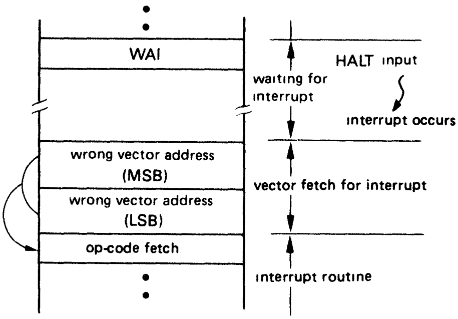
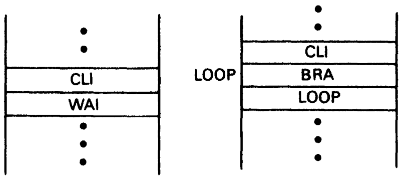


Figure 28 MAC function during WAI



i) MAL function ii) Recommended method

Figure 29 Program to wait for interrupt

■ WRITE-ONLY REGISTER

When the CPU reads a write-only register, the read data is always $FF, regardless of the value in the write-only register. Therefore, be careful of the results of instructions which read write-only register and perform an arithmetic or logical operation on its contents, such as AIM, ADD, or ROL, is executed, because the arithmetic or logical operation is always done with the data $FF. In particulars, don't use the AIM, OIM or EIM instruction to manipulate the DDR bit of PORT.

■ WARNING CONCERNING POWER START-UP

$\overline{\text{RES}}$ must be held low for at least 20 ms when the power starts up. In this case, the internal reset function is not effective until the oscillation begins at power-on. The $\overline{\text{RES}}$ signal is input to the LSI in synchronism with the internal clock ϕ (shown in Figure 30.)

Therefore, after power starts up, the LSI conditions such as its I/O ports and operating mode, are unstable. Fix the level of I/O ports by means of an external circuit to determine the level for system operation during the oscillator stabilization time.

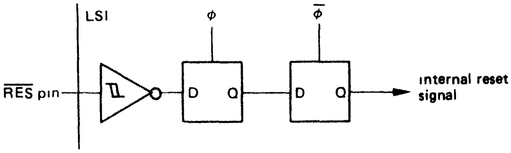


Figure 30 $\overline{\text{RES}}$ circuit

2

HD6303Y,HD63A03Y, HD63B03Y,HD63C03Y CMOS MPU (Micro Processing Unit)

The HD6303Y is a CMOS 8-bit single-chip microprocessing unit which contains a CPU compatible with the CMOS 8-bit microcomputer HD6301V, 256 bytes of RAM, 24 parallel I/O pins, Serial Communication Interface (SCI) and two timers

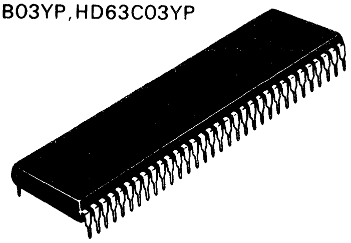
■ FEATURES

- Instruction Set Compatible with the HD6301V1
- 256 Bytes of RAM
- 24 Parallel I/O Pins
- Parallel Handshake Interface (Port 6)
- Darlington Transistor Drive (Port 2, 6)
- 16-Bit Programmable Timer
  Input Capture Register × 1
  Free Running Counter × 1
  Output Compare Register × 2
- 8-Bit Reloadable Timer
  External Event Counter
  Square Wave Generation
- Serial Communication Interface (SCI)
  Asynchronous Mode (8 Transmit Formats, Hardware Parity)
  Clocked Synchronous Mode
- Memory Ready
  3 Kinds of Memory Ready
- Halt
- Error Detection
  (Address Error, Op-code Error)
- Interrupt — External 3, Internal 7
- Maximum 65k Bytes Address Space
- Low Power Dissipation Mode
  Sleep Mode
  Standby Mode (Hardware Standby, Software Standby)
- Minimum Instruction Execution Time — 0 5μs (f = 2MHz)
- Wide Range of Operation
  V_{CC}= 3 to 5.5V (f=0.1 to 0.5MHz)
  V_{CC}=5V±10% { f=0.1 to 1.0MHz : HD6303Y
  f=0.1 to 1.5MHz . HD63A03Y
  f=0.1 to 2.0MHz HD63B03Y
  f=0.1 to 3.0MHz ; HD63C03Y }

■ PROGRAM DEVELOPMENT SUPPORT TOOLS

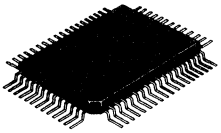
- Cross assembler and C compiler software for IBM PCs and compatibles
- In circuit emulator for use with IBM PCs and compatibles

HD6303YP,HD63A03YP,
HD63B03YP,HD63C03YP



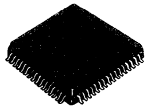
(DP-64S)

HD6303YF,HD63A03YF,
HD63B03YF,HD63C03YF



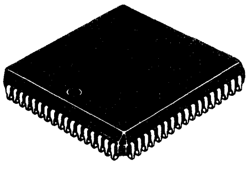
(FP-64)

HD6303YH,HD63A03YH
HD63B03YH,HD63C03YH



(FP-64A)

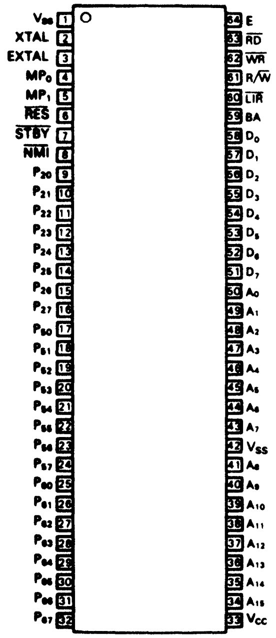
HD6303YCP,HD63A03YCP
HD63B03YCP,HD63C03YCP



(CP-68)

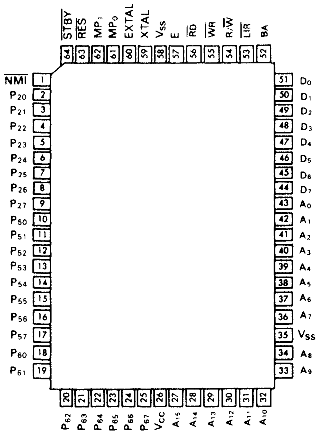
- **PIN ARRANGEMENT**

- HD6303YP, HD63A03YP, HD63B03YP, HD63C03YP



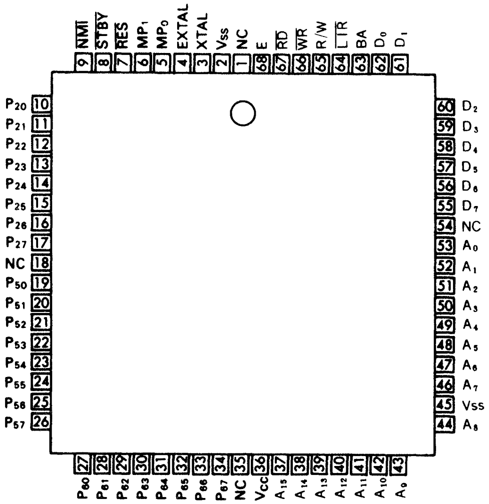
(Top View)

- HD6303YF, HD63A03YF, HD63B03YF, HD63C03YF



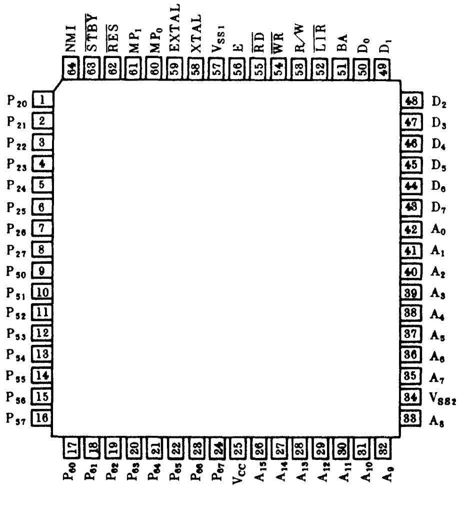
(Top View)

- HD6303YCP, HD63A03YCP, HD63B03YCP, HD63C03YCP



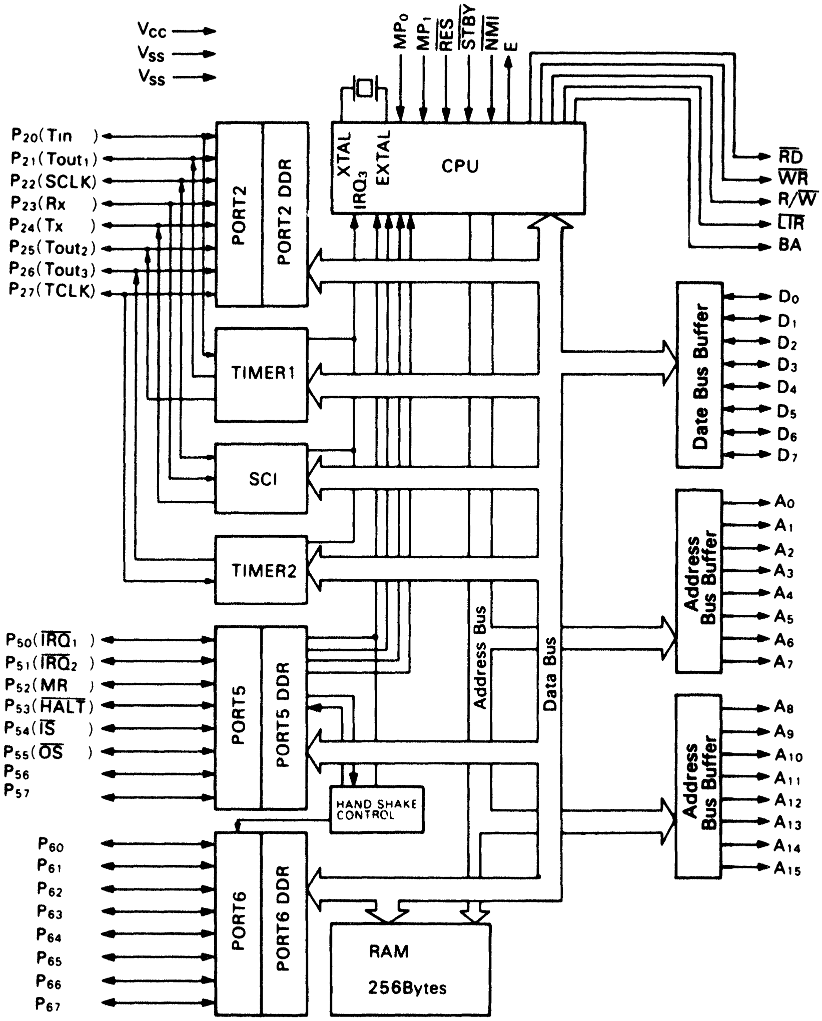
(Top View)

- HD6303YH, HD63A03YH, HD63B03YH, HD63C03YH



(Top View)

■ BLOCK DIAGRAM



■ ABSOLUTE MAXIMUM RATINGS

Item	Symbol	Value	Unit
Supply Voltage	V_{CC}	−0.3∼+7.0	V
Input Voltage	V_{in}	−0.3∼V_{CC}+0.3	V
Operating Temperature	T_{opr}	-20∼+75	°C
Storage Temperature	T_{stg}	−55∼+150	°C

(NOTE) This product has protection circuits in input terminal from high static electricity voltage and high electric field
But be careful not to apply overvoltage more than maximum ratings to these high input impedance protection circuits To assure the normal operation, we recommend V_{in}, V_{out}: $V_{SS} \leq (V_{in}$ or $V_{out}) \leq V_{CC}$.

■ ELECTRICAL CHARACTERISTICS

● DC CHARACTERISTICS (V_{CC} = 5.0V ± 10%, V_{SS} = 0V, T_a = -20°C ∼ +75°C, unless otherwise noted.)

Item		Symbol	Test Condition	min	typ	max	Unit
Input "High" Voltage	$\overline{RES}$, $\overline{STBY}$	V_{IH}		V_{CC}−0.5	−	V_{CC} +0.3	V
	EXTAL			V_{CC}×0.7	−		
	Other Inputs			2.0	−		
Input "Low" Voltage	All Inputs	V_{IL}		−0.3	−	0.8***	V
Input Leakage Current	$\overline{NMI}$, $\overline{RES}$, $\overline{STBY}$, MP_0, MP_1	$\lvert I_{in} \rvert$	V_{in} = 0 5∼V_{CC}−0.5V	−	−	1.0	μA
Three State Leakage Current	A_0∼A_{15}, D_0∼D_7, $\overline{RD}$, $\overline{WR}$, R/$\overline{W}$, Ports 2, 5, 6	$\lvert I_{TSI} \rvert$	V_{in} = 0.5∼V_{CC}−0.5V	−	−	1.0	μA
Output "High" Voltage	All Outputs	V_{OH}	I_{OH} = −200μA	2.4	−	−	V
			I_{OH} = −10μA	V_{CC}−0.7	−	−	V
Output "Low" Voltage	All Outputs	V_{OL}	I_{OL} = 1.6mA	−	−	0.4	V
Darlington Drive Current	Ports 2, 6	$-I_{OH}$	V_{out} = 1.5V	1.0	−	10.0	mA
Input Capacitance	All Inputs	C_{in}	V_{in} = 0V, f = 1MHz, Ta = 25°C	−	−	12.5	pF
Standby Current	Non Operation	I_{STB}		−	3.0	15.0	μA
Current Dissipation*		I_{SLP}	Sleeping (f=1MHz**)	−	1.5	3.0	mA
			Sleeping (f=1.5MHz**)	−	2.3	4.5	mA
			Sleeping (f=2MHz**)**	−	3.0	6.0	mA
			Sleeping (f=3 MHz)	−	4.5	9.0	mA
		I_{CC}	Operating (f=1MHz**)	−	7.0	10.0	mA
			Operating (f=1.5MHz**)	−	10.5	15.0	mA
			Operating (f=2MHz**)**	−	14.0	20.0	mA
			Operating (f=3 MHz)	−	21.0	30.0	mA
RAM Standby Voltage		V_{RAM}		2.0	−	−	V

* V_{IH} min = V_{CC} − 1.0V, V_{IL} max = 0 8V (All output terminals are at no load)

** Current Dissipation of the operating or sleeping condition is proportional to the operating frequency So the typ or max values about Current Dissipations at X MHz operation are decided according to the following formula
typ. value (f = X MHz) = typ. value (f = 1MHz) × X
max value (f = X MHz) = max. value (f = 1MHz) × X
(both the sleeping and operating)

*** SCLK 0.6V (-20°C∼0°C)

● AC CHARACTERISTICS (V_{CC} = 5.0V ± 10%, V_{SS} = 0V, T_a = -20 ~ +75°C, unless otherwise noted.)

BUS TIMING

Item		Symbol	Test Condition	HD6301Y0			HD63A01Y0			HD63B01Y0			HD63C01Y0			Unit
				min.	typ	max	min	typ	max	min	typ	max	min	typ	max	
Cycle Time		t_{cyc}		1	—	10	0.666	—	10	0 5	—	10	0 333	—	10	μs
Enable Rise Time		t_{Er}	Fig 1	—	—	25	—	—	25	—	—	25	—	—	20	ns
Enable Fall Time		t_{Ef}		—	—	25	—	—	25	—	—	25	—	—	20	ns
Enable Pulse Width "High" Level*		PW_{EH}		450	—	—	300	—	—	220	—	—	140	—	—	ns
Enable Pulse Width "Low" Level*		PW_{EL}		450	—	—	300	—	—	220	—	—	140	—	—	ns
Address, R/$\overline{W}$ Delay Time*		t_{AD}		—	—	250	—	—	190	—	—	160	—	—	120	ns
Data Delay Time	Write	t_{DDW}		—	—	200	—	—	160	—	—	120	—	—	100	ns
Data Set-up Time	Read	t_{DSR}		80	—	—	70	—	—	60	—	—	50	—	—	ns
Address, R/$\overline{W}$ Hold Time*		t_{AH1}		80	—	—	50	—	—	40	—	—	20	—	—	ns
Data Hold Time	Write*	t_{HW1}		80	—	—	50	—	—	40	—	—	20	—	—	ns
$\overline{RD}$, $\overline{WR}$ Address Hold Time*		t_{AH2}		70	—	—	50	—	—	40	—	—	20	—	—	ns
$\overline{RD}$, $\overline{WR}$ Data Hold Time*		t_{HW2}		70	—	—	50	—	—	40	—	—	20	—	—	ns
Data Hold Time	Read	t_{HR}		0	—	—	0	—	—	0	—	—	0	—	—	ns
$\overline{RD}$, $\overline{WR}$ Pulse Width*		PW_{RW}		450	—	—	300	—	—	220	—	—	140	—	—	ns
$\overline{RD}$, $\overline{WR}$ Delay Time		t_{RWD}		—	—	40	—	—	40	—	—	40	—	—	40	ns
$\overline{RD}$, $\overline{WR}$ Hold Time		t_{HRW}		—	—	20	—	—	20	—	—	20	—	—	20	ns
$\overline{LIR}$ Delay Time		t_{DLR}		—	—	200	—	—	160	—	—	120	—	—	80	ns
$\overline{LIR}$ Hold Time		t_{HLR}		10	—	—	10	—	—	10	—	—	5	—	—	ns
Peripheral Read Access Time		t_{ACC}		—	—	—	—	—	—	—	—	—	180	—	—	ns
MR Set-up Time*		t_{SLR}	Fig 2	400	—	—	280	—	—	230	—	—	170	—	—	ns
MR Hold Time*		t_{HMR}		—	—	100	—	—	70	—	—	50	—	—	25	ns
E Clock Pulse Width at MR		PW_{EMR}		—	—	9	—	—	9	—	—	9	—	—	9	μs
Processor Control Set-up Time		t_{PCS}	Fig 3, 13, 14	200	—	—	200	—	—	200	—	—	100	—	—	ns
Processor Control Rise Time		t_{PCr}	Fig. 2, 3	—	—	100	—	—	100	—	—	100	—	—	50	ns
Processor Control Fall Time		t_{PCf}		—	—	100	—	—	100	—	—	100	—	—	50	ns
BA Delay Time		t_{BA}	Fig. 3	—	—	250	—	—	190	—	—	160	—	—	120	ns
Oscillator Stabilization Time		t_{RC}	Fig. 14	20	—	—	20	—	—	20	—	—	20	—	—	ms
Reset Pulse Width		PW_{RST}		3	—	—	3	—	—	3	—	—	3	—	—	t_{cyc}

*These timings change in approximate proportion to t_{cyc} The figures in this characteristics represent those when t_{cyc} is minimum (= in the highest speed operation)

Peripheral Port Timing

Item		Symbol	Test condition	HD6303Y			HD63A03Y			HD63B03Y			HD63C03Y			Unit
				min	typ	max	min	typ	max	min	typ	max	min	typ	max	
Peripheral Data Set Up Time	Port 2,5,6	t_{PDSU}	Fig. 5	200	–	–	200	–	–	200	–	–	200	–	–	ns
Peripheral Data Hold Time	Port 2,5,6	t_{PDH}		200	–	–	200	–	–	200	–	–	200	–	–	ns
Delay Time (From Enable Fall Edge to Peripheral Output)	Port 2,5,6	t_{PWD}	Fig. 6	–	–	300	–	–	300	–	–	300	–	–	300	ns
Input Strobe Pulse Width		t_{PWIS}	Fig.10	200	–	–	200	–	–	200	–	–	200	–	–	ns
Input Data Hold Time	Port 6	t_{IH}		150	–	–	150	–	–	150	–	–	150	–	–	ns
Input Data Set-Up Time	Port 6	t_{IS}		100	–	–	100	–	–	100	–	–	100	–	–	ns
Output Strobe Delay Time		t_{DSD1} t_{DSD2}	Fig.11	–	–	200	–	–	200	–	–	200	–	–	200	ns

TIMER, SCI TIMING

Item		Symbol	Test Condition	HD6303Y			HD63A03Y			HD63B03Y			HD63C03Y			Unit
				min	typ	max	min	typ	max	min	typ	max	min	typ	max	
Timer 1 Input Pulse Width		t_{PWT}	Fig 9	2 0	—	—	2.0	—	—	2 0	—	—	2 0	—	—	t_{cyc}
Delay Time (Enable Positive Transition to Timer Output)		t_{TOD}	Fig 7, 8	—	—	400	—	—	400	—	—	400	—	—	400	ns
SCI Input Clock Cycle	Async Mode	t_{Scyc}	Fig 9	1 0	—	—	1 0	—	—	1 0	—	—	1 0	—	—	t_{cyc}
	Clock Sync		Fig 4	2 0	—	—	2 0	—	—	2 0	—	—	2 0	—	—	t_{cyc}
SCI Transmit Data Delay Time (Clock Sync Mode)		t_{TXD}		—	—	220	—	—	220	—	—	220	—	—	220	ns
SCI Receive Data Set-up Time (Clock Sync Mode)		t_{SRX}	Fig 4	260	—	—	260	—	—	260	—	—	260	—	—	ns
SCI Receive Data Hold Time (Clock Sync Mode)		t_{HRX}		100	—	—	100	—	—	100	—	—	100	—	—	ns
SCI Input Clock Pulse Width		t_{PWSCK}	Fig 9	0 4	—	0.6	0.4	—	0 6	0.4	—	0.6	0 4	—	0 6	t_{Scyc}
Timer 2 Input Clock Cycle		t_{tcyc}		2 0	—	—	2 0	—	—	2 0	—	—	2 0	—	—	t_{cyc}
Timer 2 Input Clock Pulse Width		t_{PWTCK}		200	—	—	200	—	—	200	—	—	200	—	—	ns
Timer 1-2, SCI Input Clock Rise Time		t_{CKr}		—	—	100	—	—	100	—	—	100	—	—	50	ns
Timer 1-2, SCI Input Clock Fall Time		t_{CKf}		—	—	100	—	—	100	—	—	100	—	—	50	ns

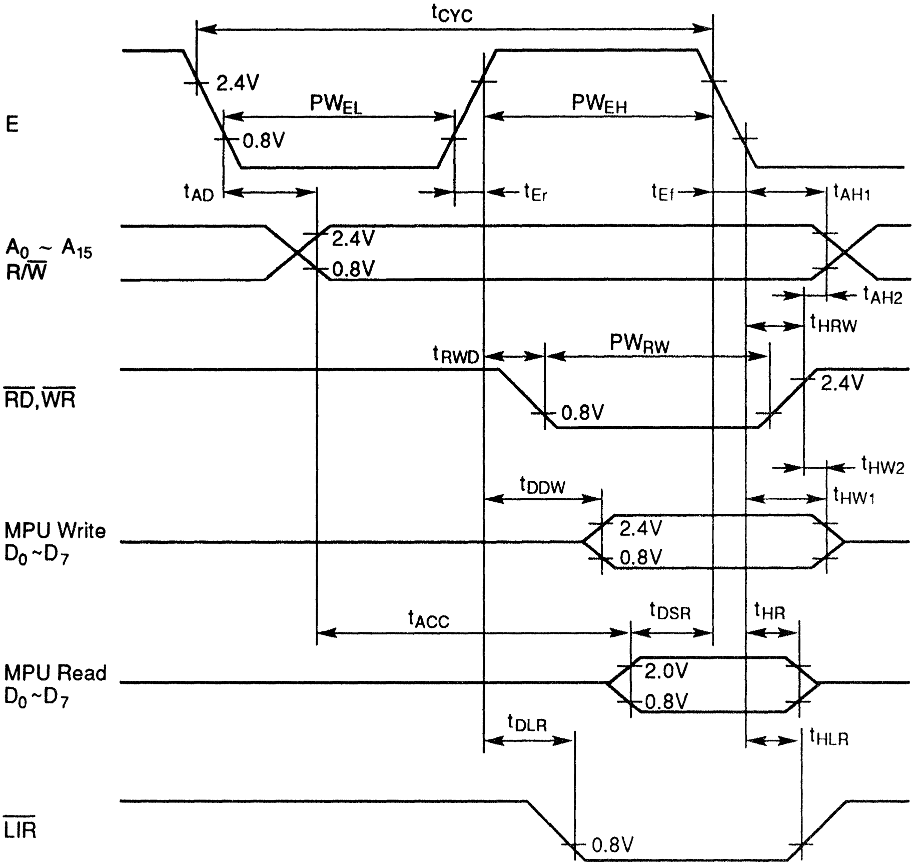


Figure 1 Bus Timing

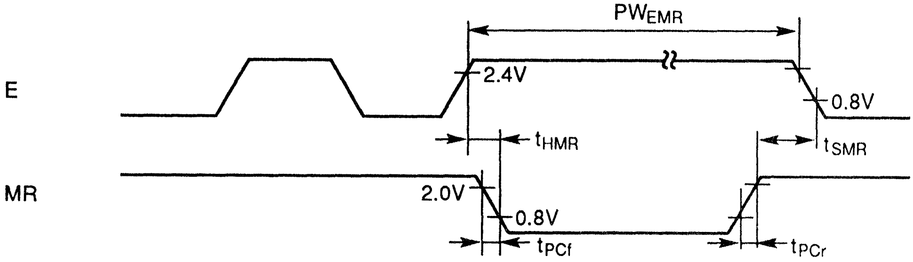


Figure 2 Memory Ready and E Clock Timing

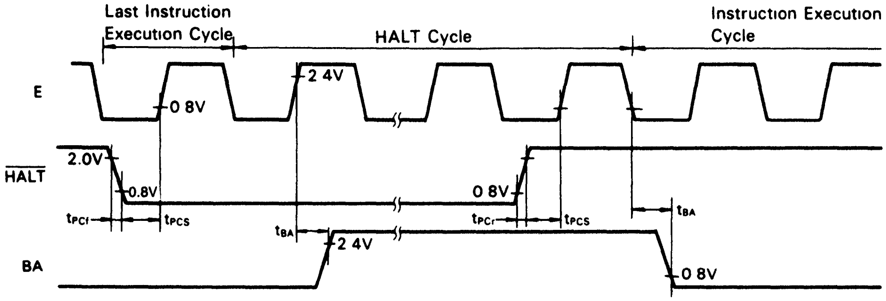
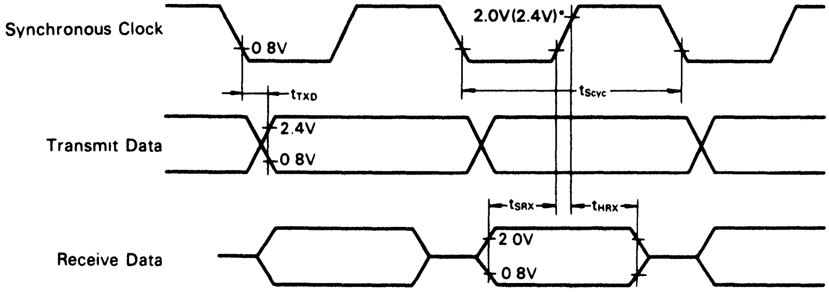


Figure 3 HALT and BA Timing



*2 0V is high level when clock input
2.4V is high level when clock output

Figure 4 SCI Clocked Synchronous Timing

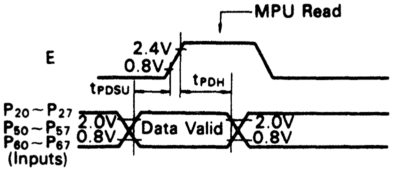


Figure 5 Port Data Set-up and Hold Times (MPU Read)

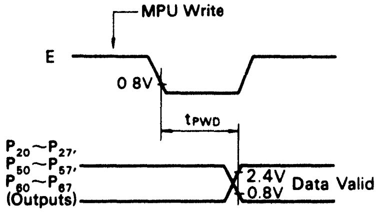


Figure 6 Port Data Delay Times (MPU Write)

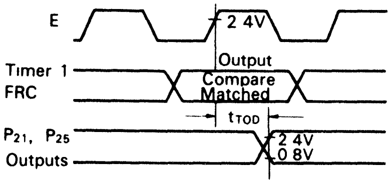
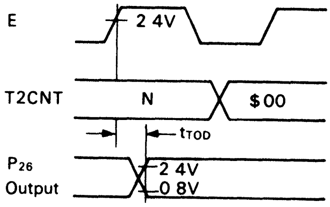
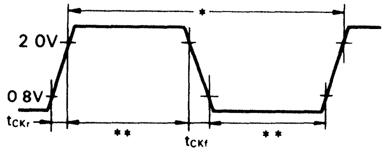


Figure 7 Timer 1 Output Timing



(TCONR=N)

Figure 8 Timer 2 Output Timing



*Timer 2 , ttcyc
SCI , tScyc

**Timer 1 , tPWT
Timer 2 , tPWTCK
SCI , tPWSCK

Figure 9 Timer 1·2, SCI Input Clock Timing

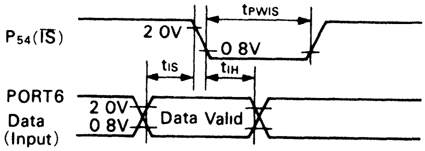


Figure 10 Port 6 Input Latch Timing

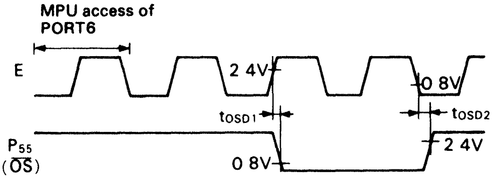
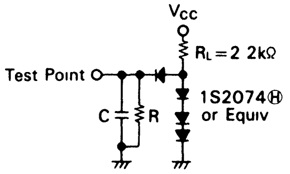


Figure 11 Output Strobe Timing



C = 90pF for $D_0 \sim D_7$, $A_0 \sim A_{15}$, E
= 30pF for Port 2, Port 5, Port 6, $\overline{RD}$, $\overline{WR}$, R/$\overline{W}$, BA, $\overline{LIR}$
R = 12kΩ

Figure 12 Bus Timing Test Loads (TTL Load)

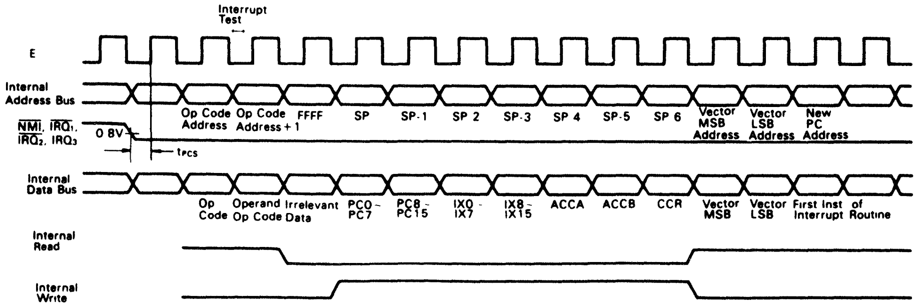


Figure 13 Interrupt Sequence

HITACHI

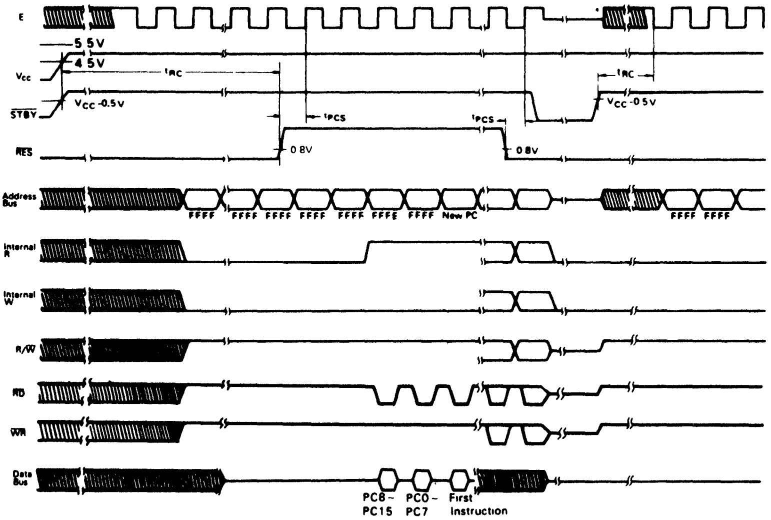


Figure 14 Reset Timing

■ **FUNCTIONAL PIN DESCRIPTION**

- **V_{CC}, V_{SS}**

V_{CC} and V_{SS} provide power to the MPU with 5V± 10% supply. In the case of low speed operation (fmax=500kHz), the MPU can operate with 3 to 5.5 volts. Two V_{SS} pins should be tied to ground.

- **XTAL, EXTAL**

These two pins interface with an AT-cut parallel resonant crystal. Divide-by-four circuit is on chip, so if 4MHz crystal oscillator is used, the system clock is 1MHz for example.

EXTAL pin can be drived by the external clock with 45% to 55% duty. The system clock which is one fourth frequency of the external clock is generated in the LSI. The external clock frequency should be less than four times of the maximum operating frequency. When using the external clock, XTAL pin should be open. Fig. 15 shows examples of connection circuit. The crystal and C_{L1}, C_{L2} should be mounted as close as possible to XTAL and EXTAL pins. Any line must not cross the line between the crystal oscillator and XTAL, EXTAL.

AT Cut Parallel Resonant Crystal Oscillator

Co=7pF max

Rs=60Ω max

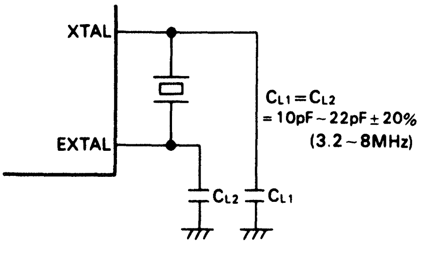


Figure 15 Connection Circuit

- **$\overline{STBY}$**

This pin makes the MPU standby mode. In "Low" level, the oscillation stops and the internal clock is stabilized to make reset condition. To retain the contents of RAM at standby mode, "0" should be written into RAM enable bit (RAME). RAME is the bit 6 of the RAM/port 5 control register at $0014. RAM is disabled by this operation and its contents is sustained.

Refer to "LOW POWER DISSIPATION MODE" for the standby mode.

- **Reset ($\overline{RES}$)**

This pin resets the MPU from power OFF state and provides a startup procedure. During power-on, $\overline{RES}$ pin must be held "Low" level for at least 20ms.

The CPU registers (accumulator, index register, stack pointer, condition code register except for interrupt mask bit), RAM and the data register of ports are not initialized during reset, so their contents are undefined in this procedure.

To reset the MPU during operation, $\overline{RES}$ should be held "Low" for at least 3 system-clock cycles. At the 3rd cycle during "Low" level, all the address buses become "High". When $\overline{RES}$ remains "Low", the address buses keep "High". If $\overline{RES}$ becomes "High", the MPU starts the next operation.

(1) Latch the value of the mode program pins; MP_0 and MP_1.

(2) Initialize each internal register (Refer to Table 4).

(3) Set the interrupt mask bit. For the CPU to recognize the maskable interrupts $\overline{IRQ_1}$, $\overline{IRQ_2}$ and IRQ_3, this bit should be cleared in advance.

(4) Put the contents (=start address) of the last two addresses ($FFFE, $FFFF) into the program counter and start the program from this address. (Refer to Table 1).

- **Enable (E)**

This pin provides a TTL-compatible system clock to external circuits. Its frequency is one fourth that of the crystal oscillator or external clock. This pin can drive one TTL load and 90pF capacitance.

- **Non-Maskable Interrupt ($\overline{NMI}$)**

When the falling edge of the input signal is detected at this pin, the CPU begins non-maskable interrupt sequence internally. As well as the $\overline{\mathrm{IRQ}}$ mentioned below, the instruction being executed at $\overline{\mathrm{NMI}}$ signal detection will proceed to its compeletion. The interrupt mask bit of the condition code register doesn't affect non-maskable interrupt at all.

In response to an $\overline{\mathrm{NMI}}$ interrupt, the contents of the program counter, index register, accumulators and condition code register will be saved onto the stack. Upon completion of this sequence, a vector is fetched from $FFFC and $FFFD to transfer their contents into the program counter and branch to the non-maskable interrupt service routine.

(Note) At reset start, the stack pointer should be initialized on an appropriate memory area and then the falling edge be input to $\overline{\mathrm{NMI}}$ pin.

- **Interrupt Request ($\overline{\mathrm{IRQ}}_1$, $\overline{\mathrm{IRQ}}_2$)**

These are level-sensitive pins which request an internal interrupt sequence to the CPU. At interrupt request, the CPU will complete the current instruction before the acceptance of the request. Unless the interrupt mask in the condition code register is set, the CPU starts an interrupt sequence; if set, the interrupt request will be ignored. When the sequence starts, the contents of the program counter, index register, accumulators and condition code register will be saved onto the stack, then the CPU sets the interrupt mask bit and will not acknowledge the maskable request. During the last cycle, the CPU fetches vectors depicted in Table 1 and transfers their contents to the program counter and branches to the service routine.

The CPU uses the external interrupt pins ($\overline{\mathrm{IRQ}}_1$ and $\overline{\mathrm{IRQ}}_2$) also as port pins P_{50} and P_{51}, so it provides an enable bit to Bit 0 and 1 of the RAM port 5 control register at $0014. Refer to "RAM/PORT 5 CONTROL REGISTER" for the details.

When one of the internal interrupts, ICI, OCI, TOI, CMI or SIO is generated, the CPU produces internal interrupt signal (IRQ_3). IRQ_3 functions just the same as $\overline{\mathrm{IRQ}}_1$ or $\overline{\mathrm{IRQ}}_2$ except for its vector address. Fig. 16 shows the block diagram of the interrupt circuit.

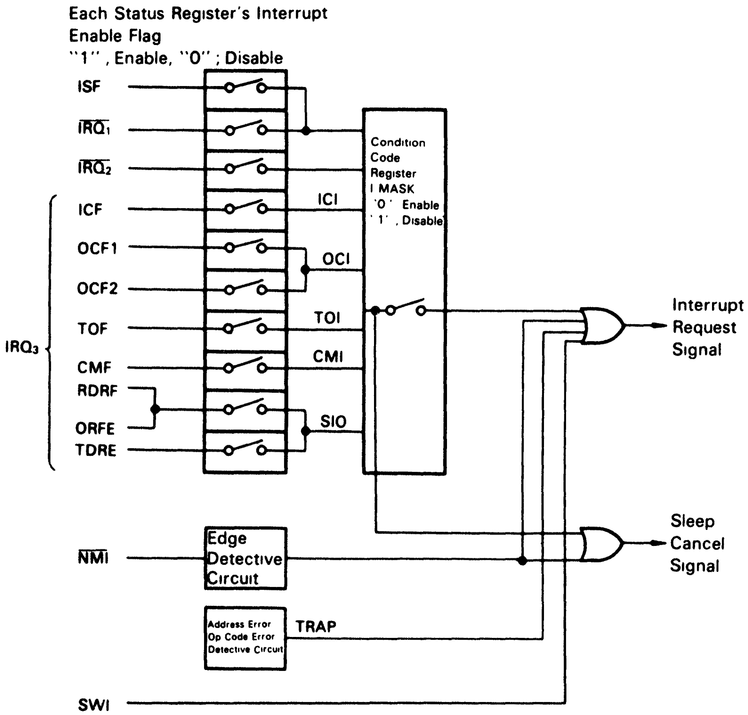


Figure 16 Interrupt Circuit Block Diagram

Table 1 Interrupt Vector Memory Map

Priority	Vector		Interrupt
	MSB	LSB	
Highest	FFFE	FFFF	$\overline{\text{RES}}$
	FFEE	FFEF	TRAP
	FFFC	FFFD	$\overline{\text{NMI}}$
	FFFA	FFFB	SWI (Software Interrupt)
	FFF8	FFF9	$\overline{\text{IRQ}}_1$, ISF (port 6 Input Strobe)
	FFF6	FFF7	ICI (Timer 1 Input Capture)
	FFF4	FFF5	OCI (Timer 1 Output Compare 1, 2)
	FFF2	FFF3	TOI (Timer 1 Overflow)
	FFEC	FFED	CMI (Timer 2 Counter Match)
	FFEA	FFEB	$\overline{\text{IRQ}}_2$
Lowest	FFF0	FFF1	SIO (RDRF+ORFE+TDRE+PER)

- **Mode Program (MP$_0$, MP$_1$)**

Set MP$_0$ "High" and MP$_1$ "Low"

- **Read/Write (R/$\overline{\text{W}}$)**

This signal, usually be in read state ("High"), shows whether the CPU is in read ("High") or write ("Low") state to the peripheral or memory devices. This can drive one TTL load and 30pF capacitance.

- **$\overline{\text{RD}}$, $\overline{\text{WR}}$**

These signals show active low outputs when the CPU is reading/writing to the peripherals or memories. This enables the CPU easy to access the peripheral LSI with $\overline{\text{RD}}$ and $\overline{\text{WR}}$ input pins. These pins can drive one TTL load and 30pF capacitance.

- **Load Instruction Register ($\overline{\text{LIR}}$)**

This signal shows the instruction opecode being on data bus (active low). This pin can drive one TTL load and 30pF capacitance.

- **Memory Ready (MR; P$_{52}$)**

This is the input control signal which stretches the system clock's "High" period to access low-speed memories. HD6303Y can select three kinds of low-speed memory access method by RAM/Port 5 Control Register's MRE bit and AMRE bit. In the case that CPU accesses low-speed memories by the external MR signal (MRE="1", AMRE="0"), the system clock operates in normal sequence when this signal is in "High".

But this signal in "Low", the "High" period of the system clock will be stretched depending on its "Low" level duration in integral multiples of the cycle time. This allows the CPU to interface with low-speed memories (See Fig. 2). Up to 9μs can be stretched.

During internal address space access or nonvalid memory access, MR is prohibited internally to prevent decrease of operation speed. Even in the halt state, MR can also stretch "High" period of system clock to allow peripheral devices to access low-speed memories. Refer to "RAM/PORT 5 CONTROL REGISTER" for more details.

- **Halt ($\overline{\text{HALT}}$; P$_{53}$)**

This is an input control signal to stop instruction execution and to release buses. When this signal switches to "Low", the CPU stops to enter into the halt state after having executed the present instruction. When entering into the halt state, it makes BA "High" and also an address bus, data bus, $\overline{\text{RD}}$, $\overline{\text{WR}}$, R/$\overline{\text{W}}$ high impedance. When an interrupt is generated in the halt state, the CPU uses the interrupt handler after the halt is cancelled. When halted during the sleep state, the CPU keeps the sleep state, while BA is "High" and releases the buses. Then the CPU returns to the previous sleep state when the $\overline{\text{HALT}}$ signal becomes "High".

(Note) Please don't switch the $\overline{\text{HALT}}$ signal to "Low" when the CPU executes the WAI instruction and is in the interrupt wait state to avoid the trouble of the CPU's operation after the halt is cancelled.

- **Bus Available (BA)**

This is an output control signal which is normally "Low" but "High" when the CPU accepts $\overline{\text{HALT}}$ and releases the buses. The HD6800 and HD6802 make BA "High" and release the buses at WAI execution, while the HD6303Y doesn't make BA "High" under the same condition.

■ PORT

The HD6303Y provides three 8-bit I/O ports. Each port provides Data Direction Register (DDR) which controls the I/O state by the bit.

Table 2 Port and Data Direction Register Address

Port	Port Address	Data Direction Register
Port 2	$0003	$0001
Port 5	$0015	$0020
Port 6	$0017	$0016

- **Port 2**

An 8-bit I/O port. Port 2 DDR (P2DDR) controls the I/O state. This port provides DDR corresponding to each bit and can define input or output by the bit ("0" for input, "1" for output).

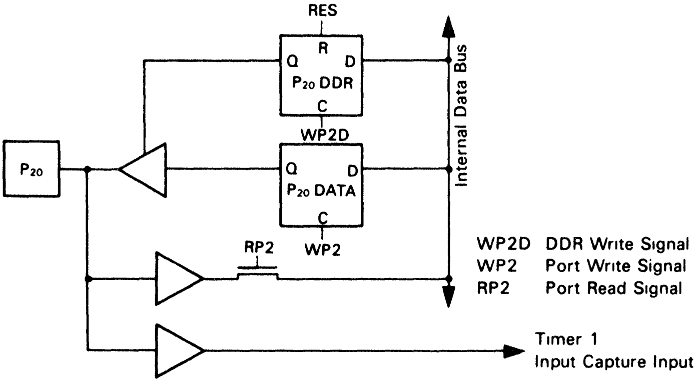
As Port 2 DDR is cleared during reset, it will be an input port.

Port 2 is also used as an I/O pin for timer 1, Timer 2 and the SCI. Pins for Timers and the SCI set or reset each DDR depending on their functions and become I/O pins. When port 2 functions as an I/O port after used as I/O pins of the timers or the SCI, the I/O direction of the pins remain as it is used as the I/O pin of timer and SCI.

Port 2 can drive one TTL load and 30pF capacitance. This port can produce 1mA when V$_{\text{out}}$=1.5V to drive directly the base of Darlington transistor.

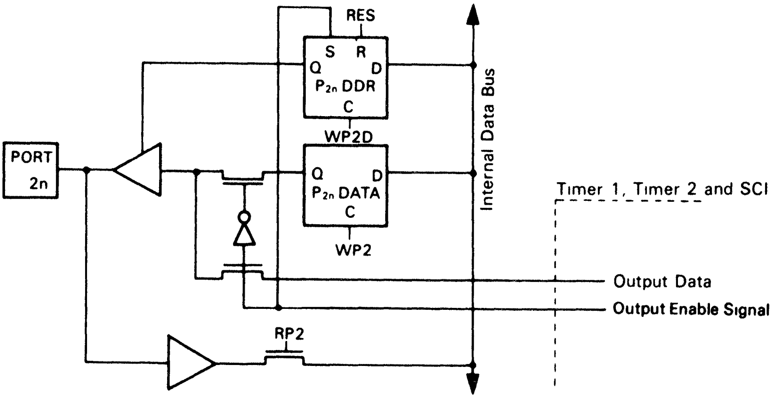
P$_{20}$ (Tin)

P$_{20}$ is also used as an external input pin for the input-capture. This pin is an I/O port which is an input or output as defined by the Data Direction Register (P$_{20}$DDR) ("0" for an input and "1" for an output) Then either a signal to or from P$_{20}$ ("to" for an output port, "from" for an input port) is always input to the Timer 1 input capture.



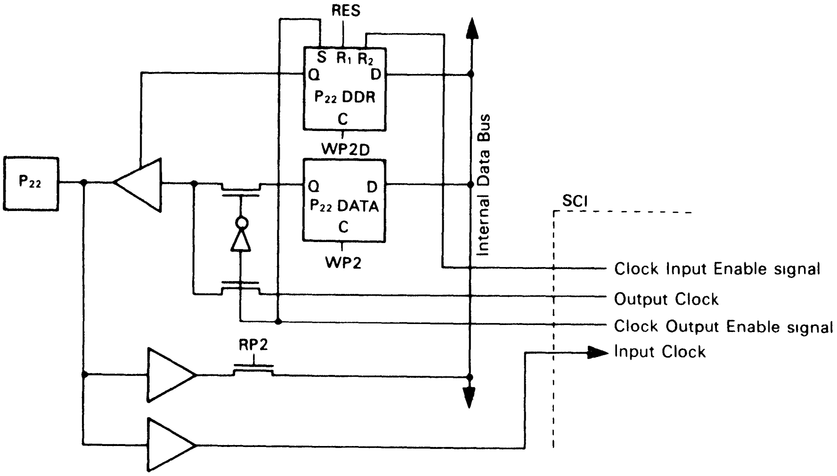
P_{21} (Tout 1), P_{24} (Tx), P_{25} (Tout 2), P_{26} (Tout 3)

These four pins can be also used as output pins for Timer 1, Timer 2 and a transmit output of the SCI. Timer 1, and the SCI have a register which enables output. By setting these registers, they automatically will be output pins of timer or the SCI.



P_{22} (SCLK)

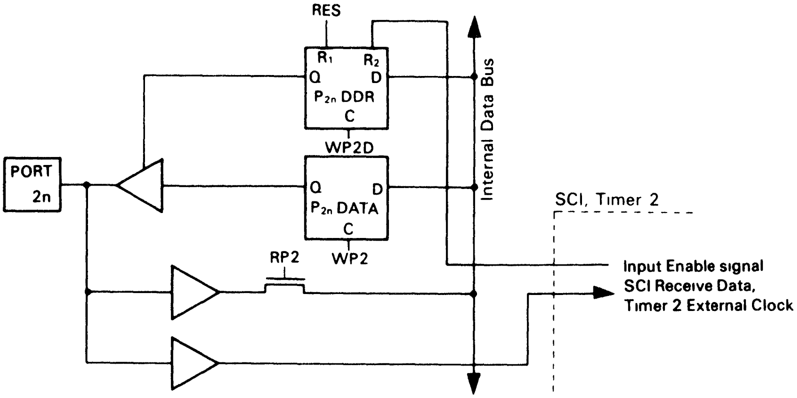
P_{22} is also used as a clock I/O pin for the SCI. It is selected as a clock input or output pin by the operating mode of the SCI. It is usable as an I/O port when the SCI has no clock input or output (as an output port if P_{22} DDR=1, as an input port if P_{22} DDR=0).



P_{23} (Rx), P_{27} (TCLK)

P_{23} and P_{27} are also used as received data input pins for the SCI and external clock input pins for Timer 2. The SCI and Timer 2 have registers which enable input. If the registers are set, the DDR (P_{23}DDR, P_{27}DDR) are cleared and P_{23} and P_{27} will be input pins for Rx and TCLK

Since the SCI will be a clocked synchronous mode by an external clock-input during reset, the DDR of P_{22} is cleared automatically and P_{22} is an input port. Set the SCI to a mode where P_{22} is not used (CC0 or CC1 of the RMC Register is "0" or "1" respectively) and write "1" to the P_{22} DDR to make P_{22} an output port



MSB ... LSB

P_{27} DDR	P_{26} DDR	P_{25} DDR	P_{24} DDR	P_{23} DDR	P_{22} DDR	P_{21} DDR	P_{20} DDR

PORT2 DDR ($0001) (Write only, $00 during reset)

P_{27}	P_{26}	P_{25}	P_{24}	P_{23}	P_{22}	P_{21}	P_{20}

PORT2 ($0003) (R/W, not initialized during reset)

- **Port 5**

An 8-bit I/O port. The DDR of port 5 controls I/O state. Each bit of port 5 has a DDR which defines I/O state ("0" for input and "1" for output).

During reset, the DDR of port 5 is cleared and port 5 becomes an input port.

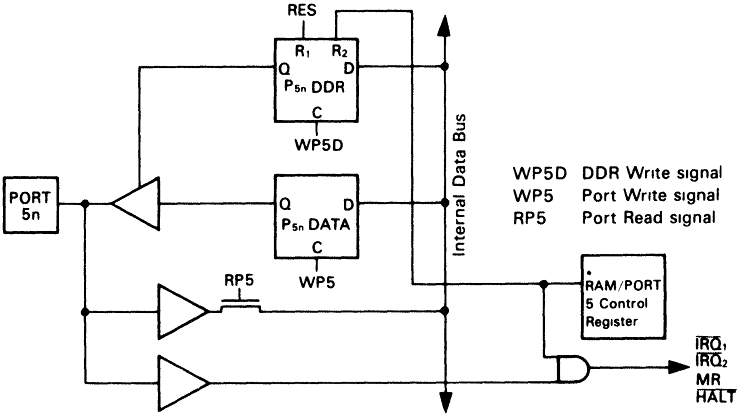
Port 5 is also usable as $\overline{IRQ_1}$, $\overline{IRQ_2}$, $\overline{HALT}$, MR and the strobed signal of port 6 for handshake ($\overline{IS}$, $\overline{OS}$). It is set to input or output automatically if it is used as these control signal pins (except P_{54}, $\overline{IS}$). Since the DDR of port 5, as is port 2, is set or reset by the control signal, I/O directions of the I/O ports are retained after the control signal is disabled. Port 5 can drive one TTL load and 90pF capacitance.

P_{50} ($\overline{IRQ_1}$), P_{51} ($\overline{IRQ_2}$)

P_{50} and P_{51} are also usable as interrupt pins. The RAM/port 5 control registers of $\overline{IRQ_1}$ and $\overline{IRQ_2}$ have enable bits (IQ1E, IQ2E). When these bits are set to "1", P_{50} and P_{51} will automatically be interrupt input pins.

P_{52} (MR), P_{53} ($\overline{HALT}$)

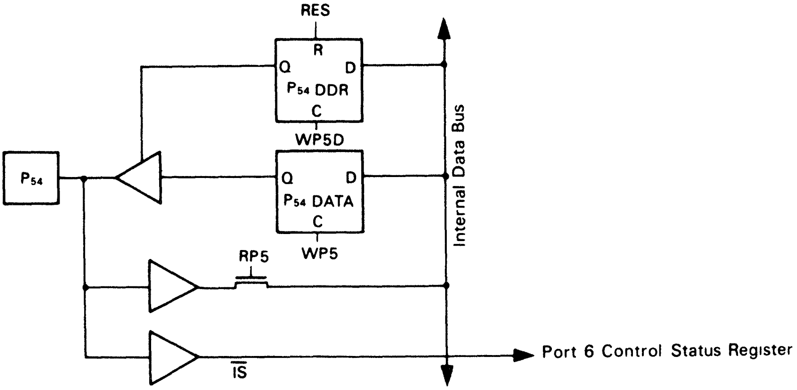
P_{52} and P_{53} are also usable as MR and $\overline{HALT}$ inputs. MR and $\overline{HALT}$ have enable bits (MRE, HLTE) in the RAM/Port 5 Control Register as $\overline{IRQ_1}$ and $\overline{IRQ_2}$. Since MRE is cleared during reset, P_{52} is usable as an I/O port, and HLTE is set during reset, the DDR of P_{53} will be automatically reset to be a $\overline{HALT}$ input pin. HLTE of the RAM/Port 5 Control Register has to be cleared to use P_{53} as an I/O port.



* Initializing value during reset;
IRQ1E = "0", IRQ2E = "0", MRE = "0", HLTE = "1"

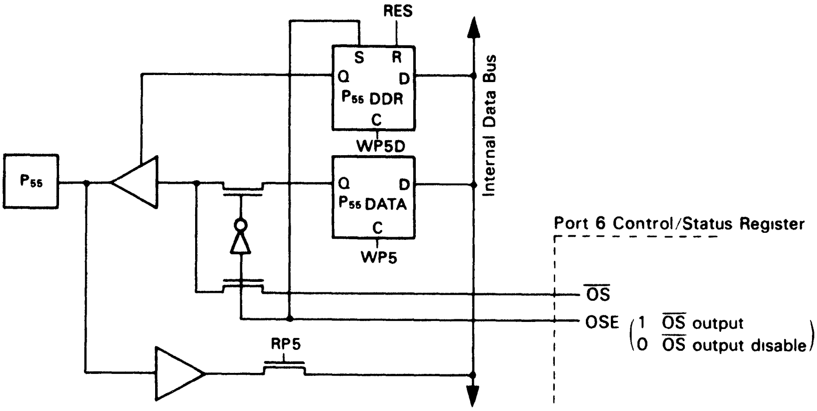
P_{54} ($\overline{IS}$)

P_{54} is also usable as the input strobe ($\overline{IS}$) for port 6 handshake interface. This pin, as is P_{20}, is always an I/O port. If P_{54} is used as an output port (set the DDR of P_{54} to "1"), an output signal from P_{54} will be the input to $\overline{IS}$.



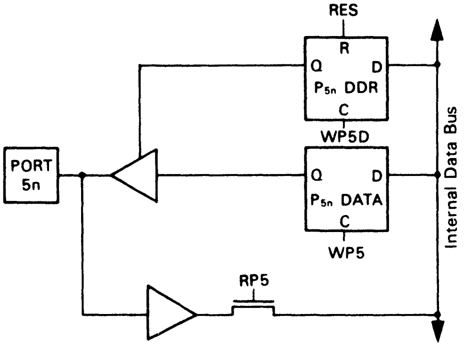
P_{55} ($\overline{OS}$)

P_{55} is also usable as the output strobe ($\overline{OS}$) for port 6 handshake interface. It will be an I/O port during reset, and an $\overline{OS}$ output pin by setting the $\overline{OS}$ enable register (OSE) of the port 6 Control Status Register (P6CSR).



P_{56}, P_{57}

P_{56} and P_{57} are I/O ports.



MSB LSB

P_{57} DDR	P_{56} DDR	P_{55} DDR	P_{54} DDR	P_{53} DDR	P_{52} DDR	P_{51} DDR	P_{50} DDR

PORT5 DDR ($0020)
(Write only, $00 during reset)

P_{57}	P_{56}	P_{55}	P_{54}	P_{53}	P_{52}	P_{51}	P_{50}

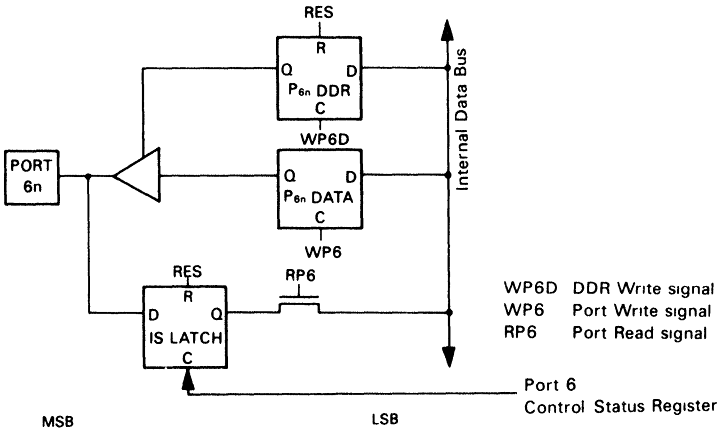
PORT5 ($0015)
(R/W, not initialized during reset)

- **Port 6**

8-bit I/O port. Port 6 DDR controls I/O state. Each bit of port 6 has a DDR and designates input or output ("0" for input, "1" for output). During reset, Port 6 DDR is cleared and port 6 becomes an input port.

Port 6 controls parallel handshake interface besides functions as an I/O port. Therefore, it provides DDRs to control and IS LATCH to latch the input data.

Port 6 can drive one TTL load and 30pF capacitance. It can drive directly the base of Darlington transistor as port 2.



MSB							LSB	
P_{67} DDR	P_{66} DDR	P_{65} DDR	P_{64} DDR	P_{63} DDR	P_{62} DDR	P_{61} DDR	P_{60} DDR	PORT6 DDR ($0016) (Write only, $00 during reset)

P_{67}	P_{66}	P_{65}	P_{64}	P_{63}	P_{62}	P_{61}	P_{60}	PORT6 ($0017) (R/W, not initialized during reset)

■ BUS

- **Address Bus ($A_0 \sim A_{15}$)**

Address Bus ($A_0 \sim A_{15}$) is used for addressing the memory and peripheral LSI.

This bus can interface with the bus of HMCS 6800 and drive one TTL load ai.d 90pF capacitance.

- **Data Bus ($D_0 \sim D_7$)**

8-bit parallel data bus for data transmit between the memory or peripheral LSI. This bus can drive one TTL load and 90pF capacitance.

■ RAM/PORT 5 CONTROL REGISTER

The control register located at $0014 controls on-chip RAM and port 5.

RAM/Port 5 Control Register (RP5CR)

7	6	5	4	3	2	1	0	
STBY PWR	RAME	STBY FLAG	AMR E	HLTE	MRE	IRQ_2 E	IRQ_1 E	$0014

Bit 0, Bit 1 $\overline{IRQ_1}$, $\overline{IRQ_2}$ Enable Bit (IRQ_1E, IRQ_2E)

When using P_{50} and P_{51} as interrupt pins, write "1" in these bits. When the bit is set to "1", the DDRs corresponding to P_{50} and P_{51} are cleared and become $\overline{IRQ_1}$ input pin and $\overline{IRQ_2}$ input pin. When IRQ_1E and IRQ_2E are set, P_{50} and P_{51} cannot be used as an output ports. When "0", the CPU doesn't accept an external interrupt or a sleep cancellation by the external interrupt. These bits are cleared during reset.

Bit 2 Memory Ready Enable Bit (MRE)

When using P_{52} as an input pin of the "memory ready" signal, write "1" in this bit When set, P_{52} DDR is automatically cleared and becomes the MR input pin. The bit is cleared during reset.

Bit 3 Halt Enable Bit (HLTE)

When using P_{53} as an input pin of the $\overline{HALT}$ signal, write "1" in this bit. When this bit is set, P_{53} DDR is automatically cleared and becomes the Halt input pin. If the bit is "0", the Halt function is inhibited and P_{53} is used as an I/O port. The bit is set to "1" during reset.

Bit 4 Auto Memory Ready Enable Bit (AMRE)

When the bit is set and the CPU accesses the external address, "memory ready" operates automatically and stretches the E clock's "High" duration for one system clock. When MRE bit of bit 2 is cleared and when the CPU accesses the external address space, the function operates. When MRE bit is set and then the CPU accesses the external address space with P_{52}(MR) pin in "low", "memory ready" operates automatically. This bit is set to "1" during reset.

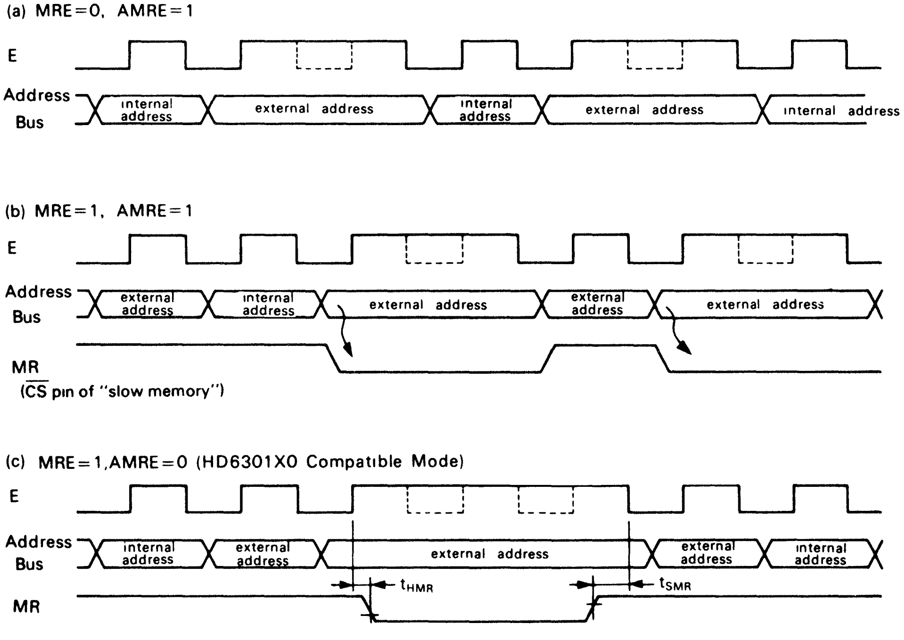
Table 3 "Memory Ready" Function

MRE	AMRE	Function
0	0	"Memory ready" inhibited.
0	1	When the CPU accesses the external address, "High" duration of E clock automatically becomes one-cycle longer This state is retained during reset
1	0	"Memory ready" operates by P_{52} (MR) pin The function is the same as that of the HD6301X0
1	1	When the CPU accesses the external address space with the P_{52} (MR) pin in "low", the "auto memory ready" operates This function is effective if it has both "high-speed memory" and "slow memory" outside Input $\overline{CS}$ signal of "slow memory" to MR pin

Bit 5 Standby Flag (STBY FLAG)
By clearing this flag, HD6303Y gets into the standby mode by software. This flag is set to "1" during reset, so the standby mode is canceled with $\overline{RES}$ pin in "low" The $\overline{RES}$ pin should be in "low" until oscillation becomes stable (min 20ms.). If the $\overline{STBY}$ pin in is in "low", the standby mode can not be canceled with the $\overline{RES}$ pin in "low".

Bit 6 RAM Enable (RAME)
On-chip RAM can be disabled by this control bit By resetting the MPU, "1" is set to this bit, and on-chip RAM is enabled When this bit is cleared (=logic "0") on-chip RAM is invalid and the CPU can read data from external memory. This bit should be "0" before getting into the standby mode to protect on-chip RAM data.

Bit 7 Standby Power Bit (STBY PWR)
When V_{CC} is not provided in standby mode, this bit is cleared This is a flag for read/write and can be read by software. If this bit is set before standyby mode, and remains set even after returning from standby mode, V_{CC} voltage is provided during standby mode and the on-chip RAM data is valid.


(a) MRE=0, AMRE=1

(b) MRE=1, AMRE=1

(c) MRE=1,AMRE=0 (HD6301X0 Compatible Mode)

Figure 17 Memory Ready Timing

- **Port 6 Control/Status Register**

This is the Control/Status Register for parallel handshake interface using Port 6. The functions are as follows,

1) Latches input data to Port 6 at the $\overline{IS}$ (P_{54}) falling edge
2) Outputs a strobe signal $\overline{OS}$ (P_{55}) outward by reading or writing to port 6.
3) When IS FLAG is set at the $\overline{IS}$ falling edge, an interrupt occurs.

The following shows Port 6 Control/Status Register (P6CSR).

7	6	5	4	3	2	1	0	
IS* FLAG	IS IRQ_1 ENABLE	OSE	OSS	LATCH ENABLE	–	–	–	$0021

*Bit 7 is Read-Only bit

Bit 0
Bit 1 Not used.
Bit 2

Bit 3: Latch Enable

This register controls the input latch for Port 6 (ISLATCH). When this bit is set to "1", the input data to port 6 will be latched inward at the $\overline{\text{IS}}$ (P_{64}) falling edge. An input latch will be canceled by reading Port 6, which enables to latch the next data. If cleared, the input latch remains canceled and this bit functions as a usual input port. This bit is cleared during reset.

Bit 4: OSS Output Strobe Select

This register initiates an output strobe ($\overline{\text{OS}}$) from P_{55} by reading or writing to port 6. When cleared, $\overline{\text{OS}}$ occurs by reading Port 6. When set, $\overline{\text{OS}}$ occurs by writing to Port 6. This bit is cleared during reset.

Bit 5: OSE Output Strobe Enable

This register decides the enabling or disabling of the output strobe. When cleared, P_{55} functions as an I/O port. When set, P_{55} functions as an $\overline{\text{OS}}$ output pin. (P_{55} DDR is set by OSE.) This bit is cleared during reset.

Bit 6: IS IRQ_1 Enable Input Strobe Interrupt Enable

When set, an $\overline{\text{IRQ}_1}$ interrupt to the CPU occurs by setting IS FLAG of bit 7. When cleared, the interrupt does not occur. This bit is cleared during reset.

Bit 7: IS Flag Input Strobe Flag

This flag is set at the IS (P_{64}) falling edge. This flag is for read-only. When set, the flag is cleared by reading or writing to Port 6 after reading the Port 6 Control Status Register. This bit is cleared during reset.

■ MEMORY MAP

The MPU can address up to 65k bytes. Memory map is shown in Fig. 20. 40 addresses ($0000 ~ $0027 except $00, $02, $04, $05, $06, $07, $18) are the internal registers as shown in Table 4.

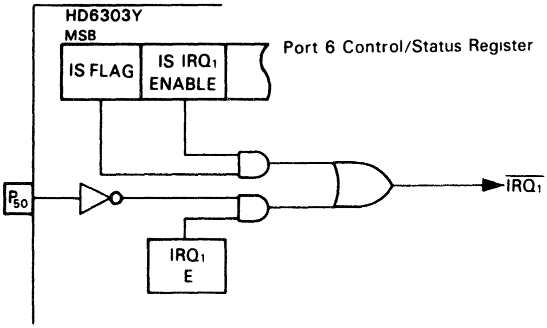


Figure 18 Input Strobe Interrupt block Diagram

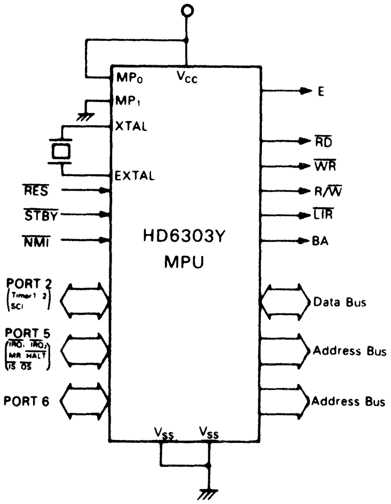
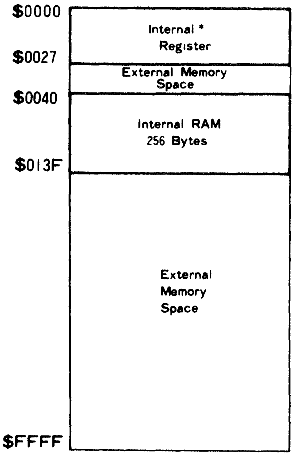


Figure 19 HD6303Y Operating Function

Table 4 Internal Register

Address	Register	Abbreviation	R/W**	Initialized value during reset***
00*	Port 1 DDR (Data Direction Register)	P1DDR	W	$FE
01	Port 2 DDR	P2DDR	W	$00
02*	Port 1	PORT1	R/W	indefinite
03	Port 2	PORT2	R/W	indefinite
04*	Port 3 DDR	P3DDR	W	$FE
05*	Port 4 DDR	P4DDR	W	$00
06*	Port 3	PORT3	R/W	indefinite
07*	Port 4	PORT4	R/W	indefinite
08	Timer Control/Status Register 1	TCSR1	R/W	$00
09	Free Running Counter (MSB)	FRCH	R/W	$00
0A	Free Running Counter (LSB)	FRCL	R/W	$00
0B	Output Compare Register 1 (MSB)	OCR1H	R/W	$FF
0C	Output Compare Register 1 (LSB)	OCR1L	R/W	$FF
0D	Input Capture Register (MSB)	ICRH	R	$00
0E	Input Capture Register (LSB)	ICRL	R	$00
0F	Timer Control/Status Register 2	TCSR2	R/W	$10
10	Rate/Mode Control Register	RMCR	R/W	$C0
11	Tx/Rx Control Status Register 1	TRCSR1	R/W	$20
12	Receive Data Register	RDR	R	$00
13	Transmit Data Register	TDR	W	indefinite
14	RAM/Port 5 Control Register	RP5CR	R/W	$F8 or $78
15	Port 5	PORT5	R/W	indefinite
16	Port 6 DDR	P6DDR	W	$00
17	Port 6	PORT6	R/W	indefinite
18	Port 7	PORT7	R/W	indefinite
19	Output Compare Register 2 (MSB)	OCR2H	R/W	$FF
1A	Output Compare Register 2 (LSB)	OCR2L	R/W	$FF
1B	Timer Control/Status Register 3	TCSR3	R/W	$20
1C	Time Constant Register	TCONR	W	$FF
1D	Timer 2 Up Counter	T2CNT	R/W	$00
1E	Tx/Rx Control Status Register 2	TRCSR2	R/W	$28
1F****	Test Register*	TSTREG	–	–
20	PORT 5 DDR	P5DDR	W	$00
21	PORT 6 Control/Status Register	P6CSR	R/W	$07
22	— Reserved	–	–	–
23	—	–	–	–
24	—	–	–	–
25	—	–	–	–
26	—	–	–	–
27	—	–	–	–

* External address
** R Read-only register, W Write-only register, R/W Read/Write register
*** When empty bit is in the register, it is set to "1"
**** Register for test Don't access this register



*This mode does not include the addresses. $00, $02, $04, $05, $06, $07 or $18 which can be used externally.

Figure 20 HD6303Y Memory Map

■ TIMER 1

The HD6303Y provides a 16-bit programmable timer which can simultaneously measure an input waveform and generate two independent output waveforms. The pulse widths of both input/output waveforms vary from microseconds to seconds.

Timer 1 is configured as follows (refer to Fig. 22).

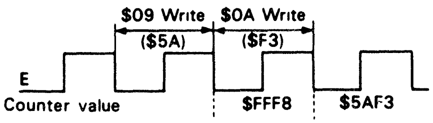
- Control/Status Register 1 (8 bit)
- Control/Status Register 2 (7 bit)
- Free Running Counter (16 bit)
- Output Compare Register 1 (16 bit)
- Output Compare Register 2 (16 bit)
- Input Capture Register (16 bit)

● Free-Running Counter (FRC)($0009:000A)

The key timer element is a 16-bit free-running counter driven and incremented by system clock The counter value is readable by software without affecting the counter. The counter is cleared during reset.

When writing to the upper byte ($09), the CPU writes the preset value ($FFF8) into the counter (address $09, $0A) regardless of the write data value. But when writing to the lower byte ($0A) after the upper byte writing, the CPU writes not only lower byte data into lower 8 bit, but also upper byte data into higher 8 bit of the FRC.

The counter will be as follows when the CPU writes to it by double store instructions (STD, STX, etc.)



In the case of the CPU write ($5AF3) to the FRC

Figure 21 Counter Write Timing

● Output Compare Register (OCR) ($000B, $000C; OCR1) ($0019, $001A: OCR2)

The output compare register is a 16-bit read/write register which can control an output waveform. The data of OCR is always compared with the FRC.

When the data matches, output compare flag (OCF) in the timer control/status register (TCSR) is set. If an output enable bit (OE) in the TCSR2 is "1", an output level bit(OLVL) in the TCSR will be output to bit 1 (OCR 1) and bit 5 (OCR 2) of port 2. To control the output level again by the next compare, the value of OCR and OLVL should be changed. The OCR is set to $FFFF at reset. The compare function is inhibited for a cycle just after a write to the upper byte of the OCR or FRC. This is to set the 16-bit value valid in the counter register for compare. In addition, it is because counter is to set $FFF8 at the next cycle of the CPU's upper byte write to the FRC.

* For data write to the FRC or the OCR, 2-byte transfer instruction (such as STX, etc.) should be used.

● Input Capture Register (ICR) ($000D : 000E)

The input capture register is a 16-bit read-only register which stores the FRC's value when external input signal transition generates an input capture pulse. Such transition is controled by input edge bit (IEDG) in the TCSR1.

In order to input the external input signal to the edge detector, a bit of the DDR corresponding to bit 0 of port 2 should be cleared ("0"). When an input capture pulse occurs by external input signal transition at the next cycle of CPU's high-byte read of the ICR, the input capture pulse will be delayed by one cycle. In order to ensure the input capture operation, a CPU read of the ICR needs 2-byte transfer instruction. The input pulse width should be at least 2 system cycles. This register is cleared ($0000) during reset.

● Timer Control/Status Register 1 (TCSR1) ($0008)

The timer control/status register 1 is an 8-bit register. All bits are readable and the lower 5 bits are also writable. The upper 3 bits are read-only which indicate the following timer status.

Bit 5 The counter value reached to $0000 as a result of counting-up (TOF).

Bit 6 A match has occurred between the FRC and the OCR 1 (OCF1).

Bit 7 Defined transition of the timer input signal causes the counter to transfer its data to the ICR (ICF).

The followings are the each bit descriptions.

Timer Control/Status Register 1

7	6	5	4	3	2	1	0	
ICF	OCF1	TOF	EICI	EOCI1	ETOI	IEDG	OLVL1	$0008

Bit 0 OLVL1 Output Level 1

OLVL1 is transferred to port 2, bit 1 when a match occurs between the counter and the OCR1. If bit 0 of the TCSR2 (OE1), is set to "1", OLVL1 will appear at bit 1 of port 2.

Bit 1 IEDG Input Edge

This bit determines which edge, rising or falling, of input signal of bit 0 of port 2 will trigger data transfer from the counter to the ICR For this function, the DDR corresponding to port 2, bit 0 should be cleared beforehand.

IEDG=0, triggered on a falling edge
("High" to "Low")
IEDG=1, triggered on a rising edge
("Low" to "High")

Bit 2 ETOI Enable Timer Overflow Interrupt

When this bit is set, an internal interrupt (IRQ_3) by TOI interrupt is enabled. When cleared, the interrupt is inhibited.

Bit 3 EOCI1 Enable Output Compare Interrupt 1

When this bit is set, an internal interrupt (IRQ_3) by OCI1 interrupt is enabled. When cleared, the interrupt is inhibited.

Bit 4 EICI Enable Input Capture Interrupt
When this bit is set, an internal interrupt (IRQ_3) by ICI interrupt is enabled. When cleared, the interrupt is inhibited

Bit 5 TOF Timer Overflow Flag
This read-only bit is set when the counter increments from $FFFF by 1. Cleared when the counter's MSB byte ($0009) is read by the CPU after the TCSR1 read at TOF=1.

Bit 6 OCF1 Output Compare Flag 1
This read-only bit is set when a match occurs between the OCR1 and the FRC. Cleared when writing to the OCR1 ($000B or $000C) after the TCSR1 or TCSR2 read at OCF=1

Bit 7 ICF Input Capture Flag
This read-only bit is set when an input signal of port 2, bit 0 makes a transition as defined by IEDG and the FRC is transferred to the ICR. Cleared when reading the upper byte ($000D) of the ICR after the TCSR1 or TCSR2 read at ICF=1.

- **Timer Control/Status Register 2 (TCSR2) ($000F)**

The timer control/status register 2 is a 7-bit register. All bits are readable and the lower 4 bits are also writable But the upper 3 bits are read-only which indicate the following timer status.

Bit 5 A match has occurred between the FRC and the OCR2 (OCF2).

Bit 6

Timer Control/Status Register 2

7	6	5	4	3	2	1	0	
ICF	OCF1	OCF2	–	EOCI2	OLVL2	OE2	OE1	$000F

Bit 7 The same status flag as the ICF flag of the TCSR1, bit 7
The followings are the each bit descriptions

Bit 0 OE1 Output Enable 1
This bit enables the OLVL1 to appear at port 2, bit 1 when a match has occurred between the counter and the output compare register 1 When this bit is cleared, bit 1 of port 2 will be an I/O port When set, it will be an output of OLVL1 automatically

Bit 1 OE2 Output Enable 2
This bit enables the OLVL2 to appear at port 2, bit 5 when a match has occurred between the counter and the output compare register 2 When this bit is cleared, port 2, bit 5 will be an I/O port. When set, it will be an output of OLVL2 automatically.

Bit 2 OLVL2 Output Level 2
OLVL2 is transferred to port 2, bit 5 when a match has occurred between the counter and the OCR2 If bit 5 of the TCSR2 (OE2), is set to "1", OLVL2 will appear at port 2, bit 5.

Bit 3 EOCI2 Enable Output Compare Interrupt 2
When this bit is set, an internal interrupt (IRQ_3) by OCI2 interrupt is enabled. When cleared, the interrupt is inhibited

Bit 4 Not used

Bit 5 OCF2 Output Compare Flag 2
This read-only bit is set when a match has occurred between the counter and the OCR2 Cleared when writing to the OCR2 ($0019 or $001A) after the TCSR2 read at OCF2=1

Bit 6 OCF1 Output Compare Flag 1

Bit 7 ICF Input Capture Flag
OCF1 and ICF are dual addressed If which register, TCSR1 or TCSR2, CPU reads, it can read OCF1 and ICF to bit 6 and bit 7.
Both the TCSR1 and TCSR2 will be cleared during reset.

(Note) If OE1 or OE2 is set to "1" before the first output compare match occurs after reset restart, bit 1 or bit 5 of port 2 will produce "0" respectively.

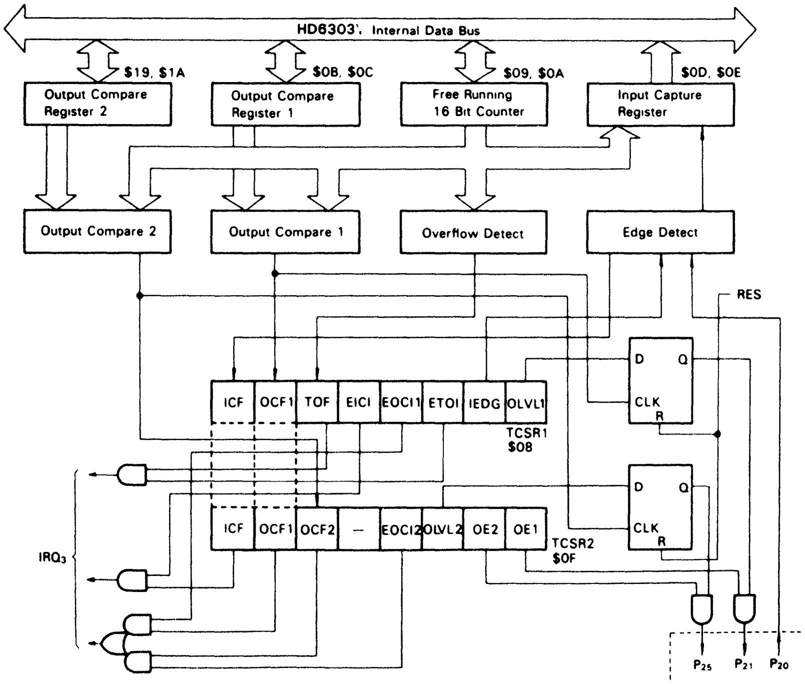


Figure 22 Timer 1 Block Diagram

■ TIMER 2

In addition to the timer 1, the HD6303Y provides an 8-bit reloadable timer, which is capable of counting the external event. The timer 2 contains a timer output, so the MPU can generate three independent waveforms. (Refer to Fig. 23.)

The timer 2 is configured as follows:

- Control/Status Register 3 (7 bits)
- 8-bit Up Counter
- Time Constant Register (8 bits)

• Timer 2 Up Counter (T2CNT) ($001D)

This is an 8-bit up counter which operates with the clock decided by CKS0 and CKS1 of the TCSR3. The CPU can read the value of the counter without affecting the counter. In addition, any value can be written to the counter by software even during counting.

The counter is cleared when a match occurs between the counter and the TCONR or during reset.

If the write operation is made by software to the counter at the cycle of counter clear, it does not reset the counter but put the write data to the counter.

• Time Constant Register (TCONR) ($001C)

The time constant register is an 8-bit write only register. The data of register is always compared with the counter.

When a match has occurred, the counter match flag (CMF) of the timer control status register 3 (TCSR3) is set and the value selected by TOS0 and TOS1 of the TCSR3 will appear at port 2, bit 6. When CMF is set, the counter will be cleared simultaneously and then start counting from $00. This enables regular interrupts and waveform outputs without any software support. The TCONR is set to "$FF" during reset.

• Timer Control/Status Register 3 (TCSR3) ($001B)

The timer control/status register 3 is a 7-bit register. All bits are readable and 6 bits except for CMF can be written.

The followings are each pin descriptions.

Timer Control/Status Register 3

7	6	5	4	3	2	1	0	
CMF	ECMI	–	T2E	TOS1	TOS0	CKS1	CKS0	$001B

Bit 0 CKS0 Input Clock Select 0
Bit 1 CKS1 Input Clock Select 1

Input clock to the counter is selected as shown in Table 5 depending on these two bits. When an external clock is selected, bit 7 of port 2 will be a clock input automatically. Timer 2 detects the rising edge of the external clock and increments the counter. The external clock is countable up to half the frequency of the system clock.

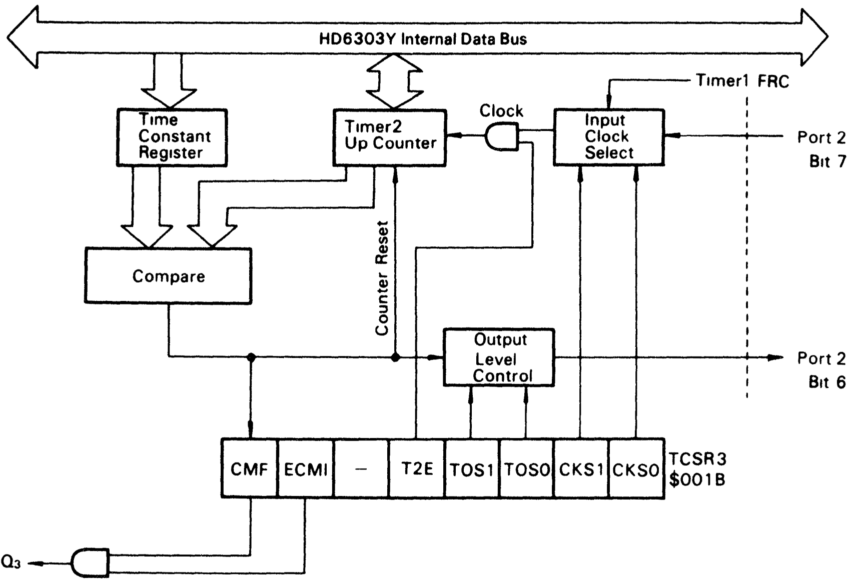


Figure 23 Timer 2 Block Diagram

Table 5 Input Clock Select

CKS1	CKSO	Input Clock to the Counter
0	0	E clock
0	1	E clock/8*
1	0	E clock/128*
1	1	External clock

* These clocks come from the FRC of the timer 1 If one of these clocks is selected as an input clock to the up counter, the CPU should not write to the FRC of the timer 1

Bit 2 TOSO Timer Output Select 0
Bit 3 TOS1 Timer Output Select 1

When a match occurs between the counter and the TCONR timer 2 outputs shown in Table 6 will appear at port 2, bit 6 depending on these two bits. When both TOS0 and TOS1 are "0", bit 6 of port 2 will be an I/O port.

Table 6 Timer 2 Output Select

TOS1	TOSO	Timer Output
0	0	Timer Output Inhibited
0	1	Toggle Output*
1	0	Output "0"
1	1	Output "1"

* When a match occurs between the counter and the TCONR, timer 2 output level is reversed This leads to production of a square wave with 50% duty to the external without any software support

Bit 4 T2E Timer 2 Enable Bit

When this bit is cleared, a clock input to the up counter is inhibited and the up counter stops When set to "1", a clock selected by CKS1 and CKS0 (Table 5) is input to the up counter.

(Note) P_{26} outputs "0" when T2E bit cleared and timer 2 set in output enable condition by TOS1 or TOS0. It also outputs "0" when T2E bit set "1" and timer 2 set in output enable condition before the first counter match occurs.

Bit 5 Not Used.

Bit 6 ECMI Enable Counter Match Interrupt

When this bit is set, an internal interrupt (IRQ_3) by CMI is enabled. When cleared, the interrupt is inhibited.

Bit 7 CMF Counter Match Flag

This read-only bit is set when a match occurs between the up counter and the TCONR. Cleared by writing "0" at CMF=1 by software (unable to write "1" by software).

Each bit of the TCSR3 is cleared during reset.

■ SERIAL COMMUNICATION INTERFACE (SCI)

The Serial Communication Interface (SCI) in the HD6303Y contains the following two operating modes: asynchronous mode by the NRZ format, and clocked synchronous mode which transfers data synchronously with the clock. In the asynchronous mode, data length, parity bits and number of stop bits can be selected, and eight transfer formats are provided.

The SCI consists of the following registers as shown in Fig. 24 Block Diagram.

- Transmit/Receive Control Status Register 1 (TRCSR1)
- Rate/Mode Control Register (RMCR)
- Transmit/Receive Control Status Register 2 (TRCSR2)
- Receive Data Register (RDR)
- Recevie Shift Register
- Transmit Data Register (TDR)
- Transmit Shift Register

To operate the SCI, initialize the RMCR and TRCSR2, after selecting the desirable operating mode and transfer format. Next, set the enable bit (TE or RE) of the TRCSR1. Operating mode and transfer format should be changed when the enable bit (TE, RE) is cleared. When setting the TE or RE again after changing the operating mode or transfer format, interval of more than a 1-bit cycle of the baud rate or bit rate is necessary. If a 1-bit cycle or more is not allowed, the SCI block may not be initialized.

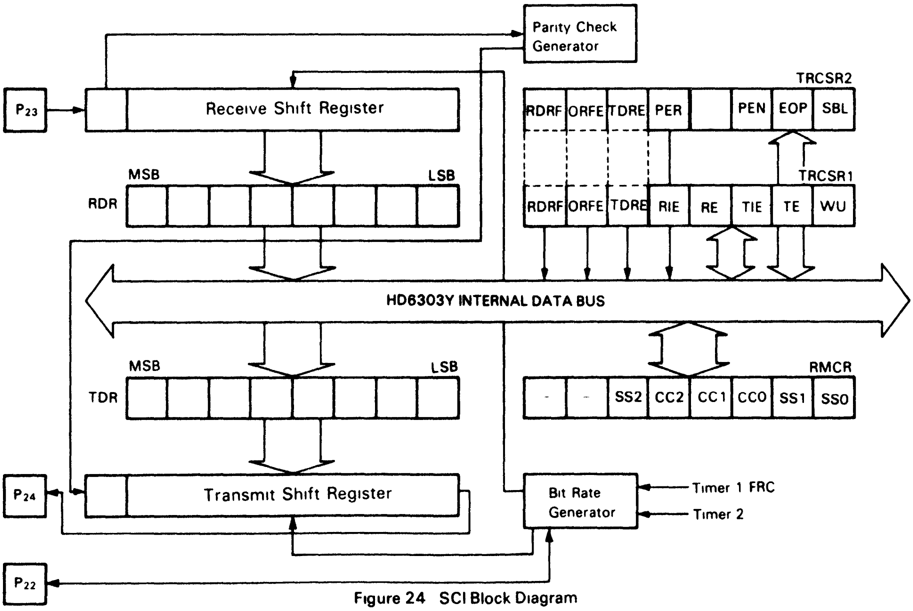


Figure 24 SCI Block Diagram

- **Asynchronous Mode**

Asynchronous mode contains 8 transfer formats as shown in Fig. 25.

Data transmission is enabled by setting TE bit of the TRCSR1, then port 2, bit 4 will unconditionally become a serial output independently of the corresponding DDR

To transmit data, set the desirable transmit format with RMCR and TRCSR2 When the TE bit is set, the data can be transmitted after transmitting the one frame of preamble ("1").

The conditions at this stage are as follows.

1) If the TDR is empty (TDRE=1), consecutive 1's are produced to indicate the idle state.
2) If the TDR contains data (TDRE=0), data is sent to the Transmit Shift Register and data transmit starts.

During data transmit, a start bit of "0" is transmitted first. Then 7-bit or 8-bit data (starts from bit 0) is transmitted. With PEN=1, the parity bit, even or odd, selected by EOP bit is added, lastly the stop bit (1 bit or 2 bis) is sent.

When the TDR is "empty", hardware sets TDRE flag bit. If the CPU doesn't respond to the flag in proper timing (the TDRE is in set condition till the next normal data transfer starts from the transmit data register to the transmit sift register), "1" is transferred instead of the start bit "0" and continues to be transferred till data is provided to the data register While the TDRE is "1", "0" is not transferred.

Data receive is possible by setting RE bit This makes port 2, bit 3 a serial input. The operation mode of data receive is decided by the contents of the TRCSR2 and RMCR at first, and set RE bit of TRCSR1. The first "0" (space) synchronizes the receive bit flow. Each bit of the following data will be strobed in the middle. If a stop bit is not "1", a framing error assumed and ORFE is set

When a framing error occurs, receive data is transferred to the Receive Data Register and the CPU can read the error-generating data. This makes it possible to detect a line break.

When PEN bit is set, the parity check is done. If the parity bit does not match the EOP bit, a parity error occurs and the PER bit is set, not the RDRF bit. Also, when the parity error occurs the receive data can be read just like in the case of the framing error.

The RDRF flag is set when the data is received without a framing error and a parity error.

If RDRF is still set when receiving the stop bit of the next data, ORFE is set to indicate the overrun generation. CPU can get the receive data by reading RDR. When 7 bit data format is selected, the 8th bit of RDR is "0".

When the CPU read the receive Data Register as a response to RDRF flag or ORFE flag after having read TRCSR, RDRF or ORFE is cleared.

(Note) Clock Source in Asynchronous Mode

If CC1:CC0=10, the internal bit rate clock is provided at P_{22} regardless of the values for TE or RE. Maximum clock rate is E÷16.

If both CC1 and CC0 are set, an external TTL compatible clock must be connected to P_{22} at sixteen times (16×) the desired bit rate, but not greater than E.

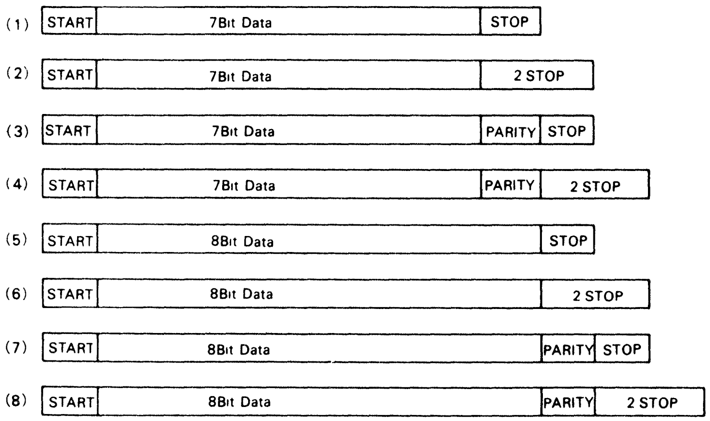

(1) START 7Bit Data STOP
(2) START 7Bit Data 2 STOP
(3) START 7Bit Data PARITY STOP
(4) START 7Bit Data PARITY 2 STOP
(5) START 8Bit Data STOP
(6) START 8Bit Data 2 STOP
(7) START 8Bit Data PARITY STOP
(8) START 8Bit Data PARITY 2 STOP

Figure 25 Asynchronous Mode Transfer Format

- **Clocked Synchronous Mode**

In the clocked synchronous mode, data transmit is synchronized with the clock pulse. The HD6303Y SCI provides functionally independent transmitter and receiver which makes full duplex operation possible in the asynchronous mode. But in the clocked synchronous mode an SCI clock I/O pin is only P_{22}, so the simultaneous receive and transmit operation is not available. In this mode, TE and RE should not be in set condition ("1") simultaneously. Fig 26 gives a synchronous clock and a data format in the clocked synchronous mode

1) Data transmit

Data transmit is realized by setting TE bit in the TRCSR1 Port 2, bit 4 becomes an output unconditionally independent of the value of the corresponding DDR

Both the RMCR and TRCSR should be set in the desirable operating condition for data transmit.

When an external clock input is selected and the TDRE flag is "0", data transmit is performed from port 2, bit 4, synchronizing with 8 clock pulses input from external to port 2, bit 2.

Data is transmitted from bit 0 and the TDRE is set when the Transmit Shift Register (TSR) is "empty". More than 9th clock pulse of external are ignored

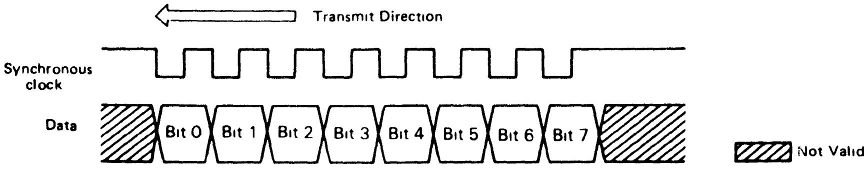
When data transmit is selected to the clock output, the MPU produces transmit data and synchronous clock at TDRE flag clear.

2) Data receive

Data receive is enabled by setting RE bit. Port 2, bit 3 will be a serial input. The operating mode of data receive is decided by the TRCSR1 and the RMCR.

If the external clock input is selected, 8 external clock pulses and the synchronized receive data are input to port 2, bit 2 and bit 3 respectively. The MPU put receive data into the receive data shift register by this clock and set the RDRF flag at the termination of 8 bit data receive. More than 9th clock pulse of external input are ignored. When RDRF is cleared, the MPU starts receiving the next data instantly. So, RDRF should be cleared with P_{22} "High".

When data receive is selected with the clock output, 8 synchronous clocks are output to the external by setting RE bit So receive data should be input from external synchronously with this clock When the first byte data is received, the RDRF flag is set. After the second byte, receive operation is performed by sending the synchronous clock to the external after clearing the RDRF bit.



- Transmit data is produced from a falling edge of a synchronous clock to the next falling edge
- Receive data is latched at the rising edge

Figure 26 Clocked Synchronous Mode Format

- **Transmit/Receive Control Status Register (TRCSR1) ($0011)**

The TRCSR1 is composed of 8 bits which are all readable Bits 0 to 4 are also writable. This register is initialized to $20 during reset. Each bit functions are as follows.

Transmit/Receive Control Status Register

7	6	5	4	3	2	1	0	
RDRF	ORFE	TDRE	RIE	RE	TIE	TE	WU	$0011

Bit 0 WU Wake-up

In a typical multi-processor configuration, the software protocol provides the destination address at the first byte of the message. In order to make uninterested MPU ignore the remaining message, a wake-up function is available By this, uninterested MPU can inhibit all further receive processing till the next message starts.

Then wake-up function is triggered by consecutive 1's with 1 frame length The software protocol should provide the idle time between messages.

By setting this bit, the MPU stops data receive till the next message The receive of consecutive "1" with one frame length wakes up and clears this bit by hardware and then the MPU restarts receive operation. However, the RE flag should be already set before setting this bit. In the clocked synchronous mode WU is not available, so this bit should not be set.

Bit 1 TE Transmit Enable

When this bit is set, transmit data will appear at port 2, bit 4 after one frame preamble in asynchronous mode, while in clocked synchronous mode it appears immediately. This is executed regardless of the value of the corresponding DDR When TE is cleared, the serial I/O doesn't affect port 2, bit 4

Bit 2 TIE Transmit Interrupt Enable

When this bit is set, an internal interrupt (IRQ_3) is enabled when TDRE (bit 5) is set. When cleared, the interrupt is inhibited.

Bit 3 RE Receive Enable

When set, a signal is input to the receiver from port 2, bit 3 regardless of the value of the DDR. When RE is cleared, the serial I/O doesn't afffect port 2, bit 3.

Bit 4 RIE Receive Interrupt Enable

When this bit is set, an internal interrupt (IRQ_3) is enabled when RDRF (bit 7) or ORFE (bit 6) is set When cleared, the interrupt is inhibited.

Bit 5 TDRE Transmit Data Register Empty

TDRE is set by hardware when the TDR is transferred to the Transmit Shift Register in the asynchronous mode, while in clocked synchronous mode when the TDSR is "empty". This bit is cleared by reading the TRCSR1 or TRCSR2 and writing new transmit data to the TDR when TDRE=1 TDRE is set to "1" during reset.

Bit 6 ORFE Overrun Framing Error

ORFE is set by hardware when an overrun or a framing error is generated (during data-receive only). An overrun error occurs when new receive data is ready to be transferred to the RDR during RDRF still being set. A framing error occurs when a stop bit is "0". But in clocked synchronous mode, this bit is not affected. This bit is cleared by reading the TRCSR1 or TRCSR2, and the RDR, when RDRF=1 ORFE is cleared during reset.

Bit 7 RDRF Receive Data Register Full

RDRF is set by hardware when data is received normally and transferred from the Receive Shift Register (RSR) to the RDR This bit is cleared by reading TRCSR1 or TRCSR2, and the RDR, when RDRF=1. This bit is cleared during reset.

- **Transmit Rate/Mode Control Register (RMCR)**

The RMCR controls the following serial I/O

· Baud Rate	· Data Format
· Clock source	· Port 2, Bit 2 Function
· Operation Mode	

All bits are readable/writable. Bit 0 to 5 of the RMCR are cleared during reset.

Transfer Rate/Mode Control Register

7	6	5	4	3	2	1	0	
–	–	SS2	CC2	CC1	CCO	SS1	SSO	$0010

Bit 0 SSO
Bit 1 SS1 Speed Select
Bit 5 SS2

These bits control the baud rate used for the SCI. Table 7 lists the available baud rates The timer 1 FRC (SS2=0) and the timer 2 up counter (SS2=1) provide the internal clock to the SCI. When selecting the timer 2 as a baud rate clock source, it functions as a baud rate generator The timer 2 generates the baud rate listed in Table 8 depending on the value of the TCONR.

(Note) When operating the SCI with internal clock, do not perform write operation to the timer/counter which is the

Table 7 SCI Bit Tımes and Transfer Rates

(1) Asynchronous Mode

			XTAL	2.4576MHz	4 0MHz	4.9152MHz
SS2	SS1	SS0	E	614 4kHz	1 0MHz	1.2288MHz
0	0	0	E−16	26 ıs/38400Baud	16μs/62500Baud	13μs/76800Baud
0	0	1	E−128	208μs/4800Baud	128μs/7812 5Baud	104 2 μs/9600Baud
0	1	0	E−1024	1 67ms/600Baud	1 024ms/976 6Baud	833 3μs/1200Baud
0	1	1	E−4096	6 67ms/150Baud	4 096ms/244 1Baud	3 333ms/300Baud
1	—	—	—	*	*	*

* When SS2 ıs "1", Tımer 2 provıdes SCI clocks Fhe baud rate ıs shown as follows wıth the TCONR as N

$$\text{Baud Rate} = \frac{f}{32\,(N+1)} \qquad \left(\begin{array}{l} f \quad \text{input clock frequency to the timer 2 counter} \\ N = 0 \sim 255 \end{array}\right)$$

(2) Clocked Synchronous Mode*

			XTAL	4 0 MHz	6 0 MHz	8 0 MHz	12 0 MHz
SS2	SS1	SS0	E	1 0 MHz	1 5 MHz	2 0 MHz	3 0 MHz
0	0	0	E − 2	2 μs/bıt	1 33 μs/bıt	1 μs/bıt	0 667 μs/bıt
0	0	1	E − 16	16 μs/bıt	10 7 μs/bıt	8 μs/bıt	5 33 μs/bıt
0	1	0	E − 128	128 μs/bıt	85 3 μs/bıt	64 μs/bıt	42 7 μs/bıt
0	1	1	E − 512	512 μs/bıt	341 μs/bıt	256 μs/bıt	171 μs/bıt
1	—	—	—	**	**	**	**

*Bıt rates ın the case of ınternal clock operatıon In the case of external clock operatıon, the external clock ıs operatable up to DC ~ 1/2 system clock

** The bıt rate ıs shown as follows wıth the TCONR as N

$$\text{Bit Rate } (\mu s/\text{bit}) = \frac{4\,(N+1)}{f} \qquad \left(\begin{array}{l} f \quad \text{input clock frequency to the timer 2 counter} \\ N = 0 \sim 255 \end{array}\right)$$

Table 8 Baud Rate and Tıme Constant Regıster Example

XTAL / Baud Rate (Baud)	2 4576MHz	3 6864MHz	4 0MHz	4 9152MHz	8 0MHz
110	21*	32*	35*	43*	70*
150	127	191	207	255	51*
300	63	95	103	127	207
600	31	47	51	63	103
1200	15	23	25	31	51
2400	7	11	12	15	25
4800	3	5	—	7	12
9600	1	2	—	3	—
19200	0	—	—	1	—
38400	—	—	—	0	—

* E/8 clock ıs ınput to the tımer 2 up counter and E clock otherwıse

Table 9 SCI Format and Clock Source Control

CC2	CC1	CC0	Format	Mode	Clock Source	Port 2, Bıt 2	Port 2, Bıt 3	Port 2, Bıt 4
0	0	0	8 bıt data	Clocked Synchronous	External	Input	When the TRCSR1, RE bıt ıs "1", bıt 3 ıs used as a serıal ınput	When the TRCSR1, TE bıt ıs "1", bıt 4 ıs used as a serıal output
0	0	1	8-bıt data	Asynchronous	Internal	Not Used**		
0	1	0	8-bıt data	Asynchronous	Internal	Output*		
0	1	1	8-bıt data	Asynchronous	External	Input		
1	0	0	8-bıt data	Clocked Synchronous	Internal	Output		
1	0	1	7-bıt data	Asynchronous	Internal	Not Used**		
1	1	0	7-bıt data	Asynchronous	Internal	Output*		
1	1	1	7-bıt data	Asynchronous	External	Input		

* Clock output regardless of the TRCSR1, bit RF and TI
** Not used for the SCI

clock source of the SCI.

Bit 2 CC0
Bit 3 CC1 Clock Control/Format Select*
Bit 4 CC2

These bits control the data format and the clock source (refer to Table 9).

* CC0, CC1 and CC2 are cleared during reset and the MPU goes to the clocked synchronous mode of the external clock operation. Then the MPU automatically set port 2, bit 2 into the clock input state. When using port 2, bit 2 as an output port, the DDR of port 2 should be set to "1" and CC1 and CC0 to "0" and "1" respectively.

Bit 6 Not Used
Bit 7 Not Used

- **Transmit/Receive Control Status Register 2 (TRCSR2)**

The TRCSR2 is a 7-bit register which can select a data format in the asynchronous mode The upper 3 bits are the same address as the TRCSR1. Therefore, the RDRF, ORFE and TDRE can be read by either the TRCSR1 or TRCSR2. Bits 0 to 2 of the TRCSR2 are used for read/write. Bits 4 to 7 are used only for read.

Transmit/Receive Control Status Register 2

7	6	5	4	3	2	1	0	
RDRF	ORFE	TDRE	PER	–	PEN	EOP	SBL	$001E

Bit 0 SBL Stop Bit Length
This bit selects the stop bit length in the asynchronous mode. If this bit is "0", the stop bit is 1-bit. If "1", the stop bit is 2-bit. This bit is cleared during reset.

Bit 1 EOP Even/Odd Parity
This bit selects the parity generated and checked when the PEN is "1". If this bit is "0", the parity is even. If "1", it is odd. This bit is cleared during reset.

Bit 2 PEN Parity Enable
This bit decides whether the parity bit should be generated and checked in the asynchronous mode or not. If this bit is "0", the parity bit is neither generated nor checked If "1", it is generated and checked. This bit is cleared during reset

The 3 bits above do not affect the SCI opertion in the clocked synchronous mode.

Bit 3 Not Used

Bit 4 PER Parity Error
This bit is set when the PEN is "1" and a parity error occurs. It is cleared by reading the RDR after reading the TRCSR2, when PER=1

Bit 5 TDRE
Transmit Data Register Empty

Bit 6 ORFE
Overrun/Framing Error

Bit 7 RDRF
Receive Data Register Full

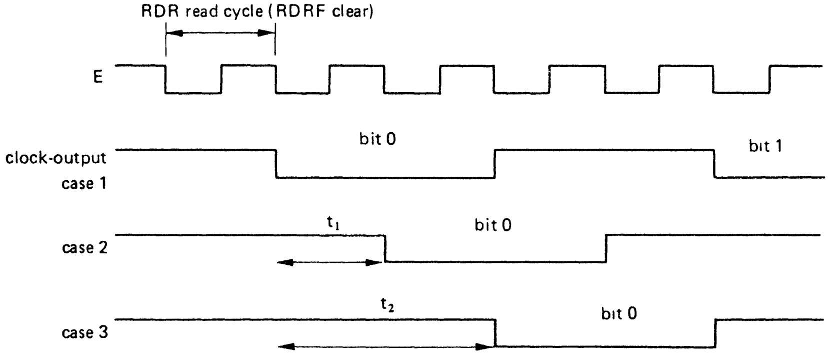
* Each flag of the TDRE, ORFE, and RDRF can be read from either the TRCSR1 or TRCSR2

■ PRECAUTION 1

In the synchronous clocked receive operation with clock-output, there are three cases for clock pulse timing after RDRF clear as shown below.

Please consider above in designing system, since transmitting receiving time is not uniform.

The clock-output of case 1 or case 2 is determined by "1" or "0" of SCI internal operation clock of RDRF clearing cycle. In addition, in the case of low voltage operation ($V_{CC} < 4.5V$), the clock-output of case 1 may transfer to case 3.



(note) When bit rate is E/2, t_1 = E, and t_2 = 2E.
E/16, t_1 = 8E, t_2 = 16E.
E/128, t_1 = 64E, t_2 = 128E.
E/512, t_1 = 256E, t_2 = 512E.

Diagram for Precaution 1

■ PRECAUTION 2

When transmitting through clock-synchronous serial communication interface, TE bit should not be cleared with TDRE of TRCSR ($11) is "0".

The TDRE set and clear conditions of SCI are shown as follows.

	Set condition	Clear condition
TDRE	1. TDR → transmit shift register (asynchronous) 2. Transmit shift register is empty. (clock-synchronous) 3. $\overline{\text{RES}}$ = 0	When writing to TDR after TRSCR read, with TDRE = 1, TDRE is cleared.

If transmit data is written to TDR, and then TE bit is cleared with TDRE = 0 to stop transmitting, TDRE remains "0".

In this case, even if TE bit is set and transmit data is written again, the TDR data is not transmitted.

Please note that TE bit must be cleared after the last data has been transmitted.

(This caution is not applied to asynchronous serial communication interface.)

■ TIMER, SCI STATUS FLAG

Table 10 shows the set and reset conditions of each status flag in the timer 1, timer 2 and SCI.

Table 10 Timer 1, Timer 2 and SCI Status Flag

		Set Condition	Clear Condition
P6CSR	IS FLAG	Falling edge input to P_{54} ($\overline{\text{IS}}$)	1 Read the P6CSR then read or write the PORT6, when IS FLAG = 1 2. $\overline{\text{RES}}$ = 0
Timer 1	ICF	FRC → ICR by Rising or Falling edge input to P_{20} (Selecting with the IEDG bit)	1 Read the TCSR1 or TCSR2 then ICRH, when ICF = 1 2 $\overline{\text{RES}}$ = 0
	OCF1	OCR1 = FRC	1 Read the TCSR1 or TCSR2 then write to the OCR1H or OCR1L, when OCF1 = 1 2 $\overline{\text{RES}}$ = 0
	OCF2	OCR2 = FRC	1 Read the TCSR2 then write to the OCR2H or OCR2L, when OCF2 = 1 2 $\overline{\text{RES}}$ = 0
	TOF	FRC = $FFFF+1 cycle	1. Read the TCSR1 then FRCH, when TOF = 1 2. $\overline{\text{RES}}$ = 0
Timer 2	CMF	T2CNT = TCONR	1. Write "0" to CMF, when CMF = 1 2 $\overline{\text{RES}}$ = 0
SCI	RDRF	Receive Shift Register → RDR	1. Read the TRCSR1 or TRCSR2 then RDR, when RDRF = 1 2 $\overline{\text{RES}}$ = 0
	ORFE	1 Framing Error (Asynchronous Mode) Stop Bit = 0 2 Overrun Error (Asynchronous Mode) Receive Shift Register → RDR when RDRF = 1	1 Read the TRCSR1 or TRCSR2 then RDR, when ORFE = 1 2 . $\overline{\text{RES}}$ = 0
	TDRE	1 Asynchronous Mode TDR → Transmit Shift Register 2 Clocked Synchronous Mode Transmit Shift Register is "empty" 3. $\overline{\text{RES}}$ = 0	Read the TRCSR1 or TRCSR2 then write to the TDR, when TDRE = 1
	PER	Parity when PEN=1	1. Read the TRCSR2 then RDR, when PER=1 2. $\overline{\text{RES}}$=0

(Note) → , Transfer = , equal

ICRH, Upper byte of ICR
OCR1H, Upper byte of OCR1
OCR2H, Upper byte of OCR2

OCR1L, Lower byte of OCR1
OCR2L; Lower byte of OCR2
FRCH, Upper byte of FRC

■ LOW POWER DISSIPATION MODE

The HD6303Y provides two low power dissipation modes; sleep and standby.

• Sleep Mode

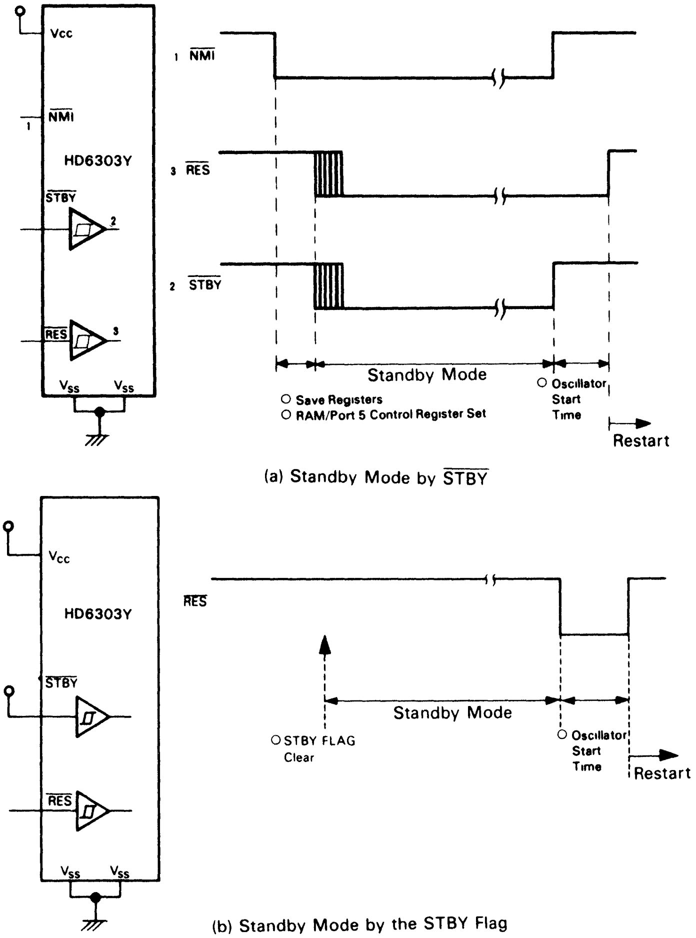
The MPU goes to the sleep mode by SLP instruction execution. In the sleep mode, the CPU stops its operation, while the registers' contents are retained. In this mode, the peripherals except the CPU such as timers, SCI, etc. continue their functions. The power dissipation of sleep-condition is one fourth that of operating condition.

The MPU returns from this mode by an interrupt, $\overline{RES}$ or $\overline{STBY}$; it goes to the reset state by $\overline{RES}$ and the standby mode by $\overline{STBY}$. When the CPU acknowledges an interrupt request, it cancels the sleep mode, returns to the operation mode and branches to the interrupt routine. When the CPU masks this interrupt, it cancels the sleep mode and executes the next instruction. However, for example, if the timer 1 or 2 prohibits a timer interrupt, the CPU doesn't cancel the sleep mode because of no interrupt request.

This sleep mode is effective to reduce the power dissipation for a system with no need of the HD6303Y's consecutive operation.

• Standby Mode

The MPU goes to the standby mode with the $\overline{STBY}$ "Low" or by clearing the STBY flag. In this mode, the HD6303Y stops all the clocks and goes to the reset state. In this mode, the power dissipation is reduced to several μA. During standby, all pins, except the power supply (V_{CC}, V_{SS}), the $\overline{STBY}$, $\overline{RES}$ and XTAL (which outputs "0"), go to the high impedance state In this mode, power (V_{CC}) is supplied to the HD6303Y, and the contents of RAM is retained. The MPU returns from this mode during reset. When the MPU goes to the standby mode with $\overline{STBY}$ "Low", it will restart at the timing shown in Fig. 27(a). When the MPU goes to the standby mode by clearing the STBY flag, it will restart only by keeping the $\overline{RES}$ "Low" for longer than the oscillating stabilization time. (Fig. 27(b))



(a) Standby Mode by $\overline{STBY}$

(b) Standby Mode by the STBY Flag

Figure 27 Standby Mode Timing

■ TRAP FUNCTION

The CPU generates an interrupt with the highest priority (TRAP) when fetching an undefined instruction or an instruction from non-memory space. The TRAP prevents the system-burst caused by noise or a program error.

• Op Code Error

When fetching an undefined op code, the CPU saves registers as well as a normal interrupt and branches to the TRAP ($FFEE, $FFEF). This has the priority next to reset.

• Address Error

When an instruction fetch is made from the address of internal register, the MPU generaters an interrupt as well as an op code error. But on the system with no memory in its external memory area, this function is not applicable if an instruction fetch is made from the external non-memory area. Addresses where an address error occurs are from $0000 to $0027.

This function is available only for an instruction fetch and is not applicable to the access of normal data read/write.

(Note) The TRAP interrupt provides a retry function differently from other interrupts This is a program flow return to the address where the TRAP occurs when a sequence returns to a main routine from the TRAP interrupt routine by RTI. The retry can prevent the system burst caused by noise, etc.

However, if another TRAP occurs, the program repeats the TRAP interrupt forever, so the consideration is necessary in programming.

■ INSTRUCTION SET

The HD6303Y provides object code upward compatible with the HD6801 to utilize all instruction set of the HMCS6800. It also reduces the execution times of key instructions for throughput improvement.

Bit manipulation instruction, change instruction of the index register and accumulator and sleep instruction are also added

The followings are explained here.

- CPU Programming Model (refer to Fig. 28)
- Addressing Mode
- Accumulator and Memory Manipulation Instruction (refer to Table 11)
- New Instruction
- Index Register and Stack Manipulation Instruction (refer to Table 12)
- Jump and Branch Instruction (refer to Table 13)
- Condition Code Register Manipulation (refer to Table 14)
- Op Code Map (refer to Table 15)

• Programming Model

Fig 28 depicts the HD6303Y programming model The double accumulator D consists of accumulator A and B, so when using the accumulator D, the contents of A and B are destroyed.

• CPU Addressing Mode

The HD6303Y provides 7 addressing modes. The addressing mode is determined by an instruction type and code. Tables 11 through 15 show addressing modes of each instruction with the execution times counted by the machine cycle

When the clock frequency is 4MHz, the machine cycle time becomes microseconds directly

Accumulator (ACCX) Addressing

Only an accumulator is addressed and the accumulator A or B is selected. This is a one-byte instruction.

Immediate Addressing

This addressing locates a data in the second byte of an instruction. However, LDS and LDX locate a data in the second and third byte exceptionally. This addressing is a 2 or 3-byte instruction.

Direct Addressing

In this addressing mode, the second byte of an instruction shows the address where a data is stored. 256 bytes ($0 through $255) can be addressed directly. Execution times can be reduced by storing data in this area so it is recommended to make it RAM for users' data area in configurating a system. This is a 2-byte instruction, while 3 byte with regard to AIM, OIM, EIM and TIM.

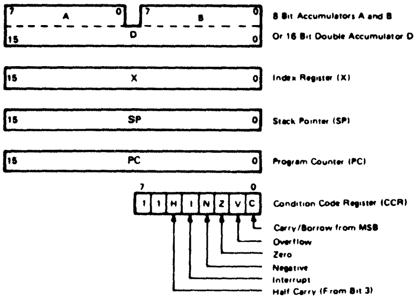


Figure 28 CPU Programming Model

Extended Addressing

In this mode, the second byte shows the upper 8 bit of the data stored address and the third byte the lower 8 bit. This indicates the absolute address of 3 byte instruction in the memory.

Indexed Addressing

The second byte of an instruction and the lower 8 bit of the index register are added in this mode As for AIM, OIM, EIM and TIM, the third byte of an instruction and the lower 8 bits of the index register are added.

This carry is added to the upper 8 bit of the index register and the result is used for addressing the memory. The modified address is retained in the temporary address register, so the contents of the index register doesn't change. This is a 2-byte instruction except AIM, OIM, EIM and TIM (3-byte instruction)

Implied Addressing

An instruction itself specifies the address This is, the instruction addresses a stack pointer, index register, etc. This is a one-byte instruction.

Relative Addressing

The second byte of an instruction and the lower 8 bits of the program counter are added. The carry or borrow is added to the upper 8 bit. So addressing from − 126 to + 129 byte of the current instruction is enabled. This is a 2-byte instruction

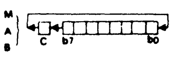
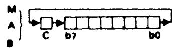
(Note) CLI, SEI Instructions and Interrupt Operation

When accepting the IRQ at a preset timing with CLI and SEI instructions, more than 2 cycles are necessary between the CLI and SEI instructions. For example, the following program (a)(b) don't accept the IRQ but (c) accepts it

```
  .      .      .
  .      .      .
  .      .     CLI
 CLI    CLI    NOP
 SEI    NOP    NOP
  .     SEI    SEI
  .      .      .
  .      .      .
  .      .      .
 (a)    (b)    (c)
```

The same thing can be said to the TAP instruction instead of the CLI and SEI instructions

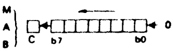
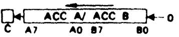
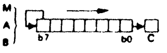
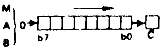
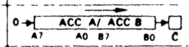
Table 11 Accumulator, Memory Manipulation Instructions

Operations	Mnemonic	Addressing Modes															Boolean/ Arithmetic Operation	Condition Code Register					
		IMMED			DIRECT			INDEX			EXTEND			IMPLIED				5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#		H	I	N	Z	V	C
Add	ADDA	8B	2	2	9B	3	2	AB	4	2	BB	4	3				A + M → A	↕	●	↕	↕	↕	↕
	ADDB	CB	2	2	DB	3	2	EB	4	2	FB	4	3				B + M → B	↕	●	↕	↕	↕	↕
Add Double	ADDD	C3	3	3	D3	4	2	E3	5	2	F3	5	3				A B + M M + 1 → A B	●	●	↕	↕	↕	↕
Add Accumulators	ABA													1B	1	1	A + B → A	↕	●	↕	↕	↕	↕
Add With Carry	ADCA	89	2	2	99	3	2	A9	4	2	B9	4	3				A + M + C → A	↕	●	↕	↕	↕	↕
	ADCB	C9	2	2	D9	3	2	E9	4	2	F9	4	3				B + M + C → B	↕	●	↕	↕	↕	↕
AND	ANDA	84	2	2	94	3	2	A4	4	2	B4	4	3				A·M → A	●	●	↕	↕	R	●
	ANDB	C4	2	2	D4	3	2	E4	4	2	F4	4	3				B·M → B	●	●	↕	↕	R	●
Bit Test	BIT A	85	2	2	95	3	2	A5	4	2	B5	4	3				A·M	●	●	↕	↕	R	●
	BIT B	C5	2	2	D5	3	2	E5	4	2	F5	4	3				B·M	●	●	↕	↕	R	●
Clear	CLR							6F	5	2	7F	5	3				00 → M	●	●	R	S	R	R
	CLRA													4F	1	1	00 → A	●	●	R	S	R	R
	CLRB													5F	1	1	00 → B	●	●	R	S	R	R
Compare	CMPA	81	2	2	91	3	2	A1	4	2	B1	4	3				A − M	●	●	↕	↕	↕	↕
	CMPB	C1	2	2	D1	3	2	E1	4	2	F1	4	3				B − M	●	●	↕	↕	↕	↕
Compare Accumulators	CBA													11	1	1	A − B	●	●	↕	↕	↕	↕
Complement, 1's	COM							63	6	2	73	6	3				$\overline{M}$ → M	●	●	↕	↕	R	S
	COMA													43	1	1	$\overline{A}$ → A	●	●	↕	↕	R	S
	COMB													53	1	1	$\overline{B}$ → B	●	●	↕	↕	R	S
Complement, 2's (Negate)	NEG							60	6	2	70	6	3				00 − M → M	●	●	↕	↕	①	②
	NEGA													40	1	1	00 − A → A	●	●	↕	↕	①	②
	NEGB													50	1	1	00 − B → B	●	●	↕	↕	①	②
Decimal Adjust, A	DAA													19	2	1	Converts binary add of BCD characters into BCD format	●	●	↕	↕	↕	③
Decrement	DEC							6A	6	2	7A	6	3				M − 1 → M	●	●	↕	↕	④	●
	DECA													4A	1	1	A − 1 → A	●	●	↕	↕	④	●
	DECB													5A	1	1	B − 1 → B	●	●	↕	↕	④	●
Exclusive OR	EORA	88	2	2	98	3	2	A8	4	2	B8	4	3				A ⊕ M → A	●	●	↕	↕	R	●
	EORB	C8	2	2	D8	3	2	E8	4	2	F8	4	3				B ⊕ M → B	●	●	↕	↕	R	●
Increment	INC							6C	6	2	7C	6	3				M + 1 → M	●	●	↕	↕	⑤	●
	INCA													4C	1	1	A + 1 → A	●	●	↕	↕	⑤	●
	INCB													5C	1	1	B + 1 → B	●	●	↕	↕	⑤	●
Load Accumulator	LDAA	86	2	2	96	3	2	A6	4	2	B6	4	3				M → A	●	●	↕	↕	R	●
	LDAB	C6	2	2	D6	3	2	E6	4	2	F6	4	3				M → B	●	●	↕	↕	R	●
Load Double Accumulator	LDD	CC	3	3	DC	4	2	EC	5	2	FC	5	3				M + 1 → B, M → A	●	●	↕	↕	R	●
Multiply Unsigned	MUL													3D	7	1	A x B → A B	●	●	●	●	●	⑪
OR, Inclusive	ORAA	8A	2	2	9A	3	2	AA	4	2	BA	4	3				A + M → A	●	●	↕	↕	R	●
	ORAB	CA	2	2	DA	3	2	EA	4	2	FA	4	3				B + M → B	●	●	↕	↕	R	●
Push Data	PSHA													36	4	1	A → Msp, SP − 1 → SP	●	●	●	●	●	●
	PSHB													37	4	1	B → Msp, SP − 1 → SP	●	●	●	●	●	●
Pull Data	PULA													32	3	1	SP + 1 → SP, Msp → A	●	●	●	●	●	●
	PULB													33	3	1	SP + 1 → SP, Msp → B	●	●	●	●	●	●
Rotate Left	ROL							69	6	2	79	6	3					●	●	↕	↕	⑥	↕
	ROLA													49	1	1		●	●	↕	↕	⑥	↕
	ROLB													59	1	1		●	●	↕	↕	⑥	↕
Rotate Right	ROR							66	6	2	76	6	3					●	●	↕	↕	⑥	↕
	RORA													46	1	1		●	●	↕	↕	⑥	↕
	RORB													56	1	1		●	●	↕	↕	⑥	↕

(Note) Condition Code Register will be explained in Note of Table 14

(continued)

Table 11 Accumulator, Memory Manipulation Instructions

Operations	Mnemonic	Addressing Modes															Boolean/ Arithmetic Operation	Condition Code Register					
		IMMED			DIRECT			INDEX			EXTEND			IMPLIED				5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#		H	I	N	Z	V	C
Shift Left Arithmetic	ASL							68	6	2	78	6	3					●	●	↕	↕	⑥	↕
	ASLA													48	1	1		●	●	↕	↕	⑥	↕
	ASLB													58	1	1		●	●	↕	↕	⑥	↕
Double Shift Left, Arithmetic	ASLD													05	1	1		●	●	↕	↕	⑥	↕
Shift Right Arithmetic	ASR							67	6	2	77	6	3					●	●	↕	↕	⑥	↕
	ASRA													47	1	1		●	●	↕	↕	⑥	↕
	ASRB													57	1	1		●	●	↕	↕	⑥	↕
Shift Right Logical	LSR							64	6	2	74	6	3					●	●	R	↕	⑥	↕
	LSRA													44	1	1		●	●	R	↕	⑥	↕
	LSRB													54	1	1		●	●	R	↕	⑥	↕
Double Shift Right Logical	LSRD													04	1	1		●	●	R	↕	⑥	↕
Store Accumulator	STAA				97	3	2	A7	4	2	B7	4	3				A → M	●	●	↕	↕	R	●
	STAB				D7	3	2	E7	4	2	F7	4	3				B → M	●	●	↕	↕	R	●
Store Double Accumulator	STD				DD	4	2	ED	5	2	FD	5	3				A → M B → M + 1	●	●	↕	↕	R	●
Subtract	SUBA	80	2	2	90	3	2	A0	4	2	B0	4	3				A - M → A	●	●	↕	↕	↕	↕
	SUBB	C0	2	2	D0	3	2	E0	4	2	F0	4	3				B - M → B	●	●	↕	↕	↕	↕
Double Subtract	SUBD	83	3	3	93	4	2	A3	5	2	B3	5	3				A B - M M + 1 → A B	●	●	↕	↕	↕	↕
Subtract Accumulators	SBA													10	1	1	A - B → A	●	●	↕	↕	↕	↕
Subtract With Carry	SBCA	82	2	2	92	3	2	A2	4	2	B2	4	3				A - M - C → A	●	●	↕	↕	↕	↕
	SBCB	C2	2	2	D2	3	2	E2	4	2	F2	4	3				B - M - C → B	●	●	↕	↕	↕	↕
Transfer Accumulators	TAB													16	1	1	A → B	●	●	↕	↕	R	●
	TBA													17	1	1	B → A	●	●	↕	↕	R	●
Test Zero or Minus	TST							6D	4	2	7D	4	3				M - 00	●	●	↕	↕	R	R
	TSTA													4D	1	1	A - 00	●	●	↕	↕	R	R
	TSTB													5D	1	1	B - 00	●	●	↕	↕	R	R
And Immediate	AIM				71	6	3	61	7	3							M IMM→M	●	●	↕	↕	R	●
OR Immediate	OIM				72	6	3	62	7	3							M+IMM →M	●	●	↕	↕	R	●
EOR Immediate	EIM				75	6	3	65	7	3							M⊕IMM →M	●	●	↕	↕	R	●
Test Immediate	TIM				7B	4	3	6B	5	3							M IMM	●	●	↕	↕	R	●

(Note) Condition Code Register will be explained in Note of Table 14.

- **Additional Instruction**

In addition to the HD6801 instruction set, the HD6303Y prepares the following new instructions

AIM .... (M)·(IMM) → (M)

Executes "AND" operation to immediate data and the memory contents and stores its result in the memory

OIM . .. . (M)+(IMM) → (M)

Executes "OR" operation to immediate data and the memory contents and stores its result in the memory.

EIM .... (M) ⊕ (IMM) → (M)

Executes "EOR" operation to immediate data and the memory contents and stores its result in the memory

TIM .... (M)·(IMM)

Executes "AND" operation to immediate data and changes the relative flag of the condition code register.

These are the 3-byte instructions; the first byte is op code, the second immediate data and the third address modifier.

XGDX .... (ACCD)↔(IX)

Exchanges the contents of accumulator and the index register.

SLP

Goes to the sleep mode. Refer to "LOW POWER DISSIPATION MODE" for more details of the sleep mode.

Table 12 Index Register, Stack Manipulation Instructions

Pointer Operations	Mnemonic	Addressing Modes															Boolean/ Arithmetic Operation	Condition Code Register					
		IMMED			DIRECT			INDEX			EXTEND			IMPLIED				5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#		H	I	N	Z	V	C
Compare Index Reg	CPX	8C	3	3	9C	4	2	AC	5	2	BC	5	3				X – M M + 1	●	●	↕	↕	↕	↕
Decrement Index Reg	DEX													09	1	1	X – 1 → X	●	●	●	↕	●	●
Decrement Stack Pntr	DES													34	1	1	SP – 1 → SP	●	●	●	●	●	●
Increment Index Reg	INX													08	1	1	X + 1 → X	●	●	●	↕	●	●
Increment Stack Pntr	INS													31	1	1	SP + 1 → SP	●	●	●	●	●	●
Load Index Reg	LDX	CE	3	3	DE	4	2	EE	5	2	FE	5	3				M → X_H, (M + 1) → X_L	●	●	⑦	↕	R	●
Load Stack Pntr	LDS	8E	3	3	9E	4	2	AE	5	2	BE	5	3				M → SP_H, (M+1) → SP_L	●	●	⑦	↕	R	●
Store Index Reg	STX				DF	4	2	EF	5	2	FF	5	3				X_H → M, X_L → (M + 1)	●	●	⑦	↕	R	●
Store Stack Pntr	STS				9F	4	2	AF	5	2	BF	5	3				SP_H → M, SP_L → (M + 1)	●	●	7	↕	R	●
Index Reg → Stack Pntr	TXS													35	1	1	X – 1 → SP	●	●	●	●	●	●
Stack Pntr → Index Reg	TSX													30	1	1	SP + 1 → X	●	●	●	●	●	●
Add	ABX													3A	1	1	B + X → X	●	●	●	●	●	●
Push Data	PSHX													3C	5	1	X_L → M_{sp}, SP – 1 → SP X_H → M_{sp}, SP – 1 → SP	●	●	●	●	●	●
Pull Data	PULX													38	4	1	SP + 1 → SP, M_{sp} → X_H SP + 1 → SP, M_{sp} → X_L	●	●	●	●	●	●
Exchange	XGDX													18	2	1	ACCD↔IX	●	●	●	●	●	●

(Note) Condition Code Register will be explained in Note of Table 14.

Table 13 Jump, Branch Instruction

Operations	Mnemonic	Addressing Modes															Branch Test	Condition Code Register					
		RELATIVE			DIRECT			INDEX			EXTEND			IMPLIED				5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#		H	I	N	Z	V	C
Branch Always	BRA	20	3	2													None	●	●	●	●	●	●
Branch Never	BRN	21	3	2													None	●	●	●	●	●	●
Branch If Carry Clear	BCC	24	3	2													C = 0	●	●	●	●	●	●
Branch If Carry Set	BCS	25	3	2													C = 1	●	●	●	●	●	●
Branch If = Zero	BEQ	27	3	2													Z = 1	●	●	●	●	●	●
Branch If ≥ Zero	BGE	2C	3	2													N ⊕ V = 0	●	●	●	●	●	●
Branch If > Zero	BGT	2E	3	2													Z + (N ⊕ V) = 0	●	●	●	●	●	●
Branch If Higher	BHI	22	3	2													C + Z = 0	●	●	●	●	●	●
Branch If ≤ Zero	BLE	2F	3	2													Z + (N ⊕ V) = 1	●	●	●	●	●	●
Branch If Lower Or Same	BLS	23	3	2													C + Z = 1	●	●	●	●	●	●
Branch If < Zero	BLT	2D	3	2													N ⊕ V = 1	●	●	●	●	●	●
Branch If Minus	BMI	2B	3	2													N = 1	●	●	●	●	●	●
Branch If Not Equal Zero	BNE	26	3	2													Z = 0	●	●	●	●	●	●
Branch If Overflow Clear	BVC	28	3	2													V = 0	●	●	●	●	●	●
Branch If Overflow Set	BVS	29	3	2													V = 1	●	●	●	●	●	●
Branch If Plus	BPL	2A	3	2													N = 0	●	●	●	●	●	●
Branch To Subroutine	BSR	8D	5	2														●	●	●	●	●	●
Jump	JMP							6E	3	2	7E	3	3					●	●	●	●	●	●
Jump To Subroutine	JSR				9D	5	2	AD	5	2	BD	6	3					●	●	●	●	●	●
No Operation	NOP													01	1	1	Advances Prog Cntr Only	●	●	●	●	●	●
Return From Interrupt	RTI													3B	10	1		⑩					
Return From Subroutine	RTS													39	5	1		●	●	●	●	●	●
Software Interrupt	SWI													3F	12	1		●	S	●	●	●	●
Wait for Interrupt*	WAI													3E	9	1		●	⑪	●	●	●	●
Sleep	SLP													1A	4	1		●	●	●	●	●	●

(Note) * WAI puts R/$\overline{W}$ high; Address Bus goes to FFFF; Data Bus goes to the three state.
Condition Code Register will be explained in Note of Table 14

Table 14 Condition Code Register Manipulation Instructions

Operations	Mnemonic	AddressingModes			Boolean Operation	Condition Code Register					
		IMPLIED				5	4	3	2	1	0
		OP	~	#		H	I	N	Z	V	C
Clear Carry	CLC	0C	1	1	0 → C	•	•	•	•	•	R
Clear Interrupt Mask	CLI	0E	1	1	0 → I	•	R	•	•	•	•
Clear Overflow	CLV	0A	1	1	0 → V	•	•	•	•	R	•
Set Carry	SEC	0D	1	1	1 → C	•	•	•	•	•	S
Set Interrupt Mask	SEI	0F	1	1	1 → I	•	S	•	•	•	•
Set Overflow	SEV	0B	1	1	1 → V	•	•	•	•	S	•
Accumulator A → CCR	TAP	06	1	1	A → CCR	⑩					
CCR → Accumulator A	TPA	07	1	1	CCR → A	•	•	•	•	•	•

LEGEND

OP	Operation Code (Hexadecimal)
~	Number of MCU Cycles
M_{SP}	Contents of memory location pointed by Stack Pointer
#	Number of Program Bytes
+	Arithmetic Plus
-	Arithmetic Minus
•	Boolean AND
+	Boolean Inclusive OR
⊕	Boolean Exclusive OR
$\overline{M}$	Complement of M
→	Transfer into
0	Bit = Zero
00	Byte = Zero

CONDITION CODE SYMBOLS

H	Half-carry from bit 3 to bit 4
I	Interrupt mask
N	Negative (sign bit)
Z	Zero (byte)
V	Overflow, 2's complement
C	Carry/Borrow from/to bit 7
R	Reset Always
S	Set Always
↕	Set if true after test or clear
•	Not Affected

(Note) Condition Code Register Notes. (Bit set if test is true and cleared otherwise)

①	(Bit V)	Test: Result = 10000000?
②	(Bit C)	Test. Result ≠ 00000000?
③	(Bit C)	Test: BCD Character of high-order byte greater than 10? (Not cleared if previously set)
④	(Bit V)	Test: Operand = 10000000 prior to execution?
⑤	(Bit V)	Test. Operand = 01111111 prior to execution?
⑥	(Bit V)	Test. Set equal to N⊕ C = 1 after the execution of instructions
⑦	(Bit N)	Test Result less than zero? (Bit 15=1)
⑧	(All Bit)	Load Condition Code Register from Stack
⑨	(Bit I)	Set when interrupt occurs If previously set, a Non-Maskable Interrupt is required to exit the wait state
⑩	(All Bit)	Set according to the contents of Accumulator A
⑪	(Bit C)	Result of Multiplication Bit 7=1? (ACCB)

Table 15 OP-Code Map

OP CODE						ACC A	ACC B	IND	EXT/DIR*	ACCA or SP				ACCB or X				
										IMM	DIR	IND	EXT	IMM	DIR	IND	EXT	
HI		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111	
LO		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0000	0	▨	SBA	BRA	TSX	NEG				SUB								0
0001	1	NOP	CBA	BRN	INS	▨	▨	AIM		CMP								1
0010	2	▨	▨	BHI	PULA	▨	▨	OIM		SBC								2
0011	3	▨	▨	BLS	PULB	COM				SUBD				ADDD				3
0100	4	LSRD	▨	BCC	DES	LSR				AND								4
0101	5	ASLD	▨	BCS	TXS	▨	▨	EIM		BIT								5
0110	6	TAP	TAB	BNE	PSHA	ROR				LDA								6
0111	7	TPA	TBA	BEQ	PSHB	ASR				▨	STA			▨	STA			7
1000	8	INX	XGDX	BVC	PULX	ASL				EOR								8
1001	9	DEX	DAA	BVS	RTS	ROL				ADC								9
1010	A	CLV	SLP	BPL	ABX	DEC				ORA								A
1011	B	SEV	ABA	BMI	RTI	▨	▨	TIM		ADD								B
1100	C	CLC	▨	BGE	PSHX	INC				CPX				LDD				C
1101	D	SEC	▨	BLT	MUL	TST				BSR	JSR			▨	STD			D
1110	E	CLI	▨	BGT	WAI	▨	▨	JMP		LDS				LDX				E
1111	F	SEI	▨	BLE	SWI	CLR				▨	STS			▨	STX			F
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	

UNDEFINED OP CODE ▨

* Only each instructions of AIM, OIM, EIM, TIM

■ CPU OPERATION

• CPU Instruction Flow

When operating, the CPU fetches an instrution from a memory and executes the required function. This sequence starts with $\overline{RES}$ cancel and repeats itself limitlessly if not affected by a special instruction or a control signal. SWI, RTI, WAI and SLP instructions change this operation, while $\overline{NMI}$, $\overline{IRQ_1}$, $\overline{IRQ_2}$, IRQ_3, $\overline{HALT}$ and $\overline{STBY}$ control it. Fig. 29 gives the CPU mode transition and Fig. 30 the CPU system flow chart. Table 16 shows CPU operating states and port states.

• Operation at Each Instruction Cycle

Table 17 shows the operation at each instruction cycle. By the pipeline control of the HD6303Y, MULT, PUL, DAA and XGDX instructions, etc. prefetch the next instruction. So attention is necessary to the counting of the instruction cycles because it is different from the usual one—from op code fetch to the next instruction op code.

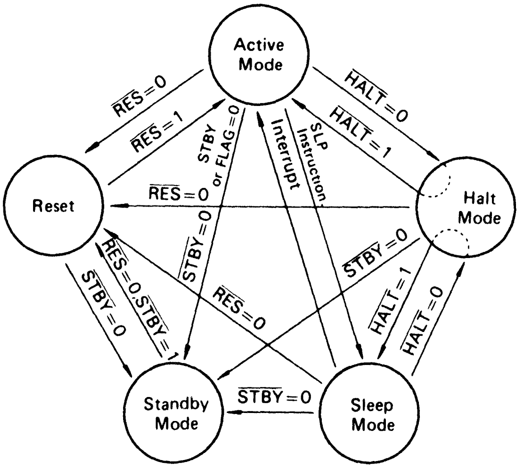
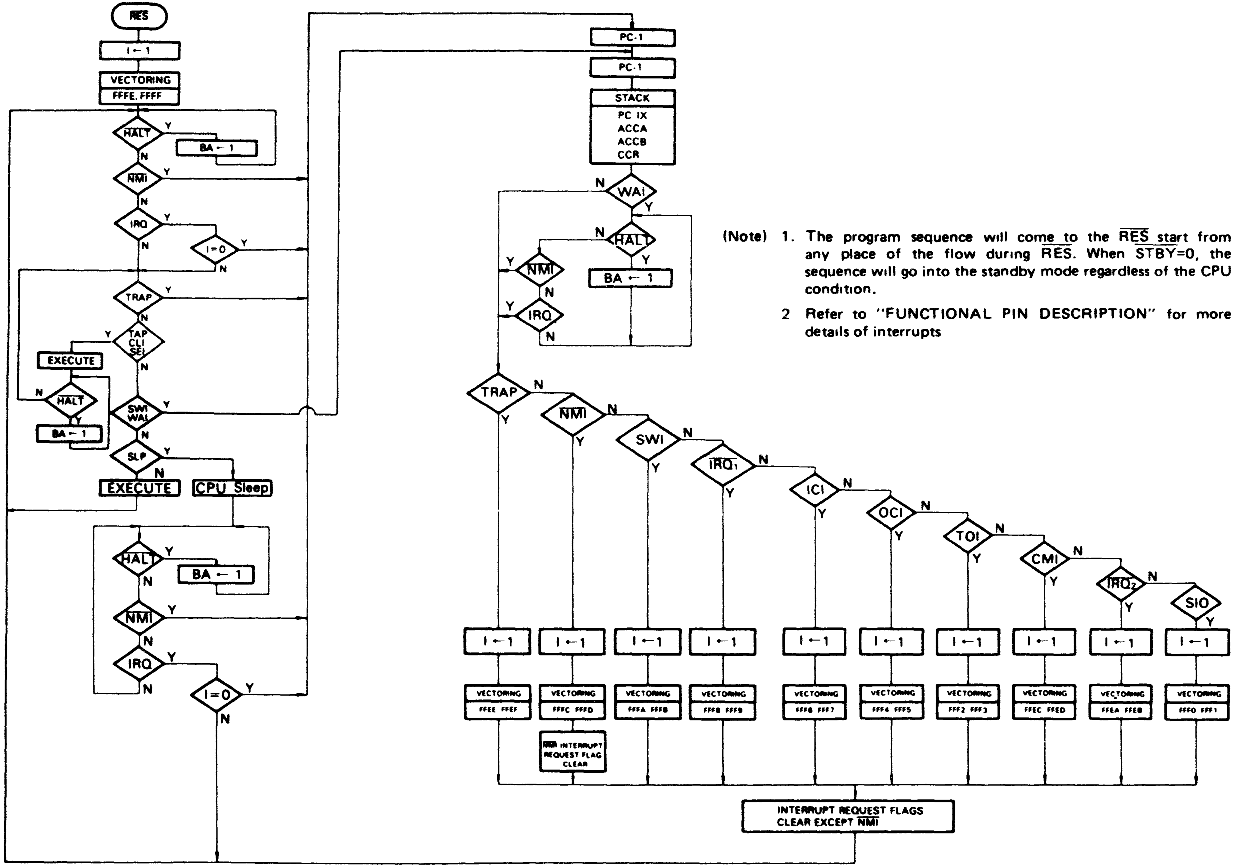


Figure 29 CPU Operation Mode Transition

Table 16 CPU Operation State and Port, Bus, Control Signal State

Port	Reset	STBY*3	HALT	Sleep
$A_0 \sim A_7$	H	T	T	H
Port 2	T	T	Keep	Keep
$D_0 \sim D_7$	T	T	T	T
$A_8 \sim A_{15}$	H	T	T	H
Port 5	T	T	Keep	Keep
Port 6	T	T	Keep	Keep
Control Signal	*1	T	*2	*1

*1 $\overline{RD}$, $\overline{WR}$, R/$\overline{W}$, $\overline{LIR}$ = H, BA = L
*2 $\overline{RD}$, $\overline{WR}$, R/$\overline{W}$ = T, $\overline{LIR}$, BA = H
*3 E pin goes to high impedance state



(Note) 1. The program sequence will come to the RES start from any place of the flow during RES. When STBY=0, the sequence will go into the standby mode regardless of the CPU condition.

2 Refer to "FUNCTIONAL PIN DESCRIPTION" for more details of interrupts

Figure 30 HD6303Y System Flow Chart

Table 17 Cycle-by-Cycle Operation

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/$\overline{W}$	$\overline{RD}$	$\overline{WR}$	$\overline{LIR}$	Data Bus
IMMEDIATE								
ADC ADD	2	1	Op Code Address+1	1	0	1	1	Operand Data
AND BIT		2	Op Code Address+2	1	0	1	0	Next Op Code
CMP EOR								
LDA ORA								
SBC SUB								
ADDD CPX	3	1	Op Code Address+1	1	0	1	1	Operand Data (MSB)
LDD LDS		2	Op Code Address+2	1	0	1	1	Operand Data (LSB)
LDX SUBD		3	Op Code Address+3	1	0	1	0	Next Op Code
DIRECT								
ADC ADD	3	1	Op Code Address+1	1	0	1	1	Address of Operand (LSB)
AND BIT		2	Address of Operand	1	0	1	1	Operand Data
CMP EOR		3	Op Code Address+2	1	0	1	0	Next Op Code
LDA ORA								
SBC SUB								
STA	3	1	Op Code Address+1	1	0	1	1	Destination Address
		2	Destination Address	0	1	0	1	Accumulator Data
		3	Op Code Address+2	1	0	1	0	Next Op Code
ADDD CPX	4	1	Op Code Address+1	1	0	1	1	Address of Operand (LSB)
LDD LDS		2	Address of Operand	1	0	1	1	Operand Data (MSB)
LDX SUBD		3	Address of Operand+1	1	0	1	1	Operand Data (LSB)
		4	Op Code Address+2	1	0	1	0	Next Op Code
STD STS	4	1	Op Code Address+1	1	0	1	1	Destination Address (LSB)
STX		2	Destination Address	0	1	0	1	Register Data (MSB)
		3	Destination Address+1	0	1	0	1	Register Data (LSB)
		4	Op Code Address+2	1	0	1	0	Next Op Code
JSR	5	1	Op Code Address+1	1	0	1	1	Jump Address (LSB)
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	Stack Pointer	0	1	0	1	Return Address (LSB)
		4	Stack Pointer−1	0	1	0	1	Return Address (MSB)
		5	Jump Address	1	0	1	0	First Subroutine Op Code
TIM	4	1	Op Code Address+1	1	0	1	1	Immediate Data
		2	Op Code Address+2	1	0	1	1	Address of Operand (LSB)
		3	Address of Operand	1	0	1	1	Operand Data
		4	Op Code Address+3	1	0	1	0	Next Op Code
AIM EIM	6	1	Op Code Address+1	1	0	1	1	Immediate Data
OIM		2	Op Code Address+2	1	0	1	1	Address of Operand (LSB)
		3	Address of Operand	1	0	1	1	Operand Data
		4	FFFF	1	1	1	1	Restart Address (LSB)
		5	Address of Operand	0	1	0	1	New Operand Data
		6	Op Code Address+3	1	0	1	0	Next Op Code

(Continued)

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/$\overline{W}$	$\overline{RD}$	$\overline{WR}$	$\overline{LIR}$	Data Bus
INDEXED								
JMP	3	1	Op Code Address + 1	1	0	1	1	Offset
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	Jump Address	1	0	1	0	First Op Code of Jump Routine
ADC ADD	4	1	Op Code Address + 1	1	0	1	1	Offset
AND BIT		2	FFFF	1	1	1	1	Restart Address (LSB)
CMP EOR		3	IX + Offset	1	0	1	1	Operand Data
LDA ORA		4	Op Code Address + 2	1	0	1	0	Next Op Code
SBC SUB								
TST								
STA	4	1	Op Code Address + 1	1	0	1	1	Offset
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	IX + Offset	0	1	0	1	Accumulator Data
		4	Op Code Address + 2	1	0	1	0	Next Op Code
ADDD	5	1	Op Code Address + 1	1	0	1	1	Offset
CPX LDD		2	FFFF	1	1	1	1	Restart Address (LSB)
LDS LDX		3	IX + Offset	1	0	1	1	Operand Data (MSB)
SUBD		4	IX + Offset + 1	1	0	1	1	Operand Data (LSB)
		5	Op Code Address + 2	1	0	1	0	Next Op Code
STD STS	5	1	Op Code Address + 1	1	0	1	1	Offset
STX		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	IX + Offset	0	1	0	1	Register Data (MSB)
		4	IX + Offset + 1	0	1	0	1	Register Data (LSB)
		5	Op Code Address + 2	1	0	1	0	Next Op Code
JSR	5	1	Op Code Address + 1	1	0	1	1	Offset
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	Stack Pointer	0	1	0	1	Return Address (LSB)
		4	Stack Pointer − 1	0	1	0	1	Return Address (MSB)
		5	IX + Offset	1	0	1	0	First Subroutine Op Code
ASL ASR	6	1	Op Code Address + 1	1	0	1	1	Offset
COM DEC		2	FFFF	1	1	1	1	Restart Address (LSB)
INC LSR		3	IX + Offset	1	0	1	1	Operand Data
NEG ROL		4	FFFF	1	1	1	1	Restart Address (LSB)
ROR		5	IX + Offset	0	1	0	1	New Operand Data
		6	OP Code Address + 2	1	0	1	0	Next Op Code
TIM	5	1	Op Code Address + 1	1	0	1	1	Immediate Data
		2	Op Code Address + 2	1	0	1	1	Offset
		3	FFFF	1	1	1	1	Restart Address (LSB)
		4	IX + Offset	1	0	1	1	Operand Data
		5	Op Code Address + 3	1	0	1	0	Next Op Code
CLR	5	1	Op Code Address + 1	1	0	1	1	Offset
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	IX + Offset	1	0	1	1	Operand Data
		4	IX + Offset	0	1	0	1	00
		5	Op Code Address + 2	1	0	1	0	Next Op Code
AIM EIM	7	1	Op Code Address + 1	1	0	1	1	Immediate Data
OIM		2	Op Code Address + 2	1	0	1	1	Offset
		3	FFFF	1	1	1	1	Restart Address (LSB)
		4	IX + Offset	1	0	1	1	Operand Data
		5	FFFF	1	1	1	1	Restart Address (LSB)
		6	IX + Offset	0	1	0	1	New Operand Data
		7	Op Code Address + 3	1	0	1	0	Next Op Code

(Continued)

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/$\overline{W}$	$\overline{RD}$	$\overline{WR}$	$\overline{LIR}$	Data Bus
EXTEND								
JMP	3	1	Op Code Address + 1	1	0	1	1	Jump Address (MSB)
		2	Op Code Address + 2	1	0	1	1	Jump Address (LSB)
		3	Jump Address	1	0	1	0	Next Op Code
ADC ADD TST	4	1	Op Code Address + 1	1	0	1	1	Address of Operand (MSB)
AND BIT		2	Op Code Address + 2	1	0	1	1	Address of Operand (LSB)
CMP EOR		3	Address of Operand	1	0	1	1	Operand Data
LDA ORA		4	Op Code Address + 3	1	0	1	0	Next Op Code
SBC SUB								
STA	4	1	Op Code Address + 1	1	0	1	1	Destination Address (MSB)
		2	Op Code Address + 2	1	0	1	1	Destination Address (LSB)
		3	Destination Address	0	1	0	1	Accumulator Data
		4	Op Code Address + 3	1	0	1	0	Next Op Code
ADDD	5	1	Op Code Address + 1	1	0	1	1	Address of Operand (MSB)
CPX LDD		2	Op Code Address + 2	1	0	1	1	Address of Operand (LSB)
LDS LDX		3	Address of Operand	1	0	1	1	Operand Data (MSB)
SUBD		4	Address of Operand + 1	1	0	1	1	Operand Data (LSB)
		5	Op Code Address + 3	1	0	1	0	Next Op Code
STD STS	5	1	Op Code Address + 1	1	0	1	1	Destination Address (MSB)
STX		2	Op Code Address + 2	1	0	1	1	Destination Address (LSB)
		3	Destination Address	0	1	0	1	Register Data (MSB)
		4	Destination Address + 1	0	1	0	1	Register Data (LSB)
		5	Op Code Address + 3	1	0	1	0	Next Op Code
JSR	6	1	Op Code Address + 1	1	0	1	1	Jump Address (MSB)
		2	Op Code Address + 2	1	0	1	1	Jump Address (LSB)
		3	FFFF	1	1	1	1	Restart Address (LSB)
		4	Stack Pointer	0	1	0	1	Return Address (LSB)
		5	Stack Pointer − 1	0	1	0	1	Return Address (MSB)
		6	Jump Address	1	0	1	0	First Subroutine Op Code
ASL ASR	6	1	Op Code Address + 1	1	0	1	1	Address of Operand (MSB)
COM DEC		2	Op Code Address + 2	1	0	1	1	Address of Operand (LSB)
INC LSR		3	Address of Operand	1	0	1	1	Operand Data
NEG ROL		4	FFFF	1	1	1	1	Restart Address (LSB)
ROR		5	Address of Operand	0	1	0	1	New Operand Data
		6	Op Code Address + 3	1	0	1	0	Next Op Code
CLR	5	1	Op Code Address + 1	1	0	1	1	Address of Operand (MSB)
		2	Op Code Address + 2	1	0	1	1	Address of Operand (LSB)
		3	Address of Operand	1	0	1	1	Operand Data
		4	Address of Operand	0	1	0	1	00
		5	Op Code Address + 3	1	0	1	0	Next Op Code

(Continued)

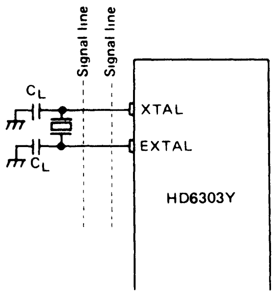
Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/$\overline{W}$	$\overline{RD}$	$\overline{WR}$	$\overline{LIR}$	Data Bus
IMPLIED								
ABA ABX ASL ASLD ASR CBA CLC CLI CLR CLV COM DEC DES DEX INC INS INX LSR LSRD ROL ROR NOP SBA SEC SEI SEV TAB TAP TBA TPA TST TSX TXS	1	1	Op Code Address + 1	1	0	1	0	Next Op Code
DAA XGDX	2	1	Op Code Address + 1	1	0	1	0	Next Op Code
		2	FFFF	1	1	1	1	Restart Address (LSB)
PULA PULB	3	1	Op Code Address + 1	1	0	1	0	Next Op Code
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	Stack Pointer + 1	1	0	1	1	Data from Stack
PSHA PSHB	4	1	Op Code Address + 1	1	0	1	1	Next Op Code
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	Stack Pointer	0	1	0	1	Accumulator Data
		4	Op Code Address + 1	1	0	1	0	Next Op Code
PULX	4	1	Op Code Address + 1	1	0	1	0	Next Op Code
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	Stack Pointer + 1	1	0	1	1	Data from Stack (MSB)
		4	Stack Pointer + 2	1	0	1	1	Data from Stack (LSB)
PSHX	5	1	Op Code Address + 1	1	0	1	1	Next Op Code
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	Stack Pointer	0	1	0	1	Index Register (LSB)
		4	Stack Pointer − 1	0	1	0	1	Index Register (MSB)
		5	Op Code Address + 1	1	0	1	0	Next Op Code
RTS	5	1	Op Code Address + 1	1	0	1	1	Next Op Code
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	Stack Pointer + 1	1	0	1	1	Return Address (MSB)
		4	Stack Pointer + 2	1	0	1	1	Return Address (LSB)
		5	Return Address	1	0	1	0	First Op Code of Return Routine
MUL	7	1	Op Code Address + 1	1	0	1	0	Next Op Code
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	FFFF	1	1	1	1	Restart Address (LSB)
		4	FFFF	1	1	1	1	Restart Address (LSB)
		5	FFFF	1	1	1	1	Restart Address (LSB)
		6	FFFF	1	1	1	1	Restart Address (LSB)
		7	FFFF	1	1	1	1	Restart Address (LSB)

(Continued)

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/$\overline{W}$	$\overline{RD}$	$\overline{WR}$	$\overline{LIR}$	Data Bus
IMPLIED								
WAI	9	1	Op Code Address+1	1	0	1	1	Next Op Code
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	Stack Pointer	0	1	0	1	Return Address (LSB)
		4	Stack Pointer−1	0	1	0	1	Return Address (MSB)
		5	Stack Pointer−2	0	1	0	1	Index Register (LSB)
		6	Stack Pointer−3	0	1	0	1	Index Register (MSB)
		7	Stack Pointer−4	0	1	0	1	Accumulator A
		8	Stack Pointer−5	0	1	0	1	Accumulator B
		9	Stack Pointer−6	0	1	0	1	Conditional Code Register
RTI	10	1	Op Code Address+1	1	0	1	1	Next Op Code
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	Stack Pointer+1	1	0	1	1	Conditional Code Register
		4	Stack Pointer+2	1	0	1	1	Accumulator B
		5	Stack Pointer+3	1	0	1	1	Accumulator A
		6	Stack Pointer+4	1	0	1	1	Index Register (MSB)
		7	Stack Pointer+5	1	0	1	1	Index Register (LSB)
		8	Stack Pointer+6	1	0	1	1	Return Address (MSB)
		9	Stack Pointer+7	1	0	1	1	Return Address (LSB)
		10	Return Address	1	0	1	0	First Op Code of Return Routine
SWI	12	1	Op Code Address+1	1	0	1	1	Next Op Code
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	Stack Pointer	0	1	0	1	Return Address (LSB)
		4	Stack Pointer−1	0	1	0	1	Return Address (MSB)
		5	Stack Pointer−2	0	1	0	1	Index Register (LSB)
		6	Stack Pointer−3	0	1	0	1	Index Register (MSB)
		7	Stack Pointer−4	0	1	0	1	Accumulator A
		8	Stack Pointer−5	0	1	0	1	Accumulator B
		9	Stack Pointer−6	0	1	0	1	Conditional Code Register
		10	Vector Address FFFA	1	0	1	1	Address of SWI Routine (MSB)
		11	Vector Address FFFB	1	0	1	1	Address of SWI Routine (LSB)
		12	Address of SWI Routine	1	0	1	0	First Op Code of SWI Routine
SLP	4	1	Op Code Address+1	1	0	1	1	Next Op Code
		2	FFFF	1	1	1	1	Restart Address (LSB)
		↕ Sleep		↓	↓	↓	↓	↓
		3	FFFF	1	1	1	1	Restart Address (LSB)
		4	Op Code Address+1	1	0	1	0	Next Op Code
RELATIVE								
BCC BCS	3	1	Op Code Address+1	1	0	1	1	Branch Offset
BEQ BGE		2	FFFF	1	1	1	1	Restart Address (LSB)
BGT BHI		3	Branch Address Test="1 Op Code Address+1 Test="0	1	0	1	0	First Op Code of Branch Routine Next Op Code
BLE BLS								
BLT BMT								
BNE BPL								
BRA BRN								
BVC BVS								
BSR	5	1	Op Code Address+1	1	0	1	1	Offset
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	Stack Pointer	0	1	0	1	Return Address (LSB)
		4	Stack Pointer−1	0	1	0	1	Return Address (MSB)
		5	Branch Address	1	0	1	0	First Op Code of Subroutine

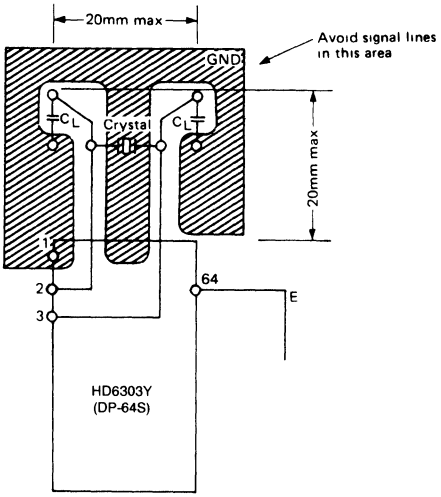
■ WARNING CONCERNING THE BOARD DESIGN OF OSCILLATION CIRCUIT

When designing a board, note that crosstalk may disturb the normal oscillation if signal lines are placed near the oscillation circuit as shown in Figure 31. Place the crystal and C_L as close to the HD6303Y as possible.



Do not use this kind of printed-circuit board design.

Figure 31 Warning concerning board design of oscillation circuit



(Top View)

Figure 32 Example of Oscillation Circuits in Board Design

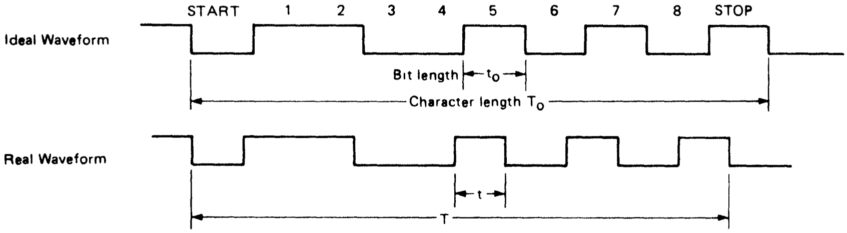
■ RECEIVE MARGIN OF THE SCI

Receive margin of the SCI contained in the HD6303Y is shown in Table 18

Note: SCI = Serial Communication ~~Interface~~

Table 18

	Bit distortion tolerance (t−to) /to	Character distortion tolerance (T−To) /To
HD6303Y	±43.7%	±4.37%



■ WARNING CONCERNING WAI INSTRUCTION

If the HALT signal is accepted by the MCU while the WAI instruction is executing, the CPU will not operate correctly after HALT mode is canceled.

WAI is a instruction which waits for an interrupt. The corresponding interrupt routine is executed after an interrupt occurs.

However, during the execution of the WAI instruction, $\overline{\text{HALT}}$ input makes the CPU malfunction and fetch an abnormal interrupt vectoring address.

In HALT mode, the CPU operates correctly without the WAI instruction, and WAI is executed correctly without $\overline{\text{HALT}}$ input. Therefore, if $\overline{\text{HALT}}$ input is necessary, make interrupts wait during the loop routine, as shown in Figure 33.

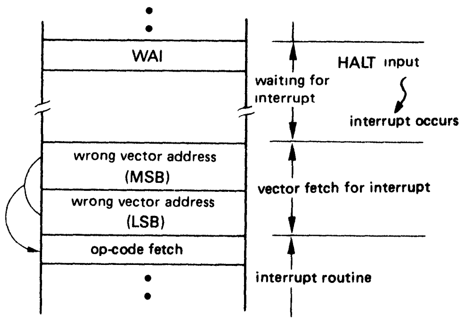
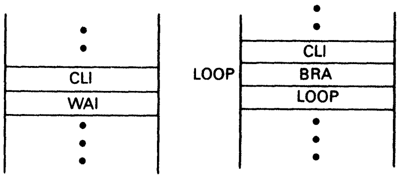


Figure 33 MAC function during WAI



i) MAL function ii) Recommended method

Figure 34 Program to wait for interrupt

■ WRITE-ONLY REGISTER

When the CPU reads a write-only register, the read data is always $FF, regardless of the value in the write-only register. Therefore, be careful of the results of instructions which read write-only register and perform an arithmetic or logical operation on its contents, such as AIM, ADD, or ROL, is executed, because the arithmetic or logical operation is always done with the data $FF. In particulars, don't use the AIM, OIM or EIM instruction to manipulate the DDR bit of PORT.

■ WARNING CONCERNING POWER START-UP

$\overline{\text{RES}}$ must be held low for at least 20 ms when the power starts up. In this case, the internal reset function is not effective until the oscillation begins at power-on. The $\overline{\text{RES}}$ signal is input to the LSI in synchronism with the internal clock ϕ (shown in Figure 35.)

Therefore, after power starts up, the LSI conditions such as its I/O ports and operating mode, are unstable. Fix the level of I/O ports by means of an external circuit to determine the level for system operation during the oscillator stabilization time.

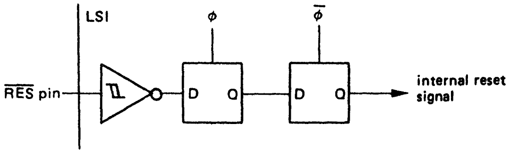


Figure 35 $\overline{\text{RES}}$ circuit

HD6305X2, HD63A05X2, HD63B05X2 — CMOS MCU (Microcomputer Unit)

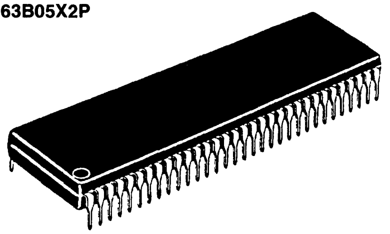
The HD6305X2 is memory expandable versions of the HD6305X0, which is CMOS 8-bit single chip microcomputer. A CPU, a clock generator, a 128-byte RAM, I/O terminals, two timers and a serial communication interface (SCI) are built in the HD6305X2.

The HD6305X2 has the same functions as the HD6305X0's except for the number of I/O terminals. The HD6305X2 is a microcomputer unit which includes no ROM and its memory space is expandable to 16k bytes externally.

■ HARDWARE FEATURES

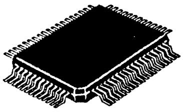
- 8-bit based MCU
- 128-bytes of RAM
- A total of 31 terminals, including 24 I/O's, 7 inputs
- Two timers
  - — 8-bit timer with a 7-bit prescaler (programmable prescaler, event counter)
  - — 15-bit timer (commonly used with the SCI clock divider)
- On-chip serial interface circuit (synchronized with clock)
- Six interrupts (two external, two timer, one serial and one software)
- Low power dissipation modes
  - — Wait..... In this mode, the clock oscillator is on and the CPU halts but the timer/serial/interrupt function is operable.
  - — Stop..... In this mode, the clock stops but the RAM data, I/O status and registers are held.
  - — Standby.. In this mode, the clock stops, the RAM data is held, and the other internal condition is reset.
- Minimum instruction cycle time
  - —HD6305X2...1 μs (f = 1 MHz)
  - —HD63A05X2...0.67 μs (f = 1.5 MHz)
  - —HD63B05X2...0.5 μ (f = 2 MHz)
- Wide operating range
  - V_{CC} = 3 to 6V (f = 0.1 to 0.5 MHz)
  - —HD6305X2.. f = 0.1 to 1 MHz (V_{CC} = 5V ± 10%)
  - —HD63A05X2. f = 0.1 to 1.5 MHz (V_{CC} = 5V ± 10%)
  - —HD63B05X2. f = 0.1 to 2 MHz (V_{CC} = 5V ± 10%)

HD6305X2P, HD63A05X2P, HD63B05X2P



(DP-64S)

HD6305X2F, HD63A05X2F, HD63B05X2F



(FP-64)

2

■ SOFTWARE FEATURES

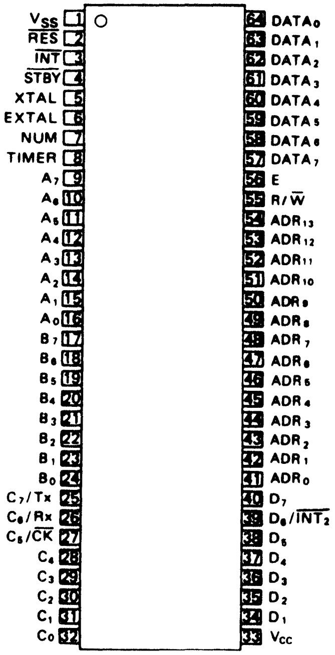
- Similar to HD6800
- Byte efficient instruction set
- Powerful bit manipulation instructions (Bit Set, Bit Clear, and Bit Test and Branch usable for all RAM bits and all I/O terminals)
- A variety of interrupt operations
- Index addressing mode useful for table processing
- A variety of conditional branch instructions
- Ten powerful addressing modes
- All addressing modes adaptable to RAM, and I/O instructions
- Three new instructions, STOP, WAIT and DAA, added to the HD6805 family instruction set

■ PROGRAM DEVELOPMENT SUPPORT TOOLS

- Cross assembler software for use with IBM PCs and compatibles
- In circuit emulator for use with IBM PCs and compatibles

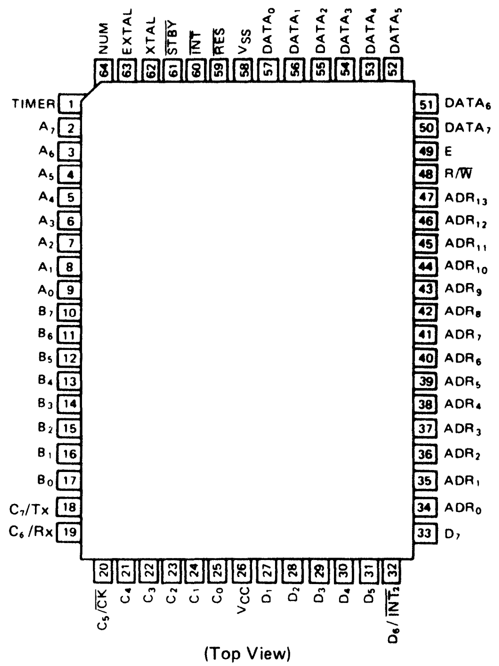
■ PIN ARRANGEMENT

- HD6305X2P, HD63A05X2P, HD63B05X2P



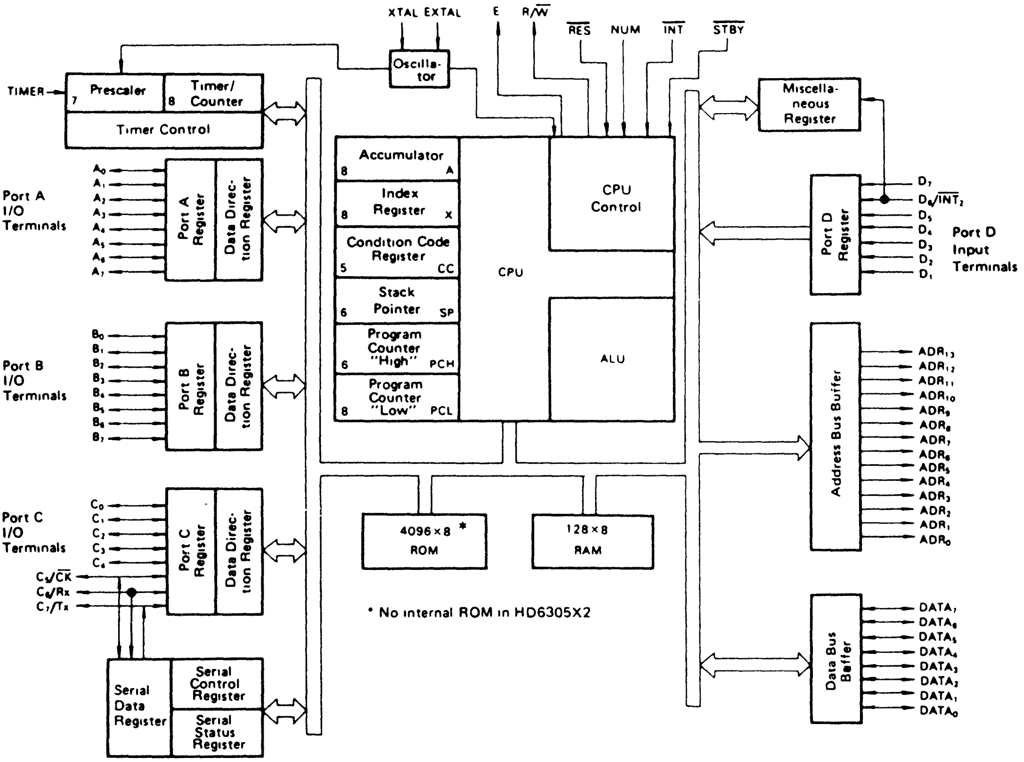
(Top View)

- HD6305X2F, HD63A05X2F, HD63B05X2F



(Top View)

■ BLOCK DIAGRAM



■ ABSOLUTE MAXIMUM RATINGS

Item	Symbol	Value	Unit
Supply Voltage	V_{CC}	-0.3 ~ +7.0	V
Input Voltage	V_{in}	-0.3 ~ V_{CC} + 0.3	V
Operating Temperature	T_{opr}	0 ~ +70	°C
Storage Temperature	T_{stg}	-55 ~ +150	°C

[NOTE] These products have a protection circuit in their input terminals against high electrostatic voltage or high electric fields Notwithstanding, be careful not to apply any voltage higher than the absolute maximum rating to these high input impedance circuits To assure normal operation, we recommended V_{in}, V_{out}: $V_{SS} \leqq (V_{in}$ or $V_{out}) \leqq V_{CC}$

■ ELECTRICAL CHARACTERISTICS

● DC CHARACTERISTICS (V_{CC} = 5.0V±10%, V_{SS} = GND, Ta = 0 ~ +70°C, unless otherwise noted.)

Item		Symbol	Test Condition	min	typ	max	Unit
Input "High" Voltage	$\overline{RES}$, $\overline{STBY}$	V_{IH}		V_{CC}-0.5	–	V_{CC}+0.3	V
	EXTAL			V_{CC}x0.7	–	V_{CC}+0.3	
	Other Inputs			2.0	–	V_{CC}+0.3	
Input "Low" Voltage	All Inputs	V_{IL}		-0.3	–	0.8	V
Output "High" Voltage	All Outputs	V_{OH}	I_{OH} = -200μA	2.4	–	–	V
			I_{OH} = -10μA	V_{CC}-0.7	–	–	
Output "Low" Voltage	All Outputs	V_{OL}	I_{OL} = 1.6mA	–	–	0.55	V
Input Leakage Current	TIMER, $\overline{INT}$, D_1 ~ D_7, $\overline{STBY}$	$\lvert I_{IL}\rvert$	Vin = 0.5 ~ V_{CC}-0.5	–	–	1.0	μA
Three-state Current	A_0 ~ A_7, B_0 ~ B_7, C_0 ~ C_7, ADR_0 ~ ADR_{13}*, $DATA_0$~ $DATA_7$, E*,R/$\overline{W}$*	$\lvert I_{TSI}\rvert$		–	–	1.0	μA
Current Dissipation**	Operating	I_{CC}	f = 1MHz***	–	5	10	mA
	Wait			–	2	5	mA
	Stop			–	2	10	μA
	Standby			–	2	10	μA
Input Capacitance	All Terminals	Cin	f = 1MHz, Vin = 0V	–	–	12	pF

* Only at standby

** V_{IH} min = V_{CC}-1 0V, V_{IL} max = 0 8V

*** The value at f = xMHz is given by using
I_{CC} (f = xMHz) = I_{CC} (f = 1MHz) x x

● AC CHARACTERISTICS (V_{CC} = 5.0V±10%, V_{SS} = GND, Ta = 0 ~ +70°C, unless otherwise noted.)

Item	Symbol	Test Condition	HD6305X2			HD63A05X2			HD63B05X2			Unit
			min	typ	max	min	typ	max	min	typ	max	
Cycle Time	t_{cyc}	Fig 1	1	–	10	0.666	–	10	0.5	–	10	μs
Enable Rise Time	t_{Er}		–	–	20	–	–	20	–	–	20	ns
Enable Fall Time	t_{Ef}		–	–	20	–	–	20	–	–	20	ns
Enable Pulse Width("High" Level)	PW_{EH}		450	–	–	300	–	–	220	–	–	ns
Enable Pulse Width("Low" Level)	PW_{EL}		450	–	–	300	–	–	220	–	–	ns
Address Delay Time	t_{AD}		–	–	250	–	–	190	–	–	180	ns
Address Hold Time	t_{AH}		40	–	–	30	–	–	20	–	–	ns
Data Delay Time	t_{DW}		–	–	200	–	–	160	–	–	120	ns
Data Hold Time (Write)	t_{HW}		40	–	–	30	–	–	20	–	–	ns
Data Set-up Time (Read)	t_{DSR}		80	–	–	60	–	–	50	–	–	ns
Data Hold Time (Read)	t_{HR}		0	–	–	0	–	–	0	–	–	ns

- **PORT TIMING (V_{CC} = 5.0V±10%, V_{SS} = GND, Ta = 0 ~ +70°C, unless otherwise noted.)**

Item	Symbol	Test Condition	HD6305X2			HD63A05X2			HD63B05X2			Unit
			min	typ	max	min	typ	max	min	typ	max	
Port Data Set-up Time (Port A, B, C, D)	t_{PDS}	Fig. 2	200	–	–	200	–	–	200	–	–	ns
Port Data Hold Time (Port A, B, C, D)	t_{PDH}		200	–	–	200	–	–	200	–	–	ns
Port Data Delay Time (Port A, B, C)	t_{PDW}	Fig. 3	–	–	300	–	–	300	–	–	300	ns

- **CONTROL SIGNAL TIMING (V_{CC} = 5.0V±10%, V_{SS} = GND, Ta = 0 ~ +70°C, unless otherwise noted.)**

Item	Symbol	Test Condition	HD6305X2			HD63A05X2			HD63B05X2			Unit
			min	typ	max	min	typ	max	min	typ	max	
$\overline{INT}$ Pulse Width	t_{IWL}		t_{cyc} +250	–	–	t_{cyc} +200	–	–	t_{cyc} +200	–	–	ns
$\overline{INT_2}$ Pulse Width	t_{IWL2}		t_{cyc} +250	–	–	t_{cyc} +200	–	–	t_{cyc} +200	–	–	ns
$\overline{RES}$ Pulse Width	t_{RWL}		5	–	–	5	–	–	5	–	–	t_{cyc}
Control Set-up Time	t_{CS}	Fig 5	250	–	–	250	–	–	250	–	–	ns
Timer Pulse Width	t_{TWL}		t_{cyc} +250	–	–	t_{cyc} +200	–	–	t_{cyc} +200	–	–	ns
Oscillation Start Time (Crystal)	t_{OSC}	Fig 5, Fig 20*	–	–	20	–	–	20	–	–	20	ms
Reset Delay Time	t_{RHL}	Fig. 19	80	–	–	80	–	–	80	–	–	ms

* C_L = 22pF ±20%, R_s = 60Ω max.

- **SCI TIMING (V_{CC} = 5.0V±10%, V_{SS}= GND, Ta = 0 ~ +70°C, unless otherwise noted.)**

Item	Symbol	Test Condition	HD6305X2			HD63A05X2			HD63B05X2			Unit
			min	typ	max	min	typ	max	min	typ	max	
Clock Cycle	t_{Scyc}	Fig 6, Fig. 7	1	–	32768	0.67	–	21845	0.5	–	16384	μs
Data Output Delay Time	t_{TXD}		–	–	250	–	–	250	–	–	250	ns
Data Set-up Time	t_{SRX}		200	–	–	200	–	–	200	–	–	ns
Data Hold Time	t_{HRX}		100	–	–	100	–	–	100	–	–	ns

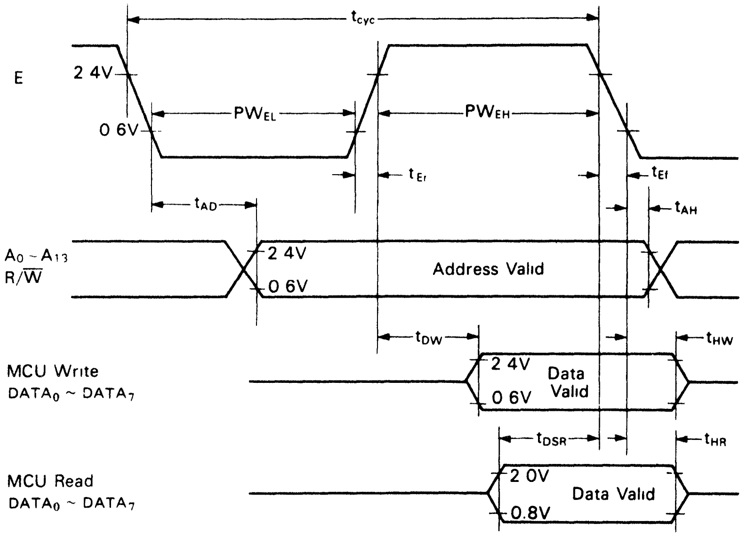


Figure 1 Bus Timing

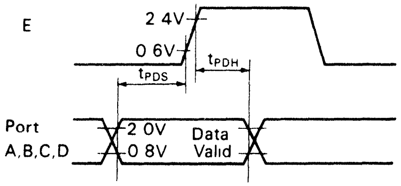


Figure 2 Port Data Set-up and Hold Times (MCU Read)

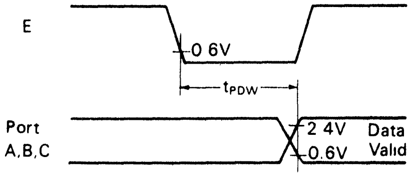


Figure 3 Port Data Delay Time (MCU Write)

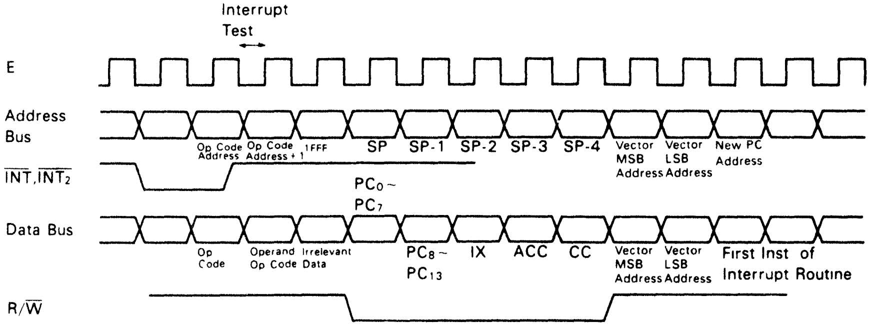


Figure 4 Interrupt Sequence

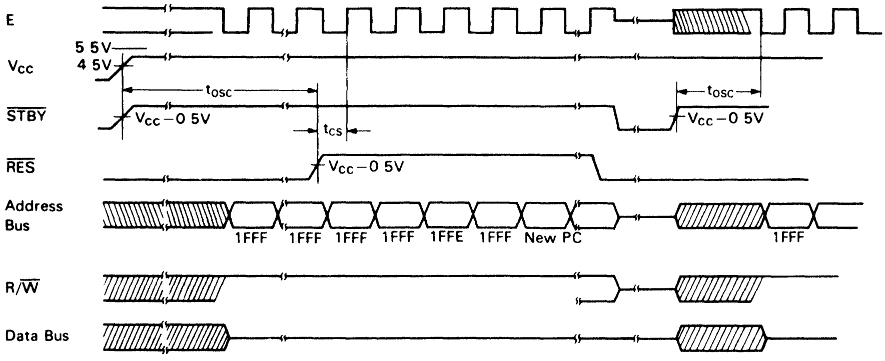
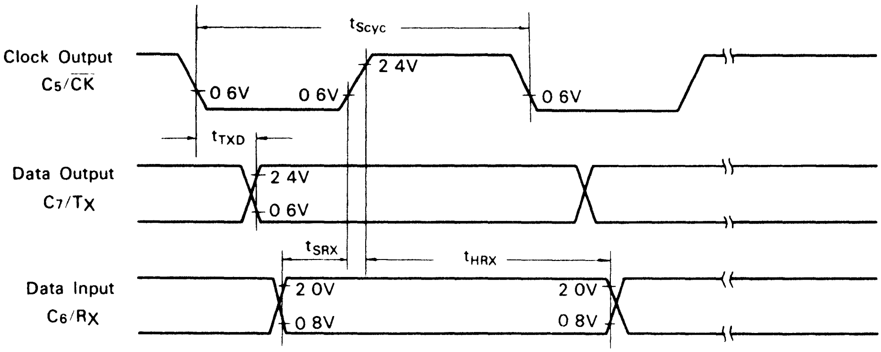
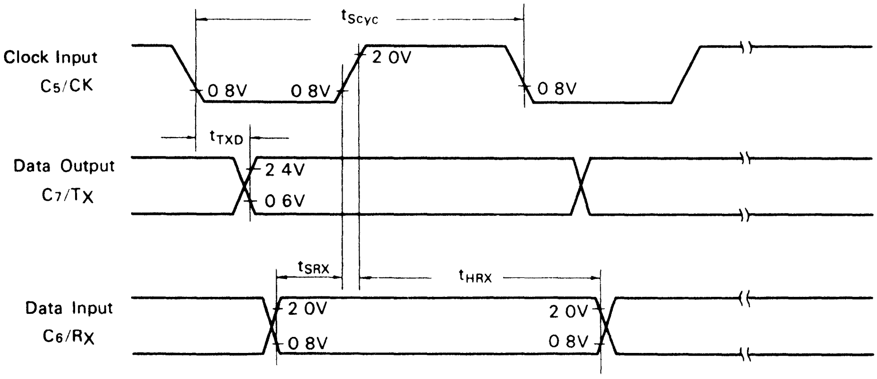


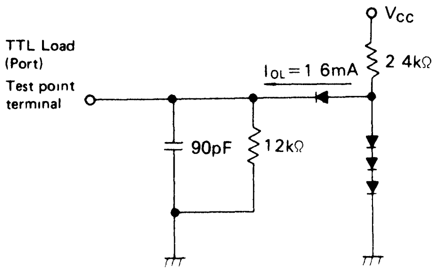
Figure5 Reset Tιmιng



Fιgure6 SCI Tιmιng (Internal Clock)



Fιgure7 SCI Tιmιng(External Clock)



[NOTES] 1. The load capacitance includes stray capacitance caused by the probe, etc
2. All diodes are 1S2074 Ⓗ

Figure 8 Test Load

■ DESCRIPTION OF TERMINAL FUNCTIONS

The input and output signals of the MCU are described here.

• V_{CC}, V_{SS}

Voltage is applied to the MCU through these two terminals. V_{CC} is 5.0V ± 10%, while V_{SS} is grounded.

• $\overline{INT}$, $\overline{INT_2}$

External interrupt request inputs to the MCU. For details, refer to "INTERRUPT" The $\overline{INT_2}$ terminal is also used as the port D_6 terminal

• XTAL, EXTAL

These terminals provide input to the on-chip clock circuit. A crystal oscillator (AT cut, 2.0 to 8.0 MHz) or ceramic filter is connected to the terminal. Refer to "INTERNAL OSCILLATOR" for using these input terminals.

• TIMER

This is an input terminal for event counter. Refer to "TIMER" for details.

• $\overline{RES}$

Used to reset the MCU. Refer to "RESET" for details.

• NUM

This terminal is not for user application. In case of the HD6305X1, this terminal should be connected to V_{CC} through 10kΩ resistance. In case of the HD6305X2, this terminal should be connected to V_{SS}.

• Enable (E)

This output terminal supplies E clock. Output is a single-phase, TTL compatible and 1/4 crystal oscillation frequency or 1/4 external clock frequency. It can drive one TTL load and a 90pF condenser.

• Read/Write (R/$\overline{W}$)

This TTL compatible output signal indicates to peripheral and memory devices whether MCU is in Read ("High"), or in Write ("Low") The normal standby state is Read ("High") Its output can drive one TTL load and a 90pF condenser.

• Data Bus ($DATA_0$ ~ $DATA_7$)

This TTL compatible three-state buffer can drive one TTL load and 90pF.

• Address Bus (ADR_0 ~ ADR_{13})

Each terminal is TTL compatible and can drive one TTL load and 90pF.

• Input/Output Terminals (A_0 ~ A_7, B_0 ~ B_7, C_0 ~ C_7)

These 24 terminals consist of three 8-bit I/O ports (A, B, C). Each of them can be used as an input or output terminal on a bit through program control of the data direction register. For details, refer to "I/O PORTS."

• Input Terminals (D_1 ~ D_7)

These seven input-only terminals are TTL or CMOS compatible. Of the port D's, D_6 is also used as $\overline{INT_2}$. If D_6 is used as a port, the $\overline{INT_2}$ interrupt mask bit of the miscellaneous register must be set to "1" to prevent an $\overline{INT_2}$ interrupt from being accidentally accepted.

• $\overline{STBY}$

This terminal is used to place the MCU into the standby mode. With $\overline{STBY}$ at "Low" level, the oscillation stops and the internal condition is reset. For details, refer to "Standby Mode."

The terminals described in the following are I/O pins for serial communication interface (SCI). They are also used as ports C_5, C_6 and C_7. For details, refer to "SERIAL COMMUNICATION INTERFACE."

• $\overline{CK}$ (C_5)

Used to input or output clocks for serial operation.

• Rx (C_6)

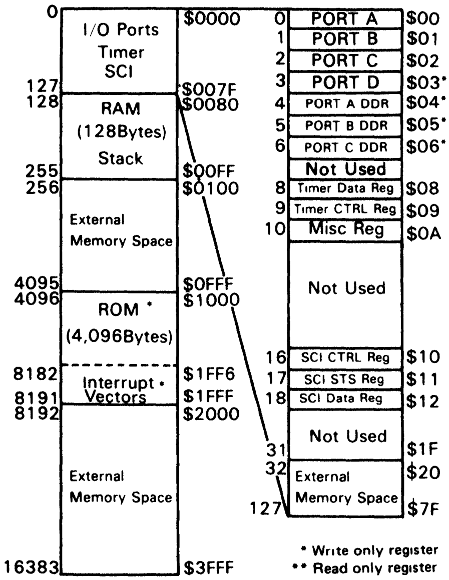
Used to receive serial data.

• Tx (C_7)

Used to transmit serial data.

■ MEMORY MAP

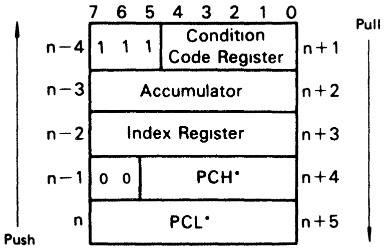
The memory map of the MCU is shown in Fig. 9. \$1000 ~ \$1FFF of the HD6305X2 are external addresses. However, care should be taken to assign vector addresses to \$1FF6 ~ \$1FFF. During interrupt processing, the contents of the CPU registers are saved into the stack in the sequence shown in Fig. 10. This saving begins with the lower byte (PCL) of the program counter. Then the value of the stack pointer is decremented and the higher byte (PCH) of the program counter, index register (X), accumulator (A) and condition code register (CC) are stacked in that order. In a subroutine call, only the contents of the program counter (PCH and PCL) are stacked.



* Write only register
** Read only register

* ROM area ($1000 ~ $1FFF) in the HD6305X2 is changed into External Memory Space

Figure 9 Memory Map of MCU



* In a subroutine call, only PCL and PCH are stacked.

Figure 10 Sequence of Interrupt Stacking

■REGISTERS

There are five registers which the programmer can operate.

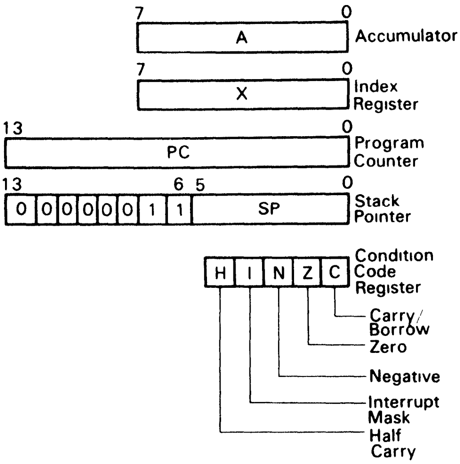


Figure 11 Programming Model

• Accumulator (A)

This accumulator is an ordinary 8-bit register which holds operands or the result of arithmetic operation or data processing.

• Index Register (X)

The index register is an 8-bit register, and is used for index addressing mode. Each of the addresses contained in the register consists of 8 bits which, combined with an offset value, provides an effective address.

In the case of a read/modify/write instruction, the index register can be used like an accumulator to hold operation data or the result of operation.

If not used in the index addressing mode, the register can be used to store data temporarily.

• Program Counter (PC)

The program counter is a 14-bit register that contains the address of the next instruction to be executed.

• Stack Pointer (SP)

The stack pointer is a 14-bit register that indicates the address of the next stacking space. Just after reset, the stack pointer is set at address $00FF. It is decremented when data is pushed, and incremented when pulled. The upper 8 bits of the stack pointer are fixed to 00000011. During the MCU being reset or during a reset stack pointer (RSP) instruction, the pointer is set to address $00FF. Since a subroutine or interrupt can use space up to address $00C1 for stacking, the subroutine can be used up to 31 levels and the interrupt up to 12 levels.

• Condition Code Register (CC)

The condition code register is a 5-bit register, each bit indicating the result of the instruction just executed. The bits can be individually tested by conditional branch instructions. The CC bits are as follows.

Half Carry (H): Used to indicate that a carry occurred between bits 3 and 4 during an arithmetic operation (ADD, ADC).

Interrupt (I): Setting this bit causes all interrupts, except a software interrupt, to be masked. If an interrupt occurs with the bit I set, it is latched. It will be processed the instant the interrupt mask bit is reset. (More specifically, it will enter the interrupt processing routine after the instruction following the CLI has been executed.)

Negative (N): Used to indicate that the result of the most recent arithmetic operation, logical operation or data processing is negative (bit 7 is logic "1").

Zero (Z): Used to indicate that the result of the most recent arithmetic operation, logical operation or data processing is zero.

Carry/Borrow (C): Represents a carry or borrow that occurred in the most recent arithmetic operation This bit is also affected by the Bit Test and Branch instruction and a Rotate instruction

■ INTERRUPT

There are six different types of interrupt. external interrupts ($\overline{INT}$, $\overline{INT_2}$), internal timer interrupts (TIMER, $TIMER_2$), serial interrupt (SCI) and interrupt by an instruction (SWI).

Of these six interrupts, the $\overline{INT_2}$ and TIMER or the SCI and $TIMER_2$ generate the same vector address, respectively.

When an interrupt occurs, the program in progress stops and the then CPU status is saved onto the stack. And then, the interrupt mask bit (I) of the condition code register is set and the start address of the interrupt processing routine is obtained from a particular interrupt vector address. Then the interrupt routine starts from the start address. System can exit from the interrupt routine by an RTI instruction. When this instruction is executed, the CPU status before the interrupt (saved onto the stack) is pulled and the CPU restarts the sequence with the instruction next to the one at which the interrupt occurred. Table 1 lists the priority of interrupts and their vector addresses.

Table 1 Priority of Interrupts

Interrupt	Priority	Vector Address
$\overline{RES}$	1	$1FFE, $1FFF
SWI	2	$1FFC, $1FFD
$\overline{INT}$	3	$1FFA, $1FFB
TIMER/$\overline{INT_2}$	4	$1FF8, $1FF9
SCI/$TIMER_2$	5	$1FF6, $1FF7

A flowchart of the interrupt sequence is shown in Fig. 12. A block diagram of the interrupt request source is shown in Fig. 13

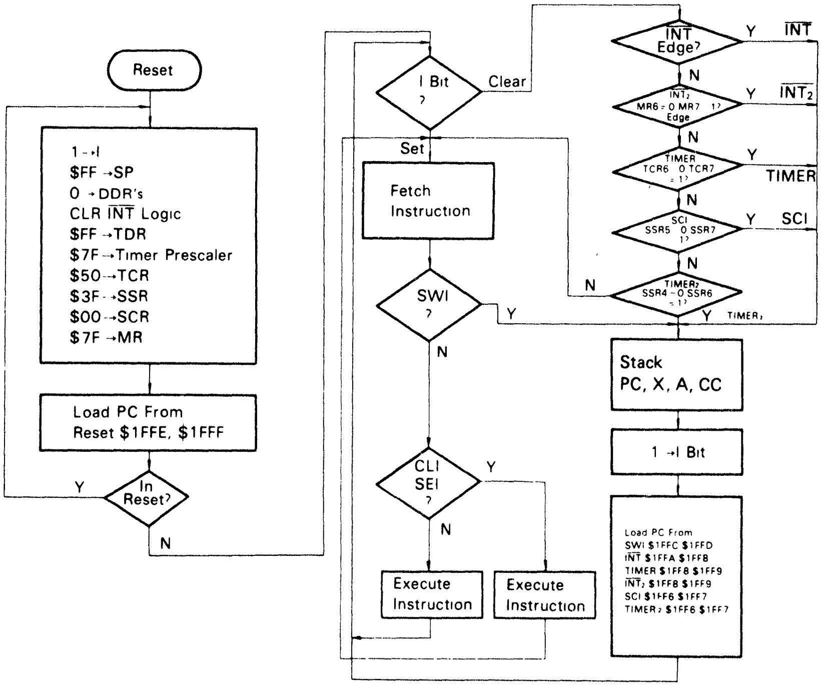


Figure 12 Interrupt Flow Chart

In the block diagram, both the external interrupts $\overline{\text{INT}}$ and $\overline{\text{INT}_2}$ are edge trigger inputs. At the falling edge of each input, an interrupt request is generated and latched. The $\overline{\text{INT}}$ interrupt request is automatically cleared if jumping is made to the $\overline{\text{INT}}$ processing routine. Meanwhile, the $\overline{\text{INT}_2}$ request is cleared if "0" is written in bit 7 of the miscellaneous register.

For the external interrupts ($\overline{\text{INT}}$, $\overline{\text{INT}_2}$), internal timer interrupts (TIMER, TIMER$_2$) and serial interrupt (SCI), each interrupt request is held, but not processed, if the I bit of the condition code register is set. Immediately after the I bit is cleared, the corresponding interrupt processing starts according to the priority.

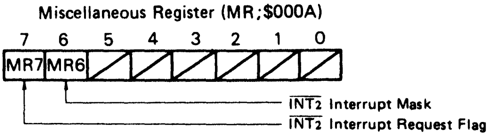
The $\overline{\text{INT}_2}$ interrupt can be masked by setting bit 6 of the miscellaneous register; the TIMER interrupt by setting bit 6 of the timer control register; the SCI interrupt by setting bit 5 of the serial status register; and the TIMER$_2$ interrupt by setting bit 4 of the serial status register.

The status of the $\overline{\text{INT}}$ terminal can be tested by a BIL or BIH instruction. The $\overline{\text{INT}}$ falling edge detector circuit and its latching circuit are independent of testing by these instructions. This is also true with the status of the $\overline{\text{INT}_2}$ terminal.

• Miscellaneous Register (MR; $000A)

The interrupt vector address for the external interrupt $\overline{\text{INT}_2}$ is the same as that for the TIMER interrupt, as shown in Table 1. For this reason, a special register called the miscellaneous register (MR; $000A) is available to control the $\overline{\text{INT}_2}$ interrupts.

Bit 7 of this register is the $\overline{\text{INT}_2}$ interrupt request flag. When the falling edge is detected at the $\overline{\text{INT}_2}$ terminal, "1" is set in bit 7. Then the software in the interrupt routine (vector addresses: $1FF8, $1FF9) checks bit 7 to see if it is $\overline{\text{INT}_2}$ interrupt. Bit 7 can be reset by software.


Miscellaneous Register (MR;$000A)
7
6
5
4
3
2
1
0
MR7
MR6
$\overline{\text{INT}_2}$ Interrupt Mask
$\overline{\text{INT}_2}$ Interrupt Request Flag

Miscellaneous Register (MR; $000A)

Bit 6 is the $\overline{\text{INT}_2}$ interrupt mask bit. If this bit is set to "1", then the $\overline{\text{INT}_2}$ interrupt is disabled. Both read and write are possible with bit 7 but "1" cannot be written in this bit by software. This means that an interrupt request by software is impossible.

When reset, bit 7 is cleared to "0" and bit 6 is set to "1"

■ TIMER

Figure 14 shows a MCU timer block diagram. The timer data register is loaded by software and, upon receipt of a clock input, begins to count down. When the timer data

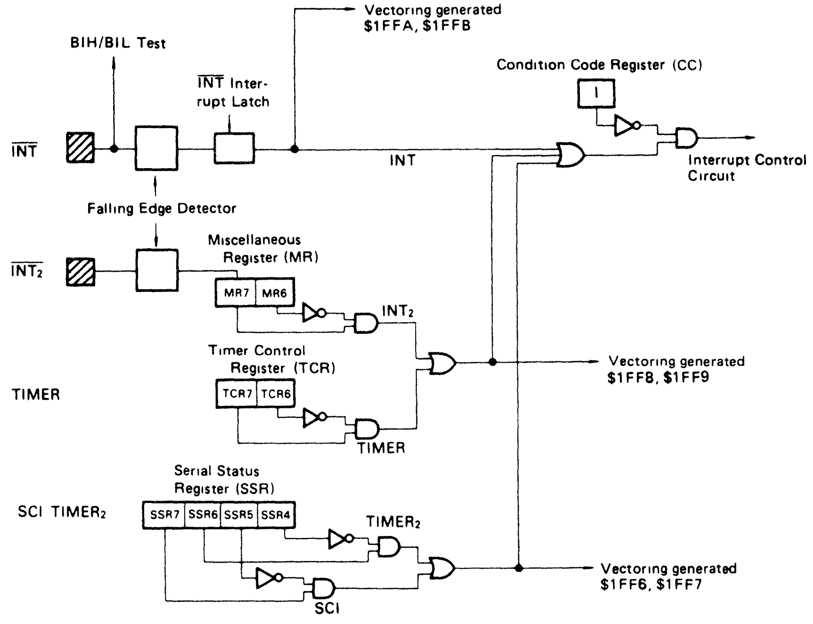


Figure 13 Interrupt Request Generation Circuitry

register (TDR) becomes "0", the timer interrupt request bit (bit 7) in the timer control register is set. In response to the interrupt request, the CPU saves its status into the stack and fetches timer interrupt routine address from addresses $1FF8 and $1FF9 and execute the interrupt routine. The timer interrupt can be masked by setting the timer interrupt mask bit (bit 6) in the timer control register. The mask bit (I) in the condition code register can also mask the timer interrupt.

The source clock to the timer can be either an external signal from the timer input terminal or the internal E signal (the oscillator clock divided by 4). If the E signal is used as the source, the clock input can be gated by the input to the timer input terminal.

Once the timer count has reached "0", it starts counting down with "$FF". The count can be monitored whenever desired by reading the timer data register. This permits the program to know the length of time having passed after the occurrence of a timer interrupt, without disturbing the contents of the counter.

When the MCU is reset, both the prescaler and counter are initialized to logic "1". The timer interrupt request bit (bit 7) then is cleared and the timer interrupt mask bit (bit 6) is set.

To clear the timer interrupt request bit (bit 7), it is necessary to write "0" in that bit.

TCR7	Timer interrupt request
0	Absent
1	Present

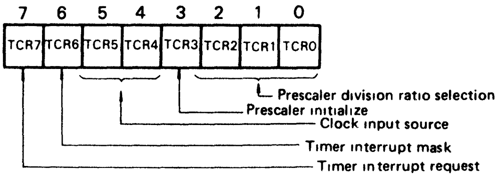
TCR6	Timer interrupt mask
0	Enabled
1	Disabled

• Timer Control Register (TCR; $0009)

Selection of a clock source, selection of a prescaler frequency division ratio, and a timer interrupt can be controlled by the timer control register (TCR; $0009).

For the selection of a clock source, any one of the four modes (see Table 2) can be selected by bits 5 and 4 of the timer control register (TCR).

Timer Control Register (TCR; $0009)



After reset, the TCR is initialized to "E under timer terminal control" (bit 5 = 0, bit 4 = 1). If the timer terminal is "1", the counter starts counting down with "$FF" immediately after reset.

When "1" is written in bit 3, the prescaler is initialized. This bit always shows "0" when read.

Table 2 Clock Source Selection

TCR		Clock input source
Bit 5	Bit 4	
0	0	Internal clock E
0	1	E under timer terminal control
1	0	No clock input (counting stopped)
1	1	Event input from timer terminal

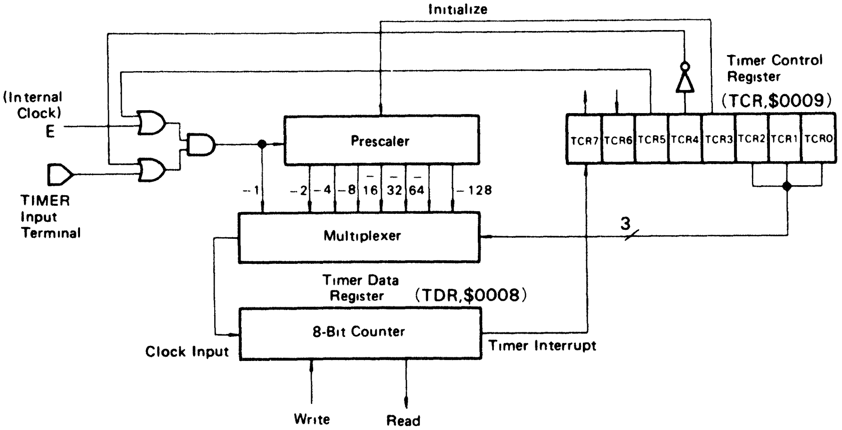


Figure 14 Timer Block Diagram

A prescaler division ratio is selected by the combination of three bits (bits 0, 1 and 2) of the timer control register (see Table 3). There are eight different division ratios: ÷1, ÷2, ÷4, ÷8, ÷16, ÷32, ÷64 and ÷128. After reset, the TCR is set to the ÷1 mode.

Table 3 Prescaler Division Ratio Selection

TCR			Prescaler division ratio
Bit 2	Bit 1	Bit 0	
0	0	0	÷1
0	0	1	÷2
0	1	0	÷4
0	1	1	÷8
1	0	0	÷16
1	0	1	÷32
1	1	0	÷64
1	1	1	÷128

A timer interrupt is enabled when the timer interrupt mask bit is "0", and disabled when the bit is "1". When a timer interrupt occurs, "1" is set in the timer interrupt request bit. This bit can be cleared by writing "0" in that bit.

■SERIAL COMMUNICATION INTERFACE (SCI)

This interface is used for serial transmission or reception of 8-bit data. Sixteen transfer rates are available in the range from 1 μs to approx. 32 ms (for oscillation at 4 MHz).

The SCI consists of three registers, one octal counter and one prescaler. (See Fig. 15.) SCI communicates with the CPU via the data bus, and with the outside world through bits 5, 6 and 7 of port C. Described below are the operations of each register and data transfer.

●SCI Control Register (SCR; $0010)

7	6	5	4	3	2	1	0
SCR7	SCR6	SCR5	SCR4	SCR3	SCR2	SCR1	SCR0

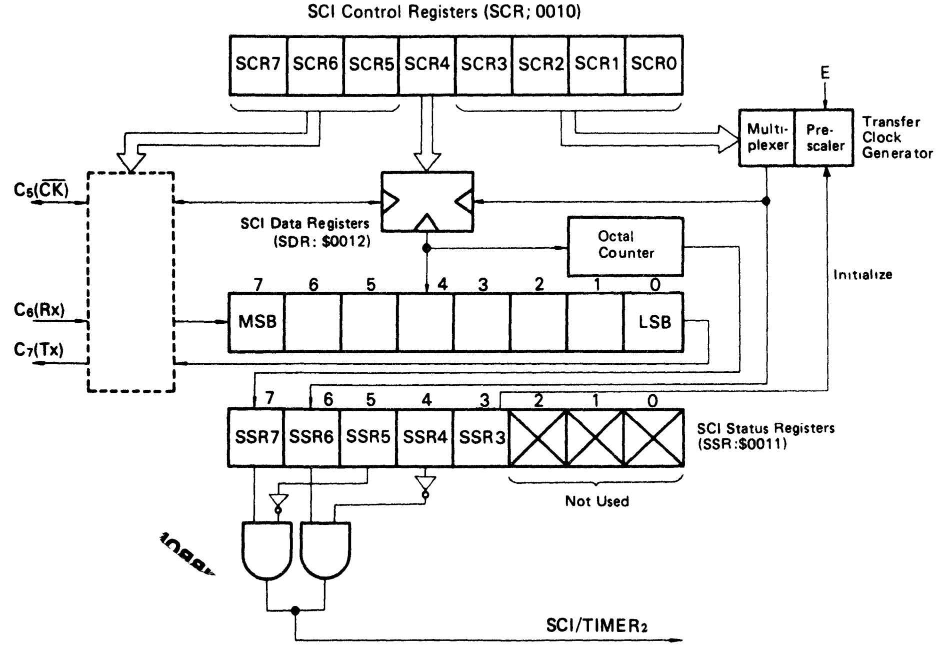


Figure 15 SCI Block Diagram

SCR7	C_7 terminal
0	Used as I/O terminal (by DDR).
1	Serial data output (DDR output)

SCR6	C_6 terminal
0	Used as I/O terminal (by DDR).
1	Serial data input (DDR input)

SCR5	SCR4	Clock source	C_5 terminal
0	0	–	Used as I/O terminal (by DDR).
0	1	–	
1	0	Internal	Clock output (DDR output)
1	1	External	Clock input (DDR input)

Bit 7 (SCR7)

When this bit is set, the DDR corresponding to the C_7 becomes "1" and this terminal serves for output of SCI data. After reset, the bit is cleared to "0"

Bit 6 (SCR6)

When this bit is set, the DDR corresponding to the C_6 becomes "0" and this terminal serves for input of SCI data. After reset, the bit is cleared to "0".

Bits 5 and 4 (SCR5, SCR4)

These bits are used to select a clock source. After reset, the bits are cleared to "0"

Bits 3 ~ 0 (SCR3 ~ SCR0)

These bits are used to select a transfer clock rate. After reset, the bits are cleared to "0".

SCR3	SCR2	SCR1	SCR0	Transfer clock rate	
				4.00 MHz	4.194 MHz
0	0	0	0	1 μs	0.95 μs
0	0	0	1	2 μs	1.91 μs
0	0	1	0	4 μs	3.82 μs
0	0	1	1	8 μs	7 64 μs
≀	≀	≀	≀	≀	≀
1	1	1	1	32768 μs	1/32 s

•SCI Data Register (SDR; $0012)

A serial-parallel conversion register that is used for transfer of data.

•SCI Status Register (SSR; $0011)

7	6	5	4	3	2	1	0
SSR7	SSR6	SSR5	SSR4	SSR3	☒	☒	☒

Bit 7 (SSR7)

Bit 7 is the SCI interrupt request bit which is set upon completion of transmitting or receiving 8-bit data. It is cleared when reset or data is written to or read from the SCI data register with the SCR5="1". The bit can also be cleared by writing "0" in it.

Bit 6 (SSR6)

Bit 6 is the TIMER$_2$ interrupt request bit. TIMER$_2$ is used commonly with the serial clock generator, and SSR6 is set each time the internal transfer clock falls. When reset, the bit is cleared. It also be cleared by writing "0" in it. (For details, see TIMER$_2$.)

Bit 5 (SSR5)

Bit 5 is the SCI interrupt mask bit which can be set or cleared by software. When it is "1", the SCI interrupt (SSR7) is masked When reset, it is set to "1".

Bit 4 (SSR4)

Bit 4 is the TIMER$_2$ interrupt mask bit which can be set or cleared by software. When the bit is "1", the TIMER$_2$ interrupt (SSR6) is masked. When reset, it is set to "1".

Bit 3 (SSR3)

When "1" is written in this bit, the prescaler of the transfer clock generator is initialized. When read, the bit always is "0".

Bits 2 ~ 0

Not used.

SSR7	SCI interrupt request
0	Absent
1	Present

SSR6	TIMER$_2$ interrupt request
0	Absent
1	Present

SSR5	SCI interrupt mask
0	Enabled
1	Disabled

SSR4	TIMER$_2$ interrupt mask
0	Enabled
1	Disabled

• Data Transmission

By writing the desired control bits into the SCI control registers, a transfer rate and a source of transfer clock are determined and bits 7 and 5 of port C are set at the serial data output terminal and the serial clock terminal, respectively. The transmit data should be stored from the accumulator or index register into the SCI data register. The data written in the SCI data register is output from the C_7/Tx terminal, starting with the LSB, synchronously with the falling edge of the serial clock. (See Fig. 16.) When 8 bit of data have been transmitted, the interrupt request bit is set in bit 7 of the SCI status register with the rising edge of the last serial clock. This request can be masked by setting bit 5 of the SCI status register. Once the data has been sent, the 8th bit data (MSB) stays at the C7/Tx terminal. If an external clock source has been selected, the transfer rate determined by bits 0 ~ 3 of the SCI control register is ignored, and the C5/$\overline{\text{CK}}$ terminal is set as input. If the internal clock has been selected, the C5/$\overline{\text{CK}}$ terminal is set as output and clocks are output at the transfer rate selected by bits 0 ~ 3 of the SCI control register.

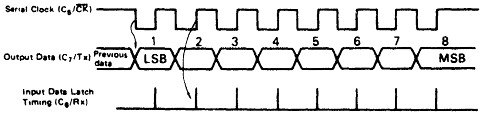

Serial Clock (C5/$\overline{\text{CK}}$)
Output Data (C7/Tx)
Previous data
1
2
3
4
5
6
7
8
LSB
MSB
Input Data Latch Timing (C6/Rx)

Figure 16 SCI Timing Chart

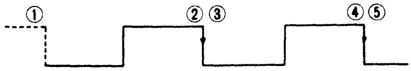
- **Data Reception**

By writing the desired control bits into the SCI control register, a transfer rate and a source of transfer clock are determined and bit 6 and 5 of port C are set at the serial data input terminal and the serial clock terminal, respectively. Then dummy-writing or -reading the SCI data register, the system is ready for receiving data. (This procedure is not needed after reading subsequent received data. It must be taken after reset and after not reading subsequent received data.

The data from the C6/Rx terminal is input to the SCI data register synchronously with the rising edge of the serial clock (see Fig. 16). When 8 bits of data have been received, the interrupt request bit is set in bit 7 of the SCI status register. This request can be masked by setting bit 5 of the SCI status register. If an external clock source have been selected, the transfer rate determined by bits 0 ~ 3 of the SCI control register is ignored and the data is received synchronously with the clock from the C5/$\overline{\text{CK}}$ terminal. If the internal clock has been selected, the C5/$\overline{\text{CK}}$ terminal is set as output and clocks are output at the transfer rate selected by bits 0 ~ 3 of the SCI control register.

- **TIMER2**

The SCI transfer clock generator can be used as a timer. The clock selected by bits 3 ~ 0 of the SCI control register (4 μs ~ approx. 32 ms (for oscillation at 4 MHz)) is input to bit 6 of the SCI status register and the TIMER2 interrupt request bit is set at each falling edge of the clock. Since interrupt requests occur periodically, TIMER2 can be used as a reload counter or clock.



① : Transfer clock generator is reset and mask bit (bit 4 of SCI status register) is cleared.
②, ④ : TIMER2 interrupt request
③, ⑤ : TIMER2 interrupt request bit cleared

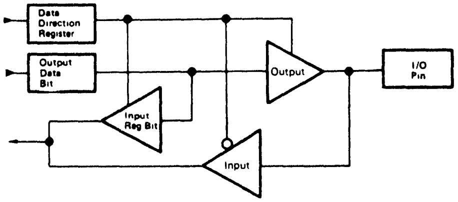
TIMER2 is commonly used with the SCI transfer clock generator. If wanting to use TIMER2 independently of the SCI, specify "External" (SCR5 = 1, SCR4 = 1) as the SCI clock source.

If "Internal" is selected as the clock source, reading or writing the SDR causes the prescaler of the transfer clock generator to be initialized.

■I/O PORTS

There are 24 input/output terminals (ports A, B, C). Each I/O terminal can be selected for either input or output by the data direction register. More specifically, an I/O port will be input if "0" is written in the data direction register, and output if "1" is written in the data direction register. Port A, B or C reads latched data if it has been programmed as output, even with the output level being fluctuated by the output load. (See Fig. 17.)

When reset, the data direction register and data register go to "0" and all the input/output terminals are used as input.



Bit of data direction register	Bit of output data	Status of output	Input to CPU
1	0	0	0
1	1	1	1
0	X	3-state	Pin

Figure 17 Input/Output Port Diagram

Seven input-only terminals are available (port D). Writing to an input terminal is invalid.

All input/output terminals and input terminals are TTL compatible and CMOS compatible in respect of both input and output.

If I/O ports or input ports are not used, they should be connected to V_{SS} via resistors. With none connected to these terminals, there is the possibility of power being consumed despite that they are not used.

■RESET

The MCU can be reset either by external reset input ($\overline{\text{RES}}$) or power-on reset. (See Fig. 18.) On power up, the reset input must be held "Low" for at least t_{OSC} to assure that the internal oscillator is stabilized. A sufficient time of delay can be obtained by connecting a capacitance to the $\overline{\text{RES}}$ input as shown in Fig. 19.

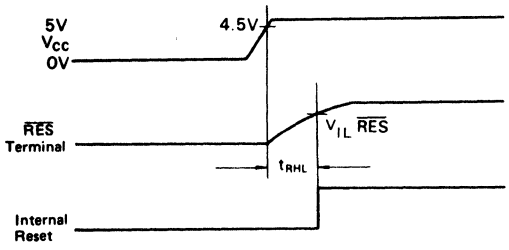


Figure 18 Power On and Reset Timing

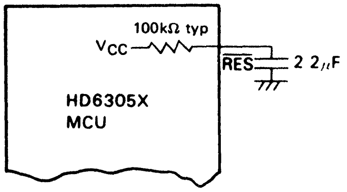
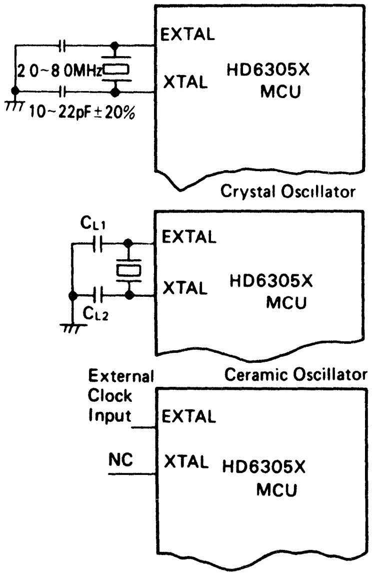


Figure 19 Input Reset Delay Circuit

■INTERNAL OSCILLATOR

The internal oscillator circuit is designed to meet the requirement for minimum external configurations. It can be driven by connecting a crystal (AT cut 2.0 ~ 8.0MHz) or ceramic oscillator between pins 5 and 6 depending on the required oscillation frequency stability.

Three different terminal connections are shown in Fig. 20. Figs. 21 and 22 illustrate the specifications and typical arrangement of the crystal, respectively.



Crystal Oscillator

Ceramic Oscillator

External Clock Drive

Figure 20 Internal Oscillator Circuit

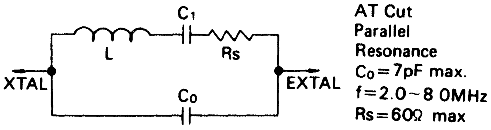
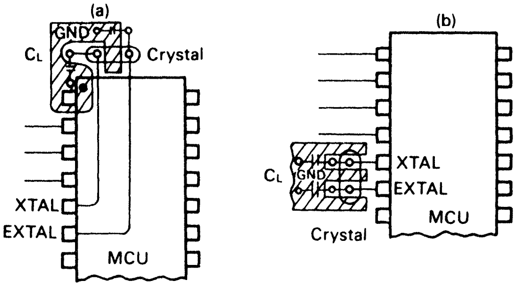


Figure 21 Parameters of Crystal



[NOTE] Use as short wirings as possible for connection of the crystal with the EXTAL and XTAL terminals. Do not allow these wirings to cross others.

Figure 22 Typical Crystal Arrangement

■LOW POWER DISSIPATION MODE

The HD6305X has three low power dissipation modes: wait, stop and standby.

• Wait Mode

When WAIT instruction being executed, the MCU enters into the wait mode. In this mode, the oscillator stays active but the internal clock stops. The CPU stops but the peripheral functions – the timer and the serial communication interface – stay active. (NOTE: Once the system has entered the wait mode, the serial communication interface can no longer be retriggered.) In the wait mode, the registers, RAM and I/O terminals hold their condition just before entering into the wait mode.

The escape from this mode can be done by interrupt ($\overline{\text{INT}}$, TIMER/$\overline{\text{INT}_2}$ or SCI/TIMER$_2$), $\overline{\text{RES}}$ or $\overline{\text{STBY}}$. The $\overline{\text{RES}}$ resets the MCU and the $\overline{\text{STBY}}$ brings it into the standby mode. (This will be mentioned later.)

When interrupt is requested to the CPU and accepted, the wait mode escapes, then the CPU is brought to the operation mode and vectors to the interrupt routine. If the interrupt is masked by the I bit of the condition code register, after releasing from the wait mode the MCU executes the instruction next to the WAIT. If an interrupt other than the $\overline{\text{INT}}$ (i.e., TIMER/$\overline{\text{INT}_2}$ or SCI/TIMER$_2$) is masked by the timer control register, miscellaneous register or serial status register, there is no interrupt request to the CPU, so the wait mode cannot be released.

Fig. 23 shows a flowchart for the wait function.

• Stop Mode

When STOP instruction being executed, MCU enters into the stop mode. In this mode, the oscillator stops and the CPU and peripheral functions become inactive but the RAM, registers and I/O terminals hold their condition just before entering into the stop mode.

The escape from this mode can be done by an external interrupt ($\overline{\text{INT}}$ or $\overline{\text{INT}_2}$), $\overline{\text{RES}}$ or $\overline{\text{STBY}}$. The $\overline{\text{RES}}$ resets the MCU and the $\overline{\text{STBY}}$ brings into the standby mode.

When interrupt is requested to the CPU and accepted, the stop mode escapes, then the CPU is brought to the operation mode and vectors to the interrupt routine. If the interrupt is masked by the I bit of the condition code register, after releasing from the stop mode, the MCU executes the instruction next to the STOP. If the $\overline{\text{INT}_2}$ interrupt is masked by the miscellaneous register, there is no interrupt request to the MCU, so the stop mode cannot be released.

Fig. 24 shows a flowchart for the stop function. Fig. 25 shows a timing chart of return to the operation mode from the stop mode.

For releasing from the stop mode by an interrupt, oscillation starts upon input of the interrupt and, after the internal delay time for stabilized oscillation, the CPU becomes active. For restarting by $\overline{\text{RES}}$, oscillation starts when the $\overline{\text{RES}}$ goes "0" and the CPU restarts when the $\overline{\text{RES}}$ goes "1". The duration of $\overline{\text{RES}}$="0" must exceed t_{OSC} to assure stabilized oscillation.

• Standby Mode

The MCU enters into the standby mode when the $\overline{\text{STBY}}$ terminal goes "Low". In this mode, all operations stop and the internal condition is reset but the contents of the RAM are hold. The I/O terminals turn to high-impedance state. The standby mode should escape by bringing $\overline{\text{STBY}}$ "High". The CPU must be restarted by reset. The timing of input signals at the $\overline{\text{RES}}$ and $\overline{\text{STBY}}$ terminals is shown in Fig. 26.

Table 4 lists the status of each parts of the MCU in each low power dissipation modes. Transitions between each mode are shown in Fig. 27.

(Note)

When I bit of condition code register is "1" and interrupt ($\overline{\text{INT}}$, TIMER/$\overline{\text{INT}_2}$, SCI/TIMER$_2$) is held, MCU does not enter WAIT mode by the execution of WAIT instruction.

In that case, after the 4 dummy cycles MCU executes the next instruction.

In the same way, when external interrupts ($\overline{\text{INT}}$, $\overline{\text{INT}_2}$) are held at the bit I set, MCU does not enter STOP mode by the execution of STOP instruction. In that case, also, MCU executes the next instruction after the 4 dummy cycles.

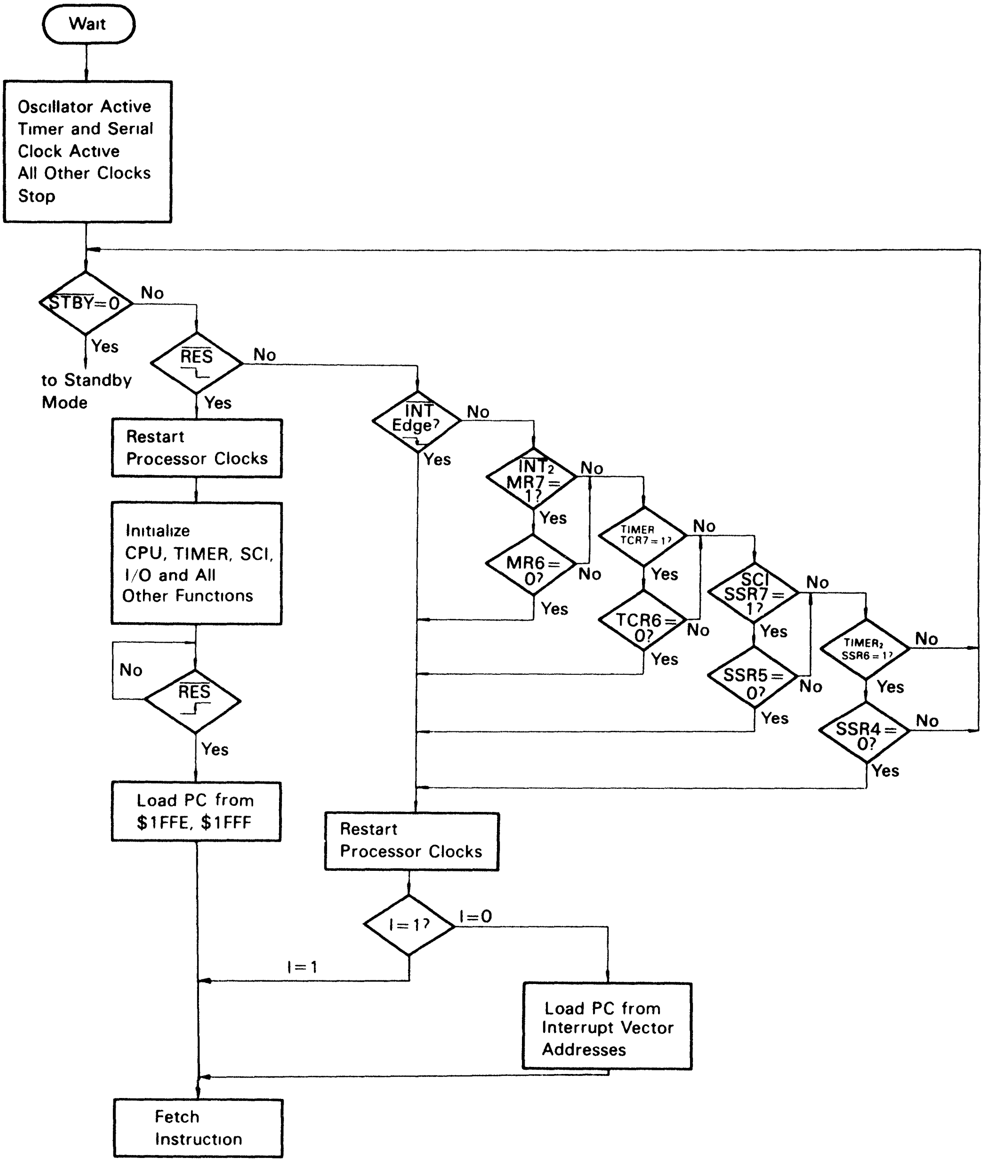


Figure 23 Wait Mode Flow Chart

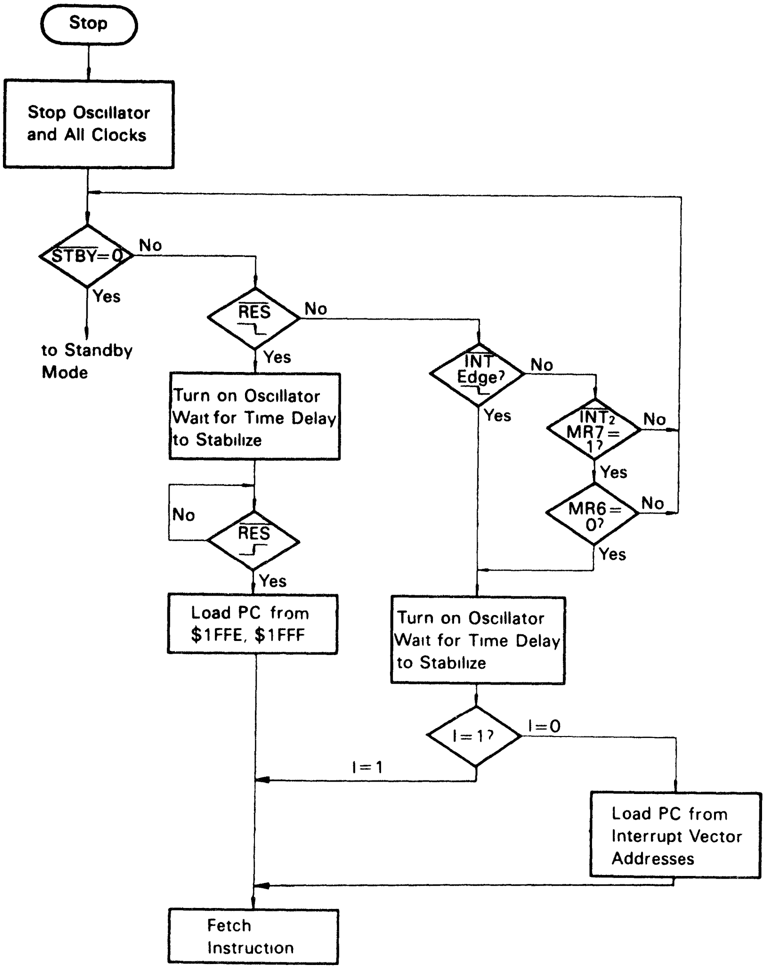
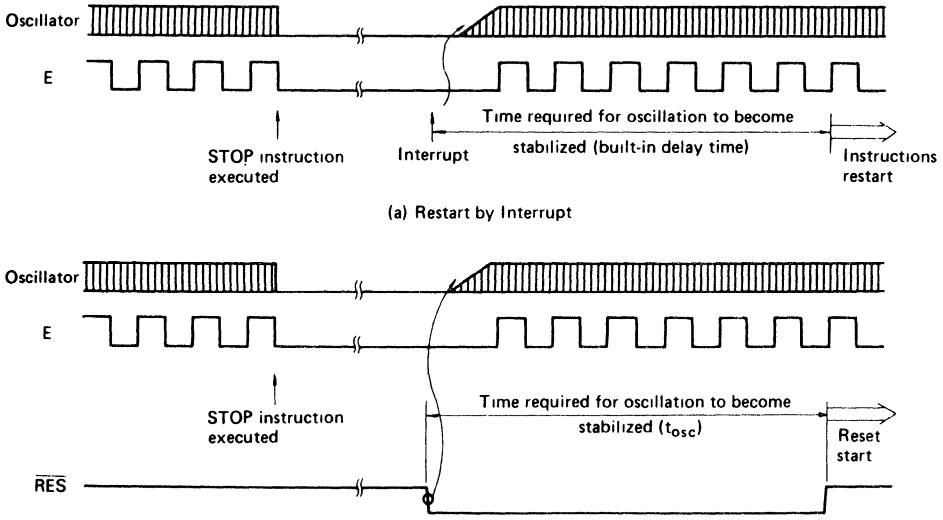

Stop
Stop Oscillator and All Clocks
STBY=0
No
Yes
to Standby Mode
RES
No
Yes
Turn on Oscillator Wait for Time Delay to Stabilize
No
RES
Yes
Load PC from $1FFE, $1FFF
INT Edge?
No
Yes
INT2 MR7=1?
No
Yes
MR6=0?
No
Yes
Turn on Oscillator Wait for Time Delay to Stabilize
I=1?
I=0
I=1
Load PC from Interrupt Vector Addresses
Fetch Instruction

Figure 24 Stop Mode Flow Chart


Oscillator
E
STOP instruction executed
Interrupt
Time required for oscillation to become stabilized (built-in delay time)
Instructions restart
Oscillator
E
STOP instruction executed
Time required for oscillation to become stabilized (t_{OSC})
Reset start
$\overline{\text{RES}}$

(a) Restart by Interrupt

(b) Restart by Reset

Figure 25 Timing Chart of Releasing from Stop Mode

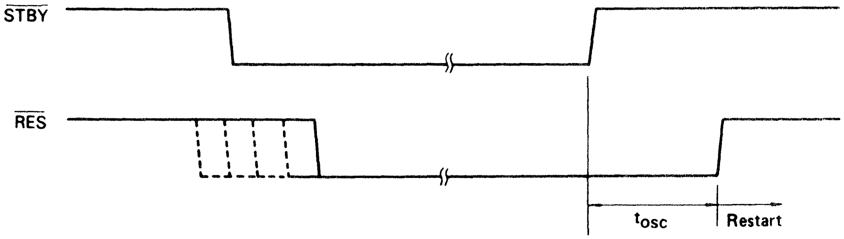


Figure 26 Timing Chart of Releasing from Standby Mode

Table 4 Status of Each Part of MCU in Low Power Dissipation Modes

Mode	Start		Condition						Escape
			Oscil-lator	CPU	Timer, Serial	Register	RAM	I/O terminal	
WAIT	Soft-ware	WAIT instruction	Active	Stop	Active	Keep	Keep	Keep	$\overline{\text{STBY}}$, $\overline{\text{RES}}$, $\overline{\text{INT}}$, $\overline{\text{INT}_2}$, each interrupt request of TIMER, TIMER$_2$, SCI
STOP		STOP instruction	Stop	Stop	Stop	Keep	Keep	Keep	$\overline{\text{STBY}}$, $\overline{\text{RES}}$, $\overline{\text{INT}}$, $\overline{\text{INT}_2}$
Stand-by	Hard-ware	$\overline{\text{STBY}}$="Low"	Stop	Stop	Stop	Reset	Keep	High impedance	$\overline{\text{STBY}}$="High"

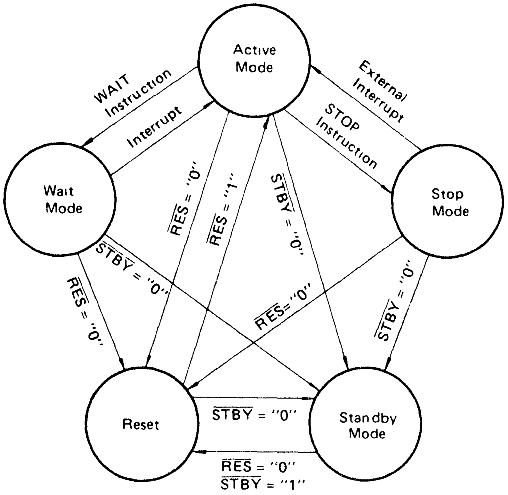


Figure 27 Transitions among Active Mode, Wait Mode, Stop Mode, Standby Mode and Reset

■BIT MANIPULATION

The MCU can use a single instruction (BSET or BCLR) to set or clear one bit of the RAM or an I/O port (except the write-only registers such as the data direction register). Every bit of memory or I/O within page 0 ($00 ~ $FF) can be tested by the BRSET or BRCLR instruction, depending on the result of the test, the program can branch to required destinations Since bits in the RAM, or I/O can be manipulated, the user may use a bit within the RAM as a flag or handle a single I/O bit as an independent I/O terminal. Fig 28 shows an example of bit manipulation and the validity of test instructions In the example, the program is configured assuming that bit 0 of port A is connected to a zero cross detector circuit and bit 1 of the same port to the trigger of a triac

The program shown can activate the triac within a time of 10μs from zero-crossing through the use of only 7 bytes on the ROM. The on-chip timer provides a required time of delay and pulse width modulation of power is also possible.

```
             :
SELF 1    BRCLR 0, PORT A, SELF 1
          BSET 1, PORT A
          BCLR 1, PORT A
             :
```

Figure 28 Example of Bit Manipulation

■ADDRESSING MODES

Ten different addressing modes are available to the MCU

• Immediate

See Fig. 29. The immediate addressing mode provides access to a constant which does not vary during execution of the program.

This access requires an instruction length of 2 bytes. The effective address (EA) is PC and the operand is fetched from the byte that follows the operation code

• Direct

See Fig. 30. In the direct addressing mode, the address of the operand is contained in the 2nd byte of the instruction The user can gain direct access to memory up to the lower 255th address All RAM and I/O registers are on page 0 of address space so that the direct addressing mode may be utilized.

• Extended

See Fig. 31. The extended addressing is used for referencing to all addresses of memory. The EA is the contents of the 2 bytes that follow the operation code An extended addressing instruction requires a length of 3 bytes

• Relative

See Fig. 32. The relative addressing mode is used with branch instructions only. When a branch occurs, the program counter is loaded with the contents of the byte following the operation code. EA = (PC) + 2 + Rel., where Rel. indicates a signed 8-bit data following the operation code. If no branch occurs, Rel = 0. When a branch occurs, the program jumps to any byte in the range +129 to −127. A branch instruction requires a length of 2 bytes

• Indexed (No Offset)

See Fig 33 The indexed addressing mode allows access up to the lower 255th address of memory. In this mode, an instruction requires a length of one byte. The EA is the contents of the index register.

• Indexed (8-bit Offset)

See Fig. 34. The EA is the contents of the byte following the operation code, plus the contents of the index register. This mode allows access up to the lower 511th address of memory. Each instruction when used in the index addressing mode (8-bit offset) requires a length of 2 bytes.

• Indexed (16-bit Offset)

See Fig. 35. The contents of the 2 bytes following the operation code are added to content of the index register to compute the value of EA. In this mode, the complete memory can be accessed. When used in the indexed addressing mode (16-bit offset), an instruction must be 3 bytes long.

• Bit Set/Clear

See Fig. 36. This addressing mode is applied to the BSET and BCLR instructions that can set or clear any bit on page 0. The lower 3 bits of the operation code specify the bit to be set or cleared. The byte that follows the operation code indicates an address within page 0.

• Bit Test and Branch

See Fig. 37. This addressing mode is applied to the BRSET and BRCLR instructions that can test any bit within page 0 and can be branched in the relative addressing mode. The byte to be tested is addressed depending on the contents of the byte following the operation code. Individual bits within the byte to be tested are specified by the lower 3 bits of the operation code. The 3rd byte represents a relative value which will be added to the program counter when a branch condition is established. Each of these instructions should be 3 bytes long. The value of the test bit is written in the carry bit of the condition code register.

• Implied

See Fig. 38. This mode involves no EA. All information needed for execution of an instruction is contained in the operation code. Direct manipulation on the accumulator and index register is included in the implied addressing mode. Other instructions such as SWI and RTI are also used in this mode. All instructions used in the implied addressing mode should have a length of one byte.

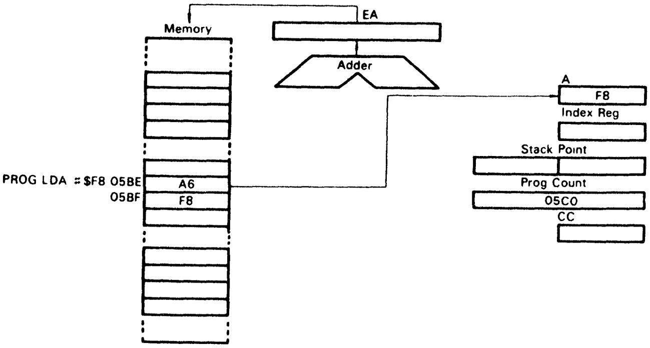


Figure 29 Example of Immediate Addressing

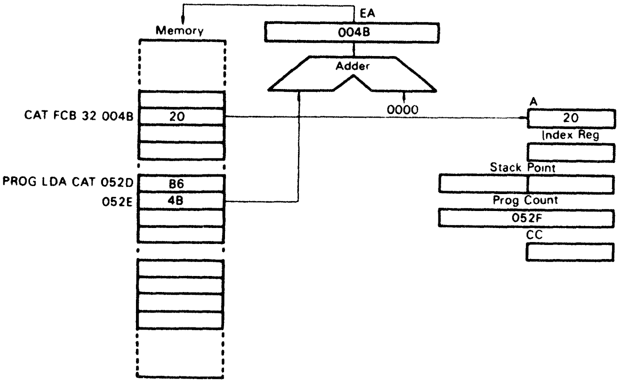


Figure 30 Example of Direct Addressing

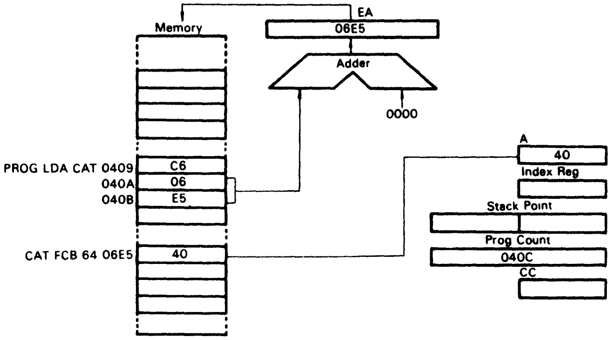


Figure 31 Example of Extended Addressing

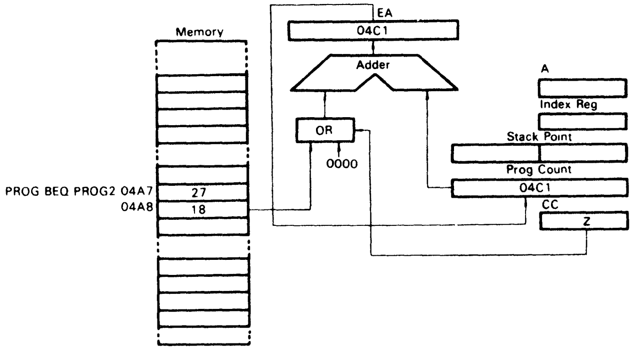


Figure 32 Example of Relative Addressing

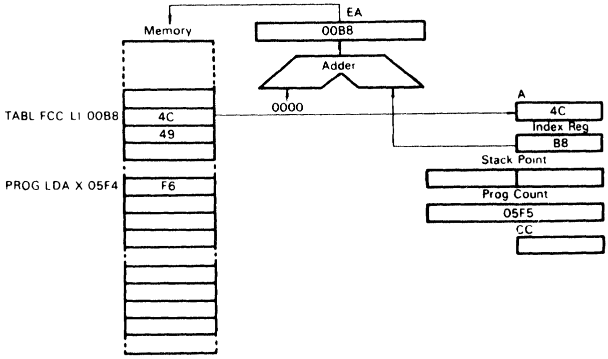


Figure 33 Example of Indexed (No Offset) Addressing

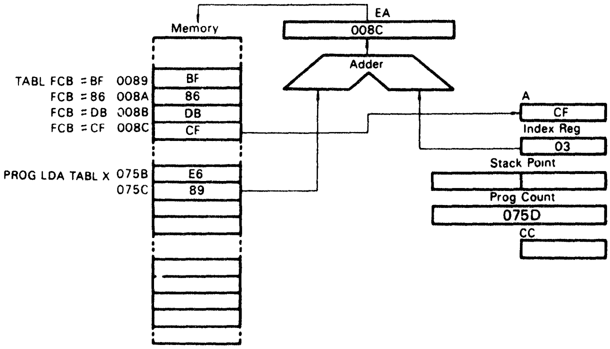


Figure 34 Example of Index (8-bit Offset) Addressing

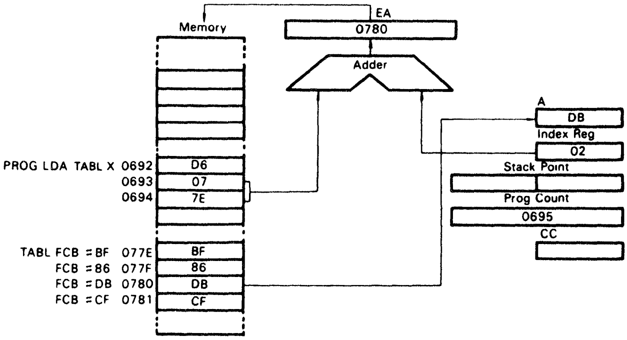


Figure 35 Example of Index (16-bit Offset) Addressing

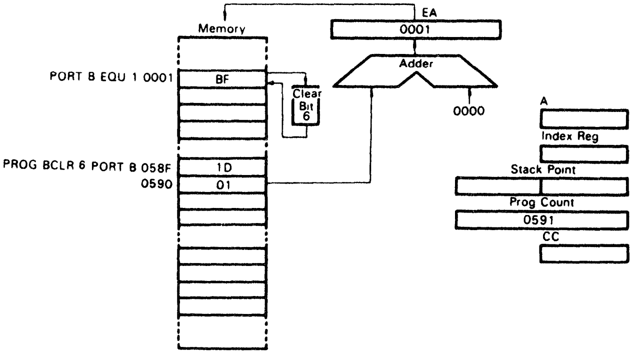


Figure 36 Example of Bit Set/Clear Addressing

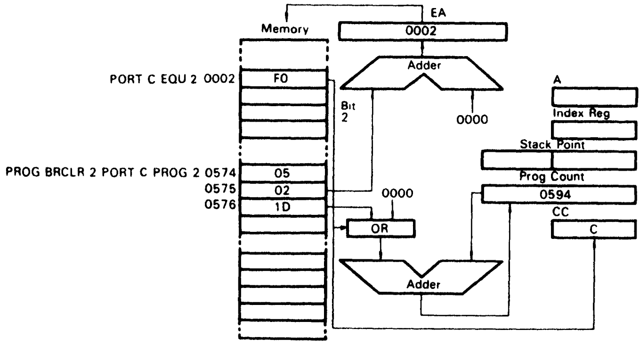


Figure 37 Example of Bit Test and Branch Addressing

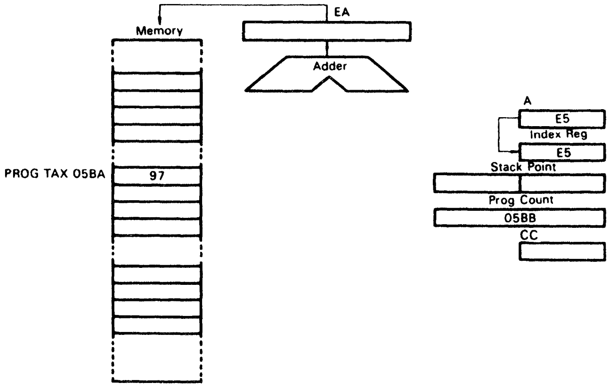


Figure 38 Example of Implied Addressing

■INSTRUCTION SET

There are 62 basic instructions available to the HD6305X MCU. They can be classified into five categories: register/memory, read/modify/write, branch, bit manipulation, and control. The details of each instruction are described in Tables 5 through 11.

• Register/Memory Instructions

Most of these instructions use two operands. One operand is either an accumulator or index register. The other is derived from memory using one of the addressing modes used on the HD6305X MCU. There is no register operand in the unconditional jump instruction (JMP) and the subroutine jump instruction (JSR). See Table 5.

• Read/Modify/Write Instructions

These instructions read a memory or register, then modify or test its contents, and write the modified value into the memory or register. Zero test instruction (TST) does not write data, and is handled as an exception in the read/modify/write group. See Table 6.

• Branch Instructions

A branch instruction branches from the program sequence in progress if a particular condition is established. See Table 7.

• Bit Manipulation Instructions

These instructions can be used with any bit located up to the lower 255th address of memory. Two groups are available; one for setting or clearing and the other for bit testing and branching. See Table 8.

• Control Instructions

The control instructions control the operation of the MCU which is executing a program. See Table 9.

• List of Instructions in Alphabetical Order

Table 10 lists all the instructions used on the HD6305X MCU in the alphabetical order.

• Operation Code Map

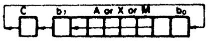
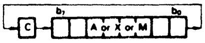
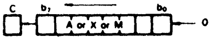
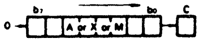
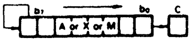
Table 11 shows the operation code map for the instructions used on the MCU.

Table 5 Register/Memory Instructions

Operations	Mnemonic	Addressing Modes																		Boolean/ Arithmetic Operation	Condition Code				
		Immediate			Direct			Extended			Indexed (No Offset)			Indexed (8-Bit Offset)			Indexed (16-Bit Offset)								
		OP	#	~	OP	#	~	OP	#	~	OP	#	~	OP	#	~	OP	#	~		H	I	N	Z	C
Load A from Memory	LDA	A6	2	2	B6	2	3	C6	3	4	F6	1	3	E6	2	4	D6	3	5	M→A	●	●	∧	∧	●
Load X from Memory	LDX	AE	2	2	BE	2	3	CE	3	4	FE	1	3	EE	2	4	DE	3	5	M→X	●	●	∧	∧	●
Store A in Memory	STA	–	–	–	B7	2	3	C7	3	4	F7	1	4	E7	2	4	D7	3	5	A→M	●	●	∧	∧	●
Store X in Memory	STX	–	–	–	BF	2	3	CF	3	4	FF	1	4	EF	2	4	DF	3	5	X→M	●	●	∧	∧	●
Add Memory to A	ADD	AB	2	2	BB	2	3	CB	3	4	FB	1	3	EB	2	4	DB	3	5	A+M→A	∧	●	∧	∧	∧
Add Memory and Carry to A	ADC	A9	2	2	B9	2	3	C9	3	4	F9	1	3	E9	2	4	D9	3	5	A+M+C→A	∧	●	∧	∧	∧
Subtract Memory	SUB	A0	2	2	B0	2	3	C0	3	4	F0	1	3	E0	2	4	D0	3	5	A−M→A	●	●	∧	∧	∧
Subtract Memory from A with Borrow	SBC	A2	2	2	B2	2	3	C2	3	4	F2	1	3	E2	2	4	D2	3	5	A−M−C→A	●	●	∧	∧	∧
AND Memory to A	AND	A4	2	2	B4	2	3	C4	3	4	F4	1	3	E4	2	4	D4	3	5	A M→A	●	●	∧	∧	●
OR Memory with A	ORA	AA	2	2	BA	2	3	CA	3	4	FA	1	3	EA	2	4	DA	3	5	A+M→A	●	●	∧	∧	●
Exclusive OR Memory with A	EOR	A8	2	2	B8	2	3	C8	3	4	F8	1	3	E8	2	4	D8	3	5	A⊕M→A	●	●	∧	∧	●
Arithmetic Compare A with Memory	CMP	A1	2	2	B1	2	3	C1	3	4	F1	1	3	E1	2	4	D1	3	5	A−M	●	●	∧	∧	∧
Arithmetic Compare X with Memory	CPX	A3	2	2	B3	2	3	C3	3	4	F3	1	3	E3	2	4	D3	3	5	X−M	●	●	∧	∧	∧
Bit Test Memory with A (Logical Compare)	BIT	A5	2	2	B5	2	3	C5	3	4	F5	1	3	E5	2	4	D5	3	5	A M	●	●	∧	∧	●
Jump Unconditional	JMP				BC	2	2	CC	3	3	FC	1	2	EC	2	3	DC	3	4		●	●	●	●	●
Jump to Subroutine	JSR				BD	2	5	CD	3	6	FD	1	5	ED	2	5	DD	3	6		●	●	●	●	●

Symbols Op = Operation
 # = Number of bytes
 ~ = Number of cycles

Table 6 Read/Modify/Write Instructions

Operations	Mnemonic	Addressing Modes															Boolean/Arithmetic Operation	Condition Code				
		Implied(A)			Implied(X)			Direct			Indexed (No Offset)			Indexed (8 Bit Offset)								
		OP	#	~	OP	#	~	OP	#	~	OP	#	~	OP	#	~		H	I	N	Z	C
Increment	INC	4C	1	2	5C	1	2	3C	2	5	7C	1	5	6C	2	6	A+1→A or X+1 →X or M+1 →M	●	●	∧	∧	●
Decrement	DEC	4A	1	2	5A	1	2	3A	2	5	7A	1	5	6A	2	6	A−1 →A or X−1→X or M−1→M	●	●	∧	∧	●
Clear	CLR	4F	1	2	5F	1	2	3F	2	5	7F	1	5	6F	2	6	00→A or 00→X or 00→M	●	●	0	1	●
Complement	COM	43	1	2	53	1	2	33	2	5	73	1	5	63	2	6	$\overline{A}$→A or $\overline{X}$→X or $\overline{M}$→M	●	●	∧	∧	1
Negate (2's Complement)	NEG	40	1	2	50	1	2	30	2	5	70	1	5	60	2	6	00−A→A or 00−X→X or 00−M →M	●	●	∧	∧	∧
Rotate Left Thru Carry	ROL	49	1	2	59	1	2	39	2	5	79	1	5	69	2	6		●	●	∧	∧	∧
Rotate Right Thru Carry	ROR	46	1	2	56	1	2	36	2	5	76	1	5	66	2	6		●	●	∧	∧	∧
Logical Shift Left	LSL	48	1	2	58	1	2	38	2	5	78	1	5	68	2	6		●	●	∧	∧	∧
Logical Shift Right	LSR	44	1	2	54	1	2	34	2	5	74	1	5	64	2	6		●	●	0	∧	∧
Arithmetic Shift Right	ASR	47	1	2	57	1	2	37	2	5	77	1	5	67	2	6		●	●	∧	∧	∧
Arithmetic Shift Left	ASL	48	1	2	58	1	2	38	2	5	78	1	5	68	2	6	Equal to LSL	●	●	∧	∧	∧
Test for Negative or Zero	TST	4D	1	2	5D	1	2	3D	2	4	7D	1	4	6D	2	5	A−00 or X−00 or M−00	●	●	∧	∧	●

Symbols Op = Operation
 # = Number of bytes
 ~ = Number of cycles

Table 7 Branch Instructions

Operations	Mnemonic	Addressing Modes			Branch Test	Condition Code				
		Relative								
		OP	#	~		H	I	N	Z	C
Branch Always	BRA	20	2	3	None	●	●	●	●	●
Branch Never	BRN	21	2	3	None	●	●	●	●	●
Branch IF Higher	BHI	22	2	3	C+Z=0	●	●	●	●	●
Branch IF Lower or Same	BLS	23	2	3	C+Z=1	●	●	●	●	●
Branch IF Carry Clear	BCC	24	2	3	C=0	●	●	●	●	●
(Branch IF Higher or Same)	(BHS)	24	2	3	C=0	●	●	●	●	●
Branch IF Carry Set	BCS	25	2	3	C=1	●	●	●	●	●
(Branch IF Lower)	(BLO)	25	2	3	C=1	●	●	●	●	●
Branch IF Not Equal	BNE	26	2	3	Z=0	●	●	●	●	●
Branch IF Equal	BEQ	27	2	3	Z=1	●	●	●	●	●
Branch IF Half Carry Clear	BHCC	28	2	3	H=0	●	●	●	●	●
Branch IF Half Carry Set	BHCS	29	2	3	H=1	●	●	●	●	●
Branch IF Plus	BPL	2A	2	3	N=0	●	●	●	●	●
Branch IF Minus	BMI	2B	2	3	N=1	●	●	●	●	●
Branch IF Interrupt Mask Bit is Clear	BMC	2C	2	3	I=0	●	●	●	●	●
Branch IF Interrupt Mask Bit is Set	BMS	2D	2	3	I=1	●	●	●	●	●
Branch IF Interrupt Line is Low	BIL	2E	2	3	INT=0	●	●	●	●	●
Branch IF Interrupt Line is High	BIH	2F	2	3	INT=1	●	●	●	●	●
Branch to Subroutine	BSR	AD	2	5	—	●	●	●	●	●

Symbols. Op = Operation
# = Number of bytes
~ = Number of cycles

Table 8 Bit Manipulation Instructions

Operations	Mnemonic	Addressing Modes						Boolean/ Arithmetic Operation	Branch Test	Condition Code				
		Bit Set/Clear			Bit Test and Branch									
		OP	#	~	OP	#	~			H	I	N	Z	C
Branch IF Bit n is set	BRSET n(n=0 7)	—	—	—	2·n	3	5	—	Mn=1	●	●	●	●	∧
Branch IF Bit n is clear	BRCLR n(n=0· 7)	—	—	—	01+2·n	3	5	—	Mn=0	●	●	●	●	∧
Set Bit n	BSET n(n=0 · 7)	10+2·n	2	5	—	—	—	1→Mn	—	●	●	●	●	●
Clear Bit n	BCLR n(n=0·· 7)	11+2·n	2	5	—	—	—	0→Mn	—	●	●	●	●	●

Symbols Op = Operation
# = Number of bytes
~ = Number of cycles

Table 9 Control Instructions

Operations	Mnemonic	Addressing Modes			Boolean Operation	Condition Code				
		Implied								
		OP	#	~		H	I	N	Z	C
Transfer A to X	TAX	97	1	2	A→X	●	●	●	●	●
Transfer X to A	TXA	9F	1	2	X→A	●	●	●	●	●
Set Carry Bit	SEC	99	1	1	1→C	●	●	●	●	1
Clear Carry Bit	CLC	98	1	1	0→C	●	●	●	●	0
Set Interrupt Mask Bit	SEI	9B	1	2	1→I	●	1	●	●	●
Clear Interrupt Mask Bit	CLI	9A	1	2	0→I	●	0	●	●	●
Software Interrupt	SWI	83	1	10		●	1	●	●	●
Return from Subroutine	RTS	81	1	5		●	●	●	●	●
Return from Interrupt	RTI	80	1	8		?	?	?	?	?
Reset Stack Pointer	RSP	9C	1	2	$FF→SP	●	●	●	●	●
No-Operation	NOP	9D	1	1	Advance Prog Cntr Only	●	●	●	●	●
Decimal Adjust A	DAA	8D	1	2	Converts binary add of BCD charcters into BCD format	●	●	∧	∧	∧*
Stop	STOP	8E	1	4		●	●	●	●	●
Wait	WAIT	8F	1	4		●	●	●	●	●

Symbols. Op = Operation
# = Number of bytes
~ = Number of cycles

* Are BCD characters of upper byte 10 or more? (They are not cleared if set in advance)

Table 10 Instruction Set (in Alphabetical Order)

Mnemonic	Addressing Modes										Condition Code				
	Implied	Immediate	Direct	Extended	Relative	Indexed (No Offset)	Indexed (8-Bit)	Indexed (16-Bit)	Bit Set/ Clear	Bit Test & Branch	H	I	N	Z	C
ADC		×	×	×		×	×	×			∧	●	∧	∧	∧
ADD		×	×	×		×	×	×			∧	●	∧	∧	∧
AND		×	×	×		×	×	×			●	●	∧	∧	●
ASL	×		×			×	×				●	●	∧	∧	∧
ASR	×		×			×	×				●	●	∧	∧	∧
BCC					×						●	●	●	●	●
BCLR									×		●	●	●	●	●
BCS					×						●	●	●	●	●
BEQ					×						●	●	●	●	●
BHCC					×						●	●	●	●	●
BHCS					×						●	●	●	●	●
BHI					×						●	●	●	●	●
(BHS)					×						●	●	●	●	●
BIH					×						●	●	●	●	●
BIL					×						●	●	●	●	●
BIT		×	×	×		×	×	×			●	●	∧	∧	●
(BLO)					×						●	●	●	●	●
BLS					×						●	●	●	●	●
BMC					×						●	●	●	●	●
BMI					×						●	●	●	●	●
BMS					×						●	●	●	●	●
BNE					×						●	●	●	●	●
BPL					×						●	●	●	●	●
BRA					×						●	●	●	●	●

(to be continued)

Condition Code Symbols

H	Half Carry (From Bit 3)	C	Carry/Borrow
I	Interrupt Mask	∧	Test and Set if True, Cleared Otherwise
N	Negative (Sign Bit)	●	Not Affected
Z	Zero	?	Load CC Register From Stack

Table 10 Instruction Set (in Alphabetical Order)

Mnemonic	Addressing Modes										Condition Code				
	Implied	Immediate	Direct	Extended	Relative	Indexed (No Offset)	Indexed (8-Bit)	Indexed (16-Bit)	Bit Set, Clear	Bit Test & Branch	H	I	N	Z	C
BRN					×						●	●	●	●	●
BRCLR										×	●	●	●	●	∧
BRSET										×	●	●	●	●	∧
BSET									×		●	●	●	●	●
BSR					×						●	●	●	●	●
CLC	×										●	●	●	●	0
CLI	×										●	0	●	●	●
CLR	×		×			×	×				●	●	0	1	●
CMP		×	×	×		×	×	×			●	●	∧	∧	∧
COM	×		×			×	×				●	●	∧	∧	1
CPX		×	×	×		×	×	×			●	●	∧	∧	∧
DAA	×										●	●	∧	∧	∧
DEC	×		×			×	×				●	●	∧	∧	●
EOR		×	×	×		×	×	×			●	●	∧	∧	●
INC	×		×			×	×				●	●	∧	∧	●
JMP			×	×		×	×	×			●	●	●	●	●
JSR			×	×		×	×	×			●	●	●	●	●
LDA		×	×	×		×	×	×			●	●	∧	∧	●
LDX		×	×	×		×	×	×			●	●	∧	∧	●
LSL	×		×			×	×				●	●	∧	∧	∧
LSR	×		×			×	×				●	●	0	∧	∧
NEG	×		×			×	×				●	●	∧	∧	∧
NOP	×										●	●	●	●	●
ORA		×	×	×		×	×	×			●	●	∧	∧	●
ROL	×		×			×	×				●	●	∧	∧	∧
ROR	×		×			×	×				●	●	∧	∧	∧
RSP	×										●	●	●	●	●
RTI	×										?	?	?	?	?
RTS	×										●	●	●	●	●
SBC		×	×	×		×	×	×			●	●	∧	∧	∧
SEC	×										●	●	●	●	1
SEI	×										●	1	●	●	●
STA			×	×		×	×	×			●	●	∧	∧	●
STOP	×										●	●	●	●	●
STX			×	×		×	×	×			●	●	∧	∧	●
SUB		×	×	×		×	×	×			●	●	∧	∧	∧
SWI	×										●	1	●	●	●
TAX	×										●	●	●	●	●
TST	×		×			×	×				●	●	∧	∧	●
TXA	×										●	●	●	●	●
WAIT	×										●	●	●	●	●

Condition Code Symbols

H	Half Carry (From Bit 3)	C	Carry Borrow
I	Interrupt Mask	∧	Test and Set if True, Cleared Otherwise
N	Negative (Sign Bit)	●	Not Affected
Z	Zero	?	Load CC Register From Stack

Table 11 Operation Code Map

	Bit Manipulation		Branch	Read/Modify/Write					Control		Register/Memory							
	Test & Branch	Set/ Clear	Rel	DIR	A	X	,X1	,X0	IMP	IMP	IMM	DIR	EXT	,X2	,X1	,X0		
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	← HIGH	
0	BRSET0	BSET0	BRA	NEG					RTI*	—	SUB						0	
1	BRCLR0	BCLR0	BRN	—					RTS*	—	CMP						1	
2	BRSET1	BSET1	BHI	—					—	—	SBC						2	
3	BRCLR1	BCLR1	BLS	COM					SWI*	—	CPX						3	L
4	BRSET2	BSET2	BCC	LSR					—	—	AND						4	O
5	BRCLR2	BCLR2	BCS	—					—	—	BIT						5	W
6	BRSET3	BSET3	BNE	ROR					—	—	LDA						6	
7	BRCLR3	BCLR3	BEQ	ASR					—	TAX*	—	STA				STA(+1)	7	
8	BRSET4	BSET4	BHCC	LSL/ASL					—	CLC	EOR						8	
9	BRCLR4	BCLR4	BHCS	ROL					–	SEC	ADC						9	
A	BRSET5	BSET5	BPL	DEC					–	CLI*	ORA						A	
B	BRCLR5	BCLR5	BMI	—					—	SEI*	ADD						B	
C	BRSET6	BSET6	BMC	INC					-	RSP*	–	JMP(−1)					C	
D	BRCLR6	BCLR6	BMS	TST(−1)	TST		TST(−1)		DAA*	NOP	BSR*	JSR(+2)		JSR(+1)		JSR(+2)	D	
E	BRSET7	BSET7	BIL	—					STOP*	—	LDX						E	
F	BRCLR7	BCLR7	BIH	CLR					WAIT*	TXA*	–	STX				STX(+1)	F	
	3/5	2/5	2/3	2 5	1 2	1/2	2/6	1/5	1/*	1/1	2/2	2/3	3/4	3/5	2/4	1/3		

(NOTES) 1 "—" is an undefined operation code

2 The lowermost numbers in each column represent a byte count and the number of cycles required (byte count/number of cycles)
The number of cycles for the mnemonics asterisked (*) is a follows

RTI = 8	DAA = 2	TAX = 2	BSR = 5
RTS = 5	STOP = 4	RSP = 2	CLI = 2
SWI = 10	WAIT = 4	TXA = 2	SEI = 2

3. The parenthesized numbers must be added to the cycle count of the particular instruction

- **Additional Instructions**

The following new instructions are used on the HD6305X.

DAA Converts the contents of the accumulator into BCD code.

WAIT Causes the MCU to enter the wait mode. For this mode, see the topic, Wait Mode.

STOP Causes the MCU to enter the stop mode. For this mode, see the topic, Stop Mode.

■ PRECAUTION 1—BOARD DESIGN OF OSCILLATION CIRCUIT

When connecting crystal and ceramic resonator with the XTAL and EXTAL pins to oscillate, observe the following in designing the board.

(1) Locate crystal, ceramic resonator, and load capacity C_1 and C_2 as near the LSI as possible. (Induction of noise from outside to the XTAL and EXTAL pins may cause trouble in oscillation.)

(2) Wire the signal lines to the neighbouring XTAL and EXTAL pins as far apart as possible.

(3) Board design of situating signal lines or power supply lines near the oscillator circuit as shown in Fig. 40, should not be used because of trouble in oscillation in induction. The resistor between the XTAL and EXTAL, and pins close to them should be 10M Ω or more.

■ PRECAUTION 2—PROGRAM OF WRITE ONLY REGISTER

Read/Modify/Write instructions are unavailable for changing the contents of Write Only Register (e.g. DDR; Data Direction Register of I/O port) of HD6305X, HD6305Y and HD63P05Y.

(1) Data cannot be read from write only register. (e.g. DDR of I/O port)

While read/modify/write instructions are executed in the following sequence.

(i) Reads the contents from appointed address.

(ii) Changes the data which has been read.

(iii) Turn the data back to the original address.

Thus, read/modify/write instructions cannot be applied to write only register such as DDR.

(2) For the same reason, do not set DDR of I/O port using BSET and BCLR instructions.

(3) Stored instructions (e.g. STA and STX, etc.) are available for writing into the write only register.

■ PRECAUTION 3—SENDING/RECEIVING PROGRAM OF SERIAL DATA

Be careful that malfunction may occur if SDR (SERIAL DATA REGISTER: $0012) is read or written during transmitting or receiving serial data.

■ PRECAUTION 4—WAIT/STOP INSTRUCTIONS PROGRAM

When I bit of condition code register is "1" and an interrupt ($\overline{\text{INT}}_2$, TIMER/$\overline{\text{INT}}_2$) is held, the MCU does not enter into WAID mode by executing the WAIT instruction.

In that case, after the 4 dummy cycles, the MCU executes the next instruction.

In the same way, when external interrupts ($\overline{\text{INT}}$, $\overline{\text{INT}}_2$) are held at the bit I set, the MCU does not enter into the STOP mode by executing STOP instruction. In that case the MCU executes the next instruction after the 4 dummy cycles.

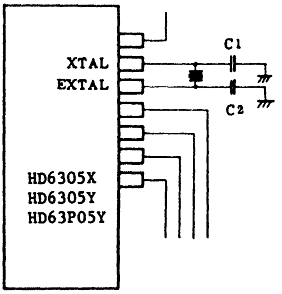


Figure 39 Design of Oscillation Circuit Board

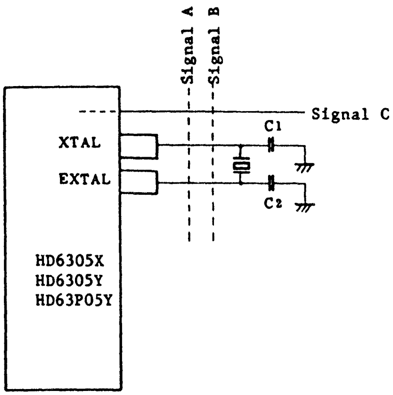


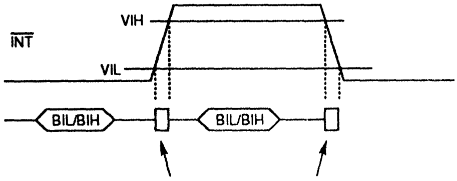
Figure 40 Example of Circuit Causing Trouble in Oscillation

■ PRECAUTION WHEN USING BIL/BIH INSTRUCTION

(1) Execute Instruction after the $\overline{INT}$ Voltage level has stabilized above V_{IH} or below V_{IL}.

(2) $\overline{INT}$ voltage level needs to be stabilized while BIL/BIH Instruction Execution.

There may be a malfunction by glitch on control signal if BIL/BIH Instruction Execution has exercized in unstablized $\overline{INT}$ signal level.



Avoid BIL/BIH Instruction Execution

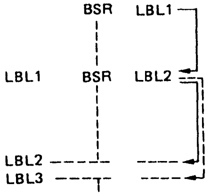
■ PRECAUTION TO USE BSR

If there is 2nd BSR programmed on the address which is directed by first BSR, 2nd BSR may not be executed correctly For this reason, BSR should not be programmed on the address which is directed by first BSR.

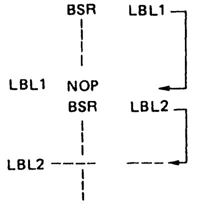
If necessary, please program as following.

(1) On the address which first BSR directed, NOP instruction should be inserted before second BSR.

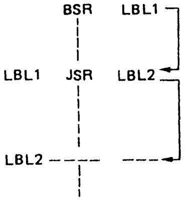
(2) On the address which first BSR directed, JSR instruction should be programmed instead of 2nd BSR.



example of malfunction
of 2nd BSR execution



example of counter measure
(NOP is inserted)



example of counter measure
(JSR is used instead of BSR)

HD6305Y2, HD63A05Y2, HD63B05Y2 — CMOS MCU (Microcomputer Unit)

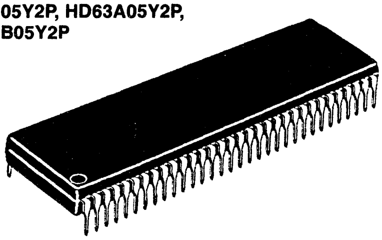
The HD6305Y2 is a CMOS 8-bit single chip microcomputer. A CPU, a clock generator, a 256 byte RAM, I/O terminals, two timers and a serial communication interface (SCI) are built in the HD6305Y2. Its memory space is expandable to 16k bytes externally.

The HD6305Y2 has the same function as the HD6305Y2's except for the number of I/O terminals. The HD6305Y2 is a microcomputer unit which includes no ROM and its memory space is expandable to 16k bytes externally

■ HARDWARE FEATURES

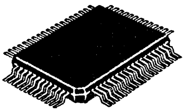
- 8-bit based MCU
- 256-bytes of RAM
- A total of 31 terminals, including 24 I/O's, 7 inputs
- Two timers
  - — 8-bit timer with a 7-bit prescaler (programmable prescaler, event counter)
  - — 15-bit timer (commonly used with the SCI clock divider)
- On-chip serial ınterface circuit (synchronized with clock)
- Six interrupts (two external, two timer, one serial and one software)
- Low power dissipation modes
  - — Wait . . . . . In this mode, the clock oscillator is on and the CPU halts but the timer/serial/interrupt function is operable.
  - — Stop . . . . . In this mode, the clock stops but the RAM data, I/O status and registers are held.
  - — Standby . . In this mode, the clock stops, the RAM data is held, and the other internal condition is reset.
- Minimum instruction cycle time
  - —HD6305Y2. . .1 μs (f = 1 MHz)
  - —HD63A05Y2. . .0.67 μs (f = 1.5 MHz)
  - —HD63B05Y2. . .0.5 μ (f = 2 MHz)
- Wide operating range
  - V_{CC} = 3 to 6V (f = 0.1 to 0.5 MHz)
  - —HD6305Y2 . . f = 0.1 to 1 MHz (V_{CC} = 5V ± 10%)
  - —HD63A05Y2. f = 0.1 to 1.5 MHz (V_{CC} = 5V ± 10%)
  - —HD63B05Y2. f = 0.1 to 2 MHz (V_{CC} = 5V ± 10%)

HD6305Y2P, HD63A05Y2P, HD63B05Y2P



(DP-64S)

HD6305Y2F, HD63A05Y2F, HD63B05Y2F



(FP-64)

■ SOFTWARE FEATURES

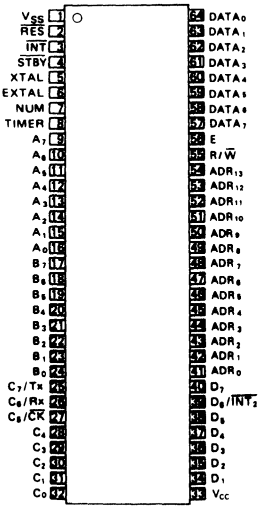
- Similar to HD6800
- Byte efficient instruction set
- Powerful bit manipulation instructions (Bit Set, Bit Clear, and Bit Test and Branch usable for 192 byte RAM bits within page 0 and I/O terminals)
- A variety of interrupt operations
- Index addressing mode useful for table processing
- A variety of conditional branch instructions
- Ten powerful addressing modes
- All addressing modes adaptable to RAM, and I/O instructions
- Three new instructions, STOP, WAIT and DAA, added to the HD6805 family instruction set

■ PROGRAM DEVELOPMENT SUPPORT TOOLS

- Cross assembler software for use with IBM PCs and compatibles
- In circuit emulator for use with IBM PCs and compatibles

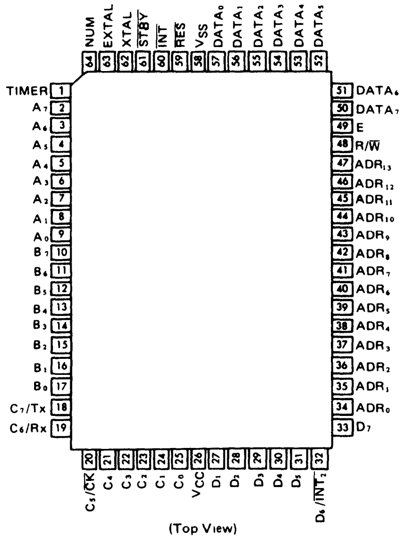
■ PIN ARRANGEMENT

- HD6305Y2P, HD63A05Y2P, HD63B05Y2P



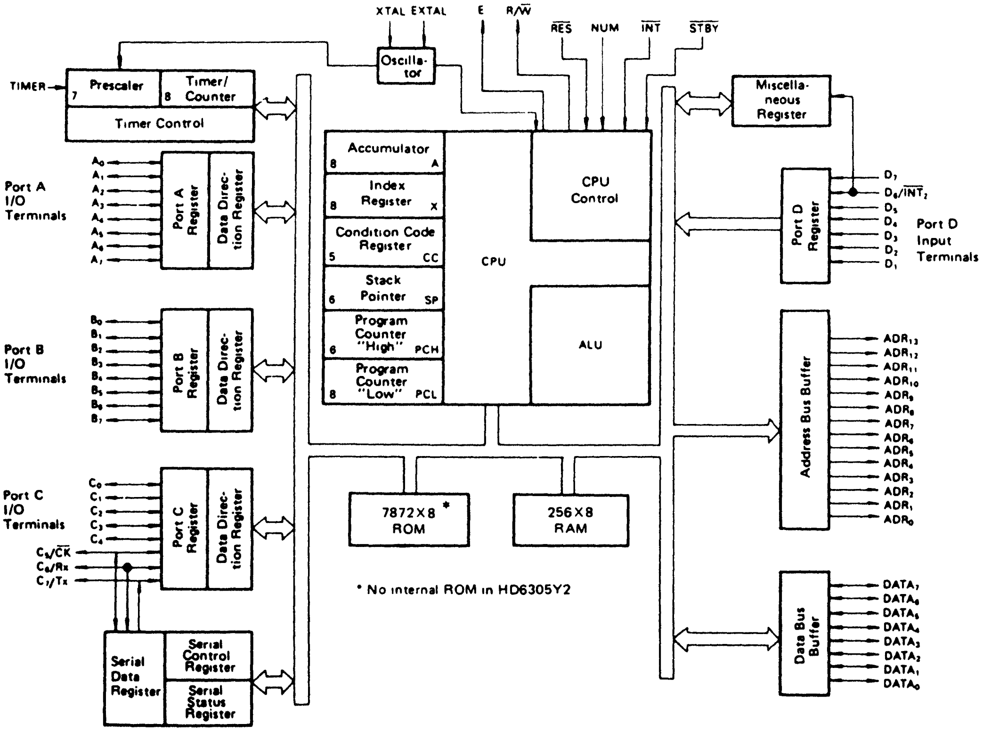
(Top View)

- HD6305Y2F, HD63A05Y2F, HD63B05Y2F



(Top View)

■ BLOCK DIAGRAM



* No internal ROM in HD6305Y2

HITACHI

■ ABSOLUTE MAXIMUM RATINGS

Item	Symbol	Value	Unit
Supply Voltage	V_{CC}	-0.3 ~ +7.0	V
Input Voltage	V_{in}	-0.3 ~ V_{CC} + 0.3	V
Operating Temperature	T_{opr}	0 ~ +70	°C
Storage Temperature	T_{stg}	-55 ~ +150	°C

[NOTE] These products have a protection circuit in their input terminals against high electrostatic voltage or high electric fields. Notwithstanding, be careful not to apply any voltage higher than the absolute maximum rating to these high input impedance circuits. To assure normal operation, we recommended V_{in}, V_{out}: $V_{SS} \leq (V_{in}$ or $V_{out}) \leq V_{CC}$.

■ ELECTRICAL CHARACTERISTICS

● DC CHARACTERISTICS (V_{CC} = 5.0V±10%, V_{SS} = GND, Ta = 0 ~ +70°C, unless otherwise noted.)

Item		Symbol	Test Condition	min	typ	max	Unit
Input "High" Voltage	$\overline{RES}$, $\overline{STBY}$	V_{IH}		V_{CC}-0.5	–	V_{CC}+0.3	V
	EXTAL			V_{CC}x0.7	–	V_{CC}+0.3	
	Other Inputs			2.0	–	V_{CC}+0.3	
Input "Low" Voltage	All Inputs	V_{IL}		-0.3	–	0.8	V
Output "High" Voltage	All Outputs	V_{OH}	I_{OH} = -200μA	2.4	–	–	V
			I_{OH} = -10μA	V_{CC}-0.7	–	–	
Output "Low" Voltage	All Outputs	V_{OL}	I_{OL} = 1.6mA	–	–	0.55	V
Input Leakage Current	TIMER, $\overline{INT}$, D_1 ~ D_7, $\overline{STBY}$	$\lvert I_{IL} \rvert$	Vin = 0.5 ~ V_{CC}-0.5	–	–	1.0	μA
Three-state Current	A_0 ~ A_7, B_0 ~ B_7, C_0 ~ C_7, ADR_0 ~ ADR_{13}*, $DATA_0$~ $DATA_7$,E*, R/$\overline{W}$*	$\lvert I_{TSI} \rvert$		–	–	1.0	μA
Current Dissipation**	Operating	I_{CC}	f = 1MHz***	–	5	10	mA
	Wait			–	2	5	mA
	Stop			–	2	10	μA
	Standby			–	2	10	μA
Input Capacitance	All Terminals	Cin	f = 1MHz, Vin = 0V	–	–	12	pF

* Only at standby

** All output and $\overline{RES}$ terminal are open, and ponetrate current of input are not included. (V_{IH} min = V_{CC} - 1.0V, V_{IL} max = 0.8V)

*** The value at f = *x*MHz is given by using
I_{CC} (f = *x*MHz) = I_{CC} (f = 1MHz) x *x*

● AC CHARACTERISTICS (V_{CC} = 5.0V±10%, V_{SS} = GND, Ta = 0 ~ +70°C, unless otherwise noted.)

Item	Symbol	Test Condition	HD6305Y2			HD63A05Y2			HD63B05Y2			Unit
			min	typ	max	min	typ	max	min	typ	max	
Cycle Time	t_{cyc}	Fig. 1	1	–	10	0.666	–	10	0.5	–	10	μs
Enable Rise Time	t_{Er}		–	–	20	–	–	20	–	–	20	ns
Enable Fall Time	t_{Ef}		–	–	20	–	–	20	–	–	20	ns
Enable Pulse Width("High" Level)	PW_{EH}		450	–	–	300	–	–	220	–	–	ns
Enable Pulse Width("Low" Level)	PW_{EL}		450	–	–	300	–	–	220	–	–	ns
Address Delay Time	t_{AD}		–	–	250	–	–	190	–	–	180	ns
Address Hold Time	t_{AH}		40	–	–	30	–	–	20	–	–	ns
Data Delay Time	t_{DW}		–	–	200	–	–	160	–	–	120	ns
Data Hold Time (Write)	t_{HW}		40	–	–	30	–	–	20	–	–	ns
Data Set-up Time (Read)	t_{DSR}		80	–	–	60	–	–	50	–	–	ns
Data Hold Time (Read)	t_{HR}		0	–	–	0	–	–	0	–	–	ns

- **PORT TIMING (V_{CC} = 5.0V±10%, V_{SS} = GND, Ta = 0 ~ +70°C, unless otherwise noted.)**

Item	Symbol	Test Condition	HD6305Y2			HD63A05Y2			HD63B05Y2			Unit
			min	typ	max	min	typ	max	min	typ	max	
Port Data Set-up Time (Port A, B, C, D)	t_{PDS}	Fig. 2	200	–	–	200	–	–	200	–	–	ns
Port Data Hold Time (Port A, B, C, D)	t_{PDH}		200	–	–	200	–	–	200	–	–	ns
Port Data Delay Time (Port A, B, C)	t_{PDW}	Fig. 3	–	–	300	–	–	300	–	–	300	ns

- **CONTROL SIGNAL TIMING (V_{CC} = 5.0V±10%, V_{SS} = GND, Ta = 0 ~ +70°C, unless otherwise noted.)**

Item	Symbol	Test Condition	HD6305Y2			HD63A05Y2			HD63B05Y2			Unit
			min	typ	max	min	typ	max	min	typ	max	
$\overline{\text{INT}}$ Pulse Width	t_{IWL}		t_{cyc} +250	–	–	t_{cyc} +200	–	–	t_{cyc} +200	–	–	ns
$\overline{\text{INT}_2}$ Pulse Width	t_{IWL2}		t_{cyc} +250	–	–	t_{cyc} +200	–	–	t_{cyc} +200	–	–	ns
$\overline{\text{RES}}$ Pulse Width	t_{RWL}		5	–	–	5	–	–	5	–	–	t_{cyc}
Control Set-up Time	t_{CS}	Fig 5	250	–	–	250	–	–	250	–	–	ns
Timer Pulse Width	t_{TWL}		t_{cyc} +250	–	–	t_{cyc} +200	–	–	t_{cyc} +200	–	–	ns
Oscillation Start Time (Crystal)	$t_{C\,SC}$	Fig 5, Fig 20*	–	–	20	–	–	20	–	–	20	ms
Reset Delay Time	t_{RHL}	Fig. 19	80	–	–	80	–	–	80	–	–	ms

* C_L = 22pF ±20%, R_s = 60Ω max.

- **SCI TIMING (V_{CC} = 5.0V±10%, V_{SS}= GND, Ta = 0 ~ +70°C, unless otherwise noted.)**

Item	Symbol	Test Condition	HD6305Y2			HD63A05Y2			HD63B05Y2			Unit
			min	typ	max	min	typ	max	min	typ	max	
Clock Cycle	t_{Scyc}	Fig. 6, Fig. 7	1	–	32768	0.67	–	21845	0.5	–	16384	μs
Data Output Delay Time	t_{TXD}		–	–	250	–	–	250	–	–	250	ns
Data Set-up Time	t_{SRX}		200	–	–	200	–	–	200	–	–	ns
Data Hold Time	t_{HRX}		100	–	–	100	–	–	100	–	–	ns

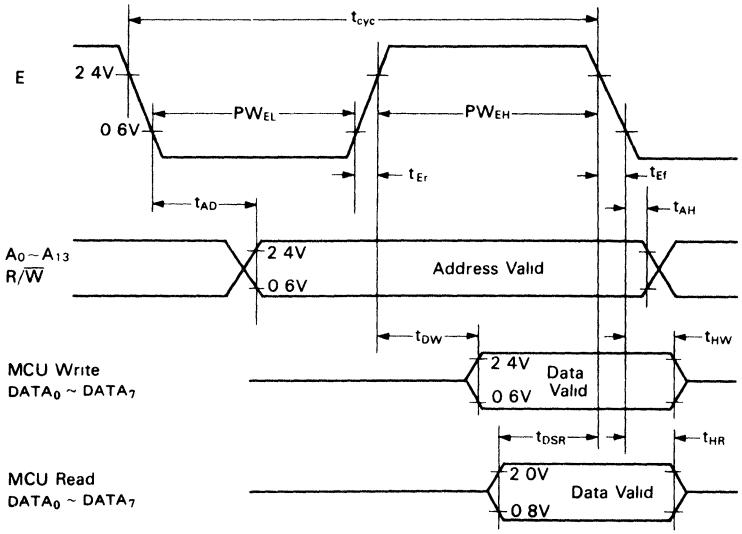


Figure 1 Bus Timing

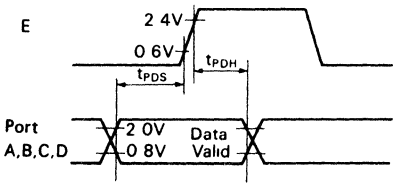


Figure 2 Port Data Set-up and Hold Times (MCU Read)

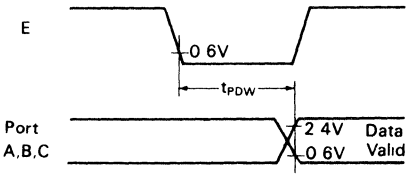


Figure 3 Port Data Delay Time (MCU Write)

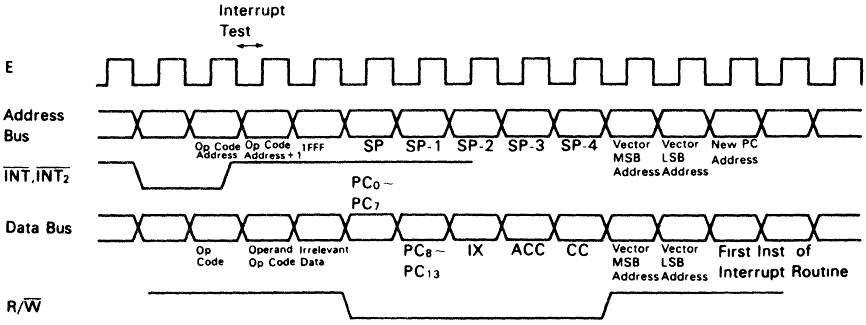


Figure 4 Interrupt Sequence

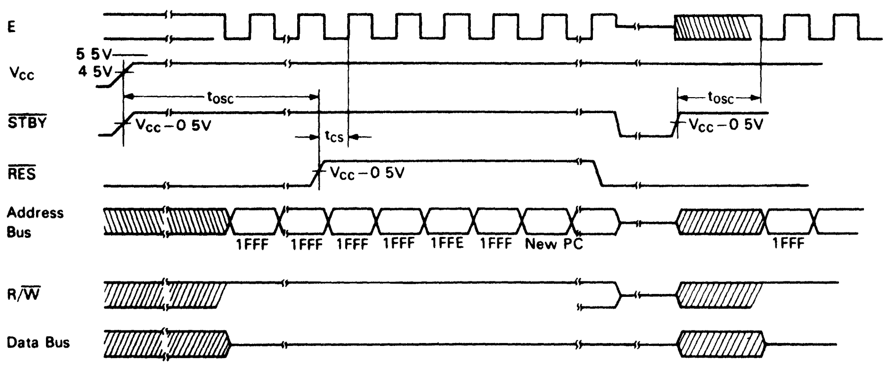


Figure5 Reset Timing

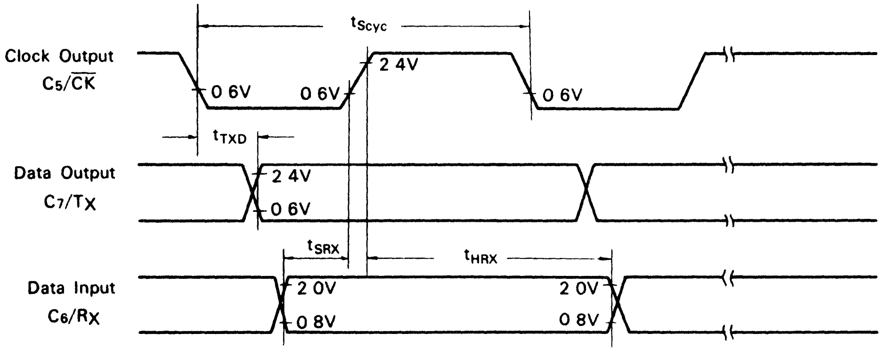


Figure6 SCI Timing (Internal Clock)

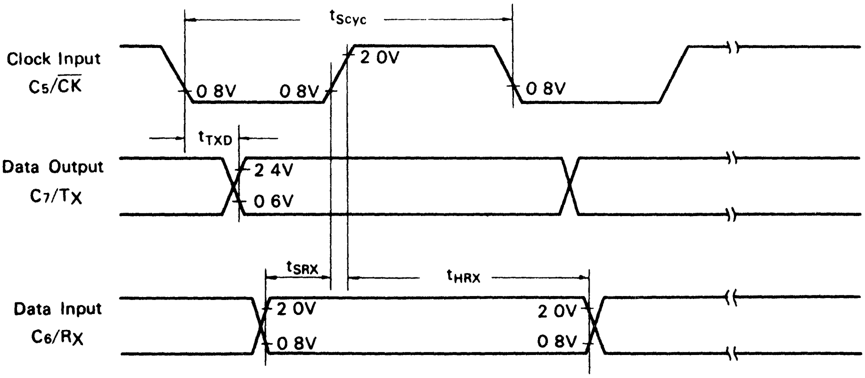
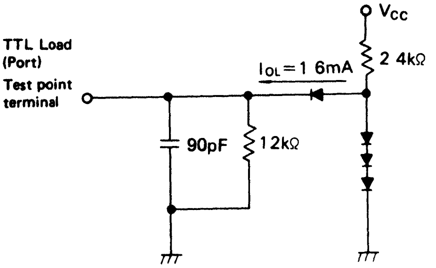


Figure7 SCI Timing(External Clock)



[NOTES] 1. The load capacitance includes stray capacitance caused by the probe, etc.
2. All diodes are 1S2074 Ⓗ

Figure 8 Test Load

■ DESCRIPTION OF TERMINAL FUNCTIONS

The input and output signals of the MCU are described here.

• V_{CC}, V_{SS}

Voltage is applied to the MCU through these two terminals. V_{CC} is 5.0V ± 10%, while V_{SS} is grounded.

• $\overline{INT}$, $\overline{INT_2}$

External interrupt request inputs to the MCU. For details, refer to "INTERRUPT". The $\overline{INT_2}$ terminal is also used as the port D_6 terminal.

• XTAL, EXTAL

These terminals provide input to the on-chip clock circuit. A crystal oscillator (AT cut, 2.0 to 8.0 MHz) or ceramic filter is connected to the terminal. Refer to "INTERNAL OSCILLATOR" for using these input terminals.

• TIMER

This is an input terminal for event counter. Refer to "TIMER" for details.

• $\overline{RES}$

Used to reset the MCU. Refer to "RESET" for details.

• NUM

This terminal is not for user application. In case of the HD6305Y1, this terminal should be connected to V_{CC} through 10kΩ resistance. In case of the HD6305Y2, this terminal should be connected to V_{SS}.

• Enable (E)

This output terminal supplies E clock. Output is a single-phase, TTL compatible and 1/4 crystal oscillation frequency or 1/4 external clock frequency. It can drive one TTL load and a 90pF condenser.

• Read/Write (R/$\overline{W}$)

This TTL compatible output signal indicates to peripheral and memory devices whether MCU is in Read ("High"), or in Write ("Low"). The normal standby state is Read ("High"). Its output can drive one TTL load and a 90pF condenser.

• Data Bus ($DATA_0$ ~ $DATA_7$)

This TTL compatible three-state buffer can drive one TTL load and 90pF.

• Address Bus (ADR_0 ~ ADR_{13})

Each terminal is TTL compatible and can drive one TTL load and 90pF.

• Input/Output Terminals (A_0 ~ A_7, B_0 ~ B_7, C_0 ~ C_7)

These 24 terminals consist of three 8-bit I/O ports (A, B, C). Each of them can be used as an input or output terminal on a bit through program control of the data direction register. For details, refer to "I/O PORTS."

• Input Terminals (D_1 ~ D_7)

These seven input-only terminals are TTL or CMOS compatible. Of the port D's, D_6 is also used as $\overline{INT_2}$. If D_6 is used as a port, the $\overline{INT_2}$ interrupt mask bit of the miscellaneous register must be set to "1" to prevent an $\overline{INT_2}$ interrupt from being accidentally accepted.

• $\overline{STBY}$

This terminal is used to place the MCU into the standby mode. With $\overline{STBY}$ at "Low" level, the oscillation stops and the internal condition is reset. For details, refer to "Standby Mode."

The terminals described in the following are I/O pins for serial communication interface (SCI). They are also used as ports C_5, C_6 and C_7. For details, refei to "SERIAL COMMUNICATION INTERFACE."

• $\overline{CK}$ (C_5)

Used to input or output clocks for serial operation.

• Rx (C_6)

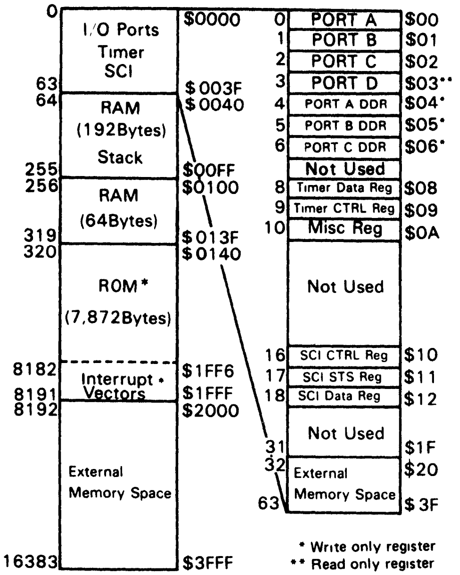
Used to receive serial data.

• Tx (C_7)

Used to transmit serial data.

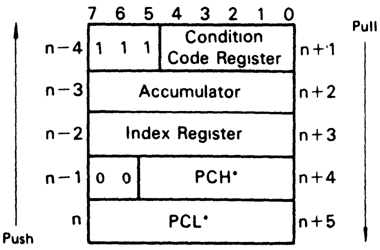
■ MEMORY MAP

The memory map of the MCU is shown in Fig. 9. \$0140 ~ \$1FFF of the HD6305Y2 are external addresses. However, care should be taken to assign vector addresses to \$1FF6 ~ \$1FFF. During interrupt processing, the contents of the CPU registers are saved into the stack in the sequence shown in Fig. 10. This saving begins with the lower byte (PCL) of the program counter. Then the value of the stack pointer is decremented and the higher byte (PCH) of the program counter, index register (X), accumulator (A) and condition code register (CC) are stacked in that order. In a subroutine call, only the contents of the program counter (PCH and PCL) are stacked.



* ROM area ($0140 ~ $1FFF) in the HD6305Y2 is changed into External Memory Space

Figure 9 Memory Map of MCU



* In a subroutine call, only PCL and PCH are stacked

Figure 10 Sequence of Interrupt Stacking

■REGISTERS

There are five registers which the programmer can operate.

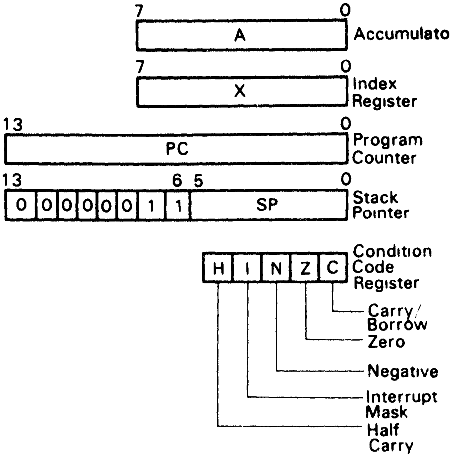


Figure 11 Programming Model

• Accumulator (A)

This accumulator is an ordinary 8-bit register which holds operands or the result of arithmetic operation or data processing.

• Index Register (X)

The index register is an 8-bit register, and is used for index addressing mode. Each of the addresses contained in the register consists of 8 bits which, combined with an offset value, provides an effective address.

In the case of a read/modify/write instruction, the index register can be used like an accumulator to hold operation data or the result of operation.

If not used in the index addressing mode, the register can be used to store data temporarily.

• Program Counter (PC)

The program counter is a 14-bit register that contains the address of the next instruction to be executed.

• Stack Pointer (SP)

The stack pointer is a 14-bit register that indicates the address of the next stacking space. Just after reset, the stack pointer is set at address $00FF. It is decremented when data is pushed, and incremented when pulled. The upper 8 bits of the stack pointer are fixed to 00000011. During the MCU being reset or during a reset stack pointer (RSP) instruction, the pointer is set to address $00FF. Since a subroutine or interrupt can use space up to address $00C1 for stacking, the subroutine can be used up to 31 levels and the interrupt up to 12 levels.

• Condition Code Register (CC)

The condition code register is a 5-bit register, each bit indicating the result of the instruction just executed. The bits can be individually tested by conditional branch instructions. The CC bits are as follows:

Half Carry (H): Used to indicate that a carry occurred between bits 3 and 4 during an arithmetic operation (ADD, ADC).

Interrupt (I): Setting this bit causes all interrupts, except a software interrupt, to be masked. If an interrupt occurs with the bit I set, it is latched. It will be processed the instant the interrupt mask bit is reset. (More specifically, it will enter the interrupt processing routine after the instruction following the CLI has been executed.)

Negative (N): Used to indicate that the result of the most recent arithmetic operation, logical operation or data processing is negative (bit 7 is logic "1").

Zero (Z) Used to indicate that the result of the most recent arithmetic operation, logical operation or data processing is zero.

Carry/Borrow (C): Represents a carry or borrow that occurred in the most recent arithmetic operation. This bit is also affected by the Bit Test and Branch instruction and a Rotate instruction.

■ INTERRUPT

There are six different types of interrupt: external interrupts ($\overline{\text{INT}}$, $\overline{\text{INT}}_2$), internal timer interrupts (TIMER, TIMER$_2$), serial interrupt (SCI) and interrupt by an instruction (SWI).

Of these six interrupts, the $\overline{\text{INT}}_2$ and TIMER or the SCI and TIMER$_2$ generate the same vector address, respectively.

When an interrupt occurs, the program in progress stops and the then CPU status is saved onto the stack. And then, the interrupt mask bit (I) of the condition code register is set and the start address of the interrupt processing routine is obtained from a particular interrupt vector address. Then the interrupt routine starts from the start address. System can exit from the interrupt routine by an RTI instruction. When this instruction is executed, the CPU status before the interrupt (saved onto the stack) is pulled and the CPU restarts the sequence with the instruction next to the one at which the interrupt occurred. Table 1 lists the priority of interrupts and their vector addresses.

Table 1 Priority of Interrupts

Interrupt	Priority	Vector Address
$\overline{\text{RES}}$	1	\$1FFE, \$1FFF
SWI	2	\$1FFC, \$1FFD
$\overline{\text{INT}}$	3	\$1FFA, \$1FFB
TIMER/$\overline{\text{INT}}_2$	4	\$1FF8, \$1FF9
SCI/TIMER$_2$	5	\$1FF6, \$1FF7

A flowchart of the interrupt sequence is shown in Fig. 12. A block diagram of the interrupt request source is shown in Fig. 13.

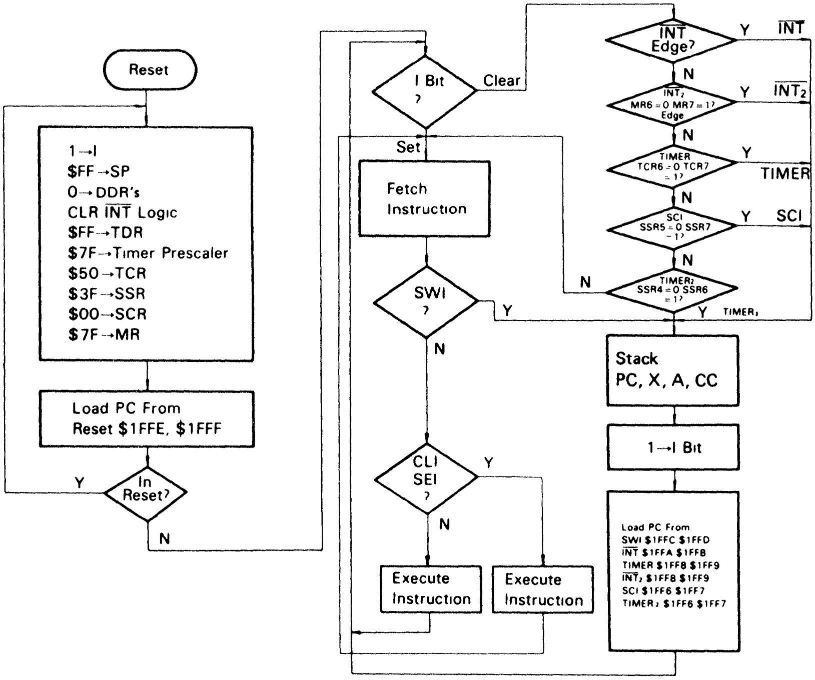


Figure 12 Interrupt Flow Chart

In the block diagram, both the external interrupts $\overline{\mathrm{INT}}$ and $\overline{\mathrm{INT}_2}$ are edge trigger inputs. At the falling edge of each input, an interrupt request is generated and latched. The $\overline{\mathrm{INT}}$ interrupt request is automatically cleared if jumping is made to the $\overline{\mathrm{INT}}$ processing routine. Meanwhile, the $\overline{\mathrm{INT}_2}$ request is cleared if "0" is written in bit 7 of the miscellaneous register.

For the external interrupts ($\overline{\mathrm{INT}}$, $\overline{\mathrm{INT}_2}$), internal timer interrupts (TIMER, TIMER$_2$) and serial interrupt (SCI), each interrupt request is held, but not processed, if the I bit of the condition code register is set. Immediately after the I bit is cleared, the corresponding interrupt processing starts according to the priority.

The $\overline{\mathrm{INT}_2}$ interrupt can be masked by setting bit 6 of the miscellaneous register; the TIMER interrupt by setting bit 6 of the timer control register; the SCI interrupt by setting bit 5 of the serial status register; and the TIMER$_2$ interrupt by setting bit 4 of the serial status register.

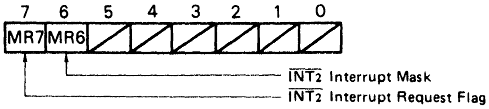
The status of the $\overline{\mathrm{INT}}$ terminal can be tested by a BIL or BIH instruction. The $\overline{\mathrm{INT}}$ falling edge detector circuit and its latching circuit are independent of testing by these instructions. This is also true with the status of the $\overline{\mathrm{INT}_2}$ terminal.

- **Miscellaneous Register (MR; $000A)**

The interrupt vector address for the external interrupt $\overline{\mathrm{INT}_2}$ is the same as that for the TIMER interrupt, as shown in Table 1. For this reason, a special register called the miscellaneous register (MR, $000A) is available to control the $\overline{\mathrm{INT}_2}$ interrupts.

Bit 7 of this register is the $\overline{\mathrm{INT}_2}$ interrupt request flag. When the falling edge is detected at the $\overline{\mathrm{INT}_2}$ terminal, "1" is set in bit 7. Then the software in the interrupt routine (vector addresses: $1FF8, $1FF9) checks bit 7 to see if it is $\overline{\mathrm{INT}_2}$ interrupt. Bit 7 can be reset by software.

Miscellaneous Register (MR;$000A)


7 6 5 4 3 2 1 0
MR7
MR6
$\overline{\mathrm{INT}_2}$ Interrupt Mask
$\overline{\mathrm{INT}_2}$ Interrupt Request Flag

Bit 6 is the $\overline{\mathrm{INT}_2}$ interrupt mask bit. If this bit is set to "1", then the $\overline{\mathrm{INT}_2}$ interrupt is disabled. Both read and write are possible with bit 7 but "1" cannot be written in this bit by software. This means that an interrupt request by software is impossible.

When reset, bit 7 is cleared to "0" and bit 6 is set to "1".

- **TIMER**

Figure 14 shows a MCU timer block diagram. The timer data register is loaded by software and, upon receipt of a clock input, begins to count down. When the timer data

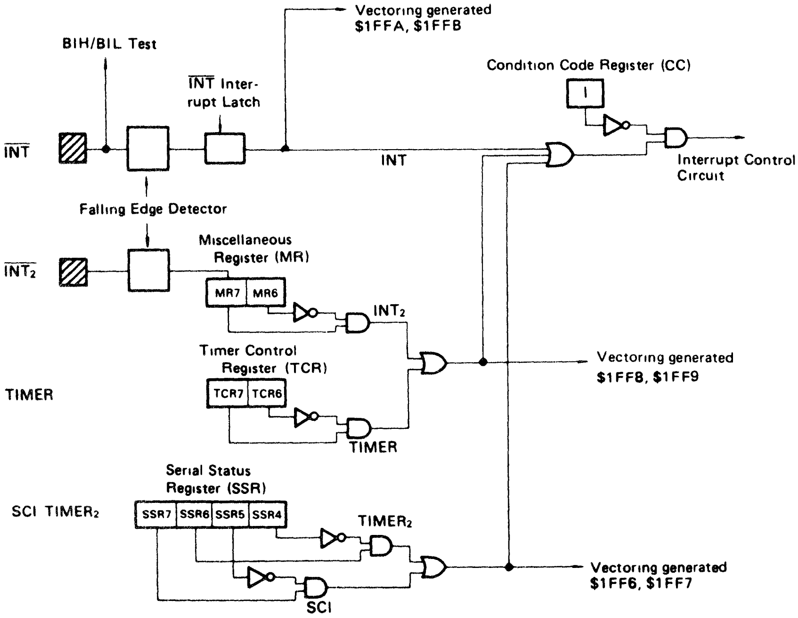

Vectoring generated
$1FFA, $1FFB
BIH/BIL Test
$\overline{\mathrm{INT}}$ Interrupt Latch
Condition Code Register (CC)
I
$\overline{\mathrm{INT}}$
INT
Interrupt Control Circuit
Falling Edge Detector
Miscellaneous Register (MR)
$\overline{\mathrm{INT}_2}$
MR7
MR6
INT$_2$
Timer Control Register (TCR)
Vectoring generated
$1FF8, $1FF9
TIMER
TCR7
TCR6
TIMER
Serial Status Register (SSR)
SCI TIMER$_2$
SSR7
SSR6
SSR5
SSR4
TIMER$_2$
Vectoring generated
$1FF6, $1FF7
SCI

Figure 13 Interrupt Request Generation Circuitry

register (TDR) becomes "0", the timer interrupt request bit (bit 7) in the timer control register is set. In response to the interrupt request, the CPU saves its status into the stack and fetches timer interrupt routine address from addresses $1FF8 and $1FF9 and execute the interrupt routine. The timer interrupt can be masked by setting the timer interrupt mask bit (bit 6) in the timer control register. The mask bit (I) in the condition code register can also mask the timer interrupt.

The source clock to the timer can be either an external signal from the timer input terminal or the internal E signal (the oscillator clock divided by 4). If the E signal is used as the source, the clock input can be gated by the input to the timer input terminal.

Once the timer count has reached "0", it starts counting down with "$FF". The count can be monitored whenever desired by reading the timer data register. This permits the program to know the length of time having passed after the occurrence of a timer interrupt, without disturbing the contents of the counter.

When the MCU is reset, both the prescaler and counter are initialized to logic "1". The timer interrupt request bit (bit 7) then is cleared and the timer interrupt mask bit (bit 6) is set.

To clear the timer interrupt request bit (bit 7), it is necessary to write "0" in that bit.

TCR7	Timer interrupt request
0	Absent
1	Present

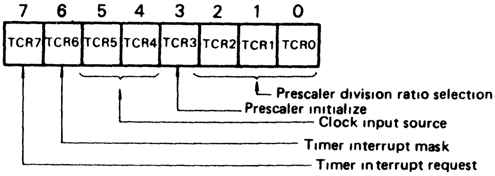
TCR6	Timer interrupt mask
0	Enabled
1	Disabled

• Timer Control Register (TCR; $0009)

Selection of a clock source, selection of a prescaler frequency division ratio, and a timer interrupt can be controlled by the timer control register (TCR; $0009).

For the selection of a clock source, any one of the four modes (see Table 2) can be selected by bits 5 and 4 of the timer control register (TCR).

Timer Control Register (TCR; $0009)



After reset, the TCR is initialized to "E under timer terminal control" (bit 5 = 0, bit 4 = 1). If the timer terminal is "1", the counter starts counting down with "$FF" immediately after reset.

When "1" is written in bit 3, the prescaler is initialized. This bit always shows "0" when read.

Table 2 Clock Source Selection

TCR		Clock input source
Bit 5	Bit 4	
0	0	Internal clock E
0	1	E under timer terminal control
1	0	No clock input (counting stopped)
1	1	Event input from timer terminal

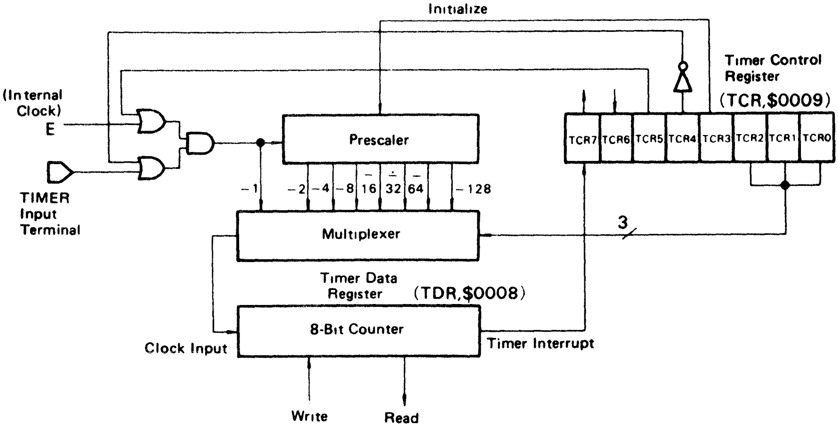


Figure 14 Timer Block Diagram

A prescaler division ratio is selected by the combination of three bits (bits 0, 1 and 2) of the timer control register (see Table 3). There are eight different division ratios: ÷1, ÷2, ÷4, ÷8, ÷16, ÷32, ÷64 and ÷128. After reset, the TCR is set to the ÷1 mode.

Table 3 Prescaler Division Ratio Selection

TCR			Prescaler division ratio
Bit 2	Bit 1	Bit 0	
0	0	0	÷1
0	0	1	÷2
0	1	0	÷4
0	1	1	÷8
1	0	0	÷16
1	0	1	÷32
1	1	0	÷64
1	1	1	÷128

A timer interrupt is enabled when the timer interrupt mask bit is "0", and disabled when the bit is "1". When a timer interrupt occurs, "1" is set in the timer interrupt request bit. This bit can be cleared by writing "0" in that bit.

■SERIAL COMMUNICATION INTERFACE (SCI)

This interface is used for serial transmission or reception of 8-bit data. Sixteen transfer rates are available in the range from 1 μs to approx. 32 ms (for oscillation at 4 MHz).

The SCI consists of three registers, one octal counter and one prescaler. (See Fig. 15.) SCI communicates with the CPU via the data bus, and with the outside world through bits 5, 6 and 7 of port C. Described below are the operations of each register and data transfer.

●SCI Control Register (SCR; $0010)

7	6	5	4	3	2	1	0
SCR7	SCR6	SCR5	SCR4	SCR3	SCR2	SCR1	SCR0

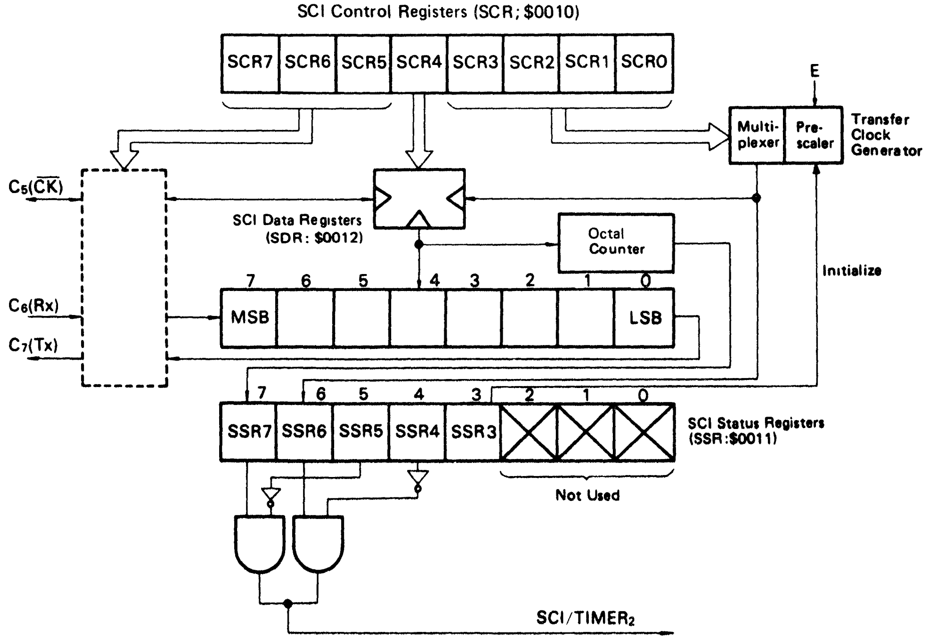


Figure 15 SCI Block Diagram

SCR7	C_7 terminal
0	Used as I/O terminal (by DDR).
1	Serial data output (DDR output)

SCR6	C_6 terminal
0	Used as I/O terminal (by DDR).
1	Serial data input (DDR input)

SCR5	SCR4	Clock source	C_5 terminal
0	0	–	Used as I/O terminal (by DDR).
0	1	–	
1	0	Internal	Clock output (DDR output)
1	1	External	Clock input (DDR input)

Bit 7 (SCR7)

When this bit is set, the DDR corresponding to the C_7 becomes "1" and this terminal serves for output of SCI data. After reset, the bit is cleared to "0".

Bit 6 (SCR6)

When this bit is set, the DDR corresponding to the C_6 becomes "0" and this terminal serves for input of SCI data. After reset, the bit is cleared to "0".

Bits 5 and 4 (SCR5, SCR4)

These bits are used to select a clock source. After reset, the bits are cleared to "0".

Bits 3 ~ 0 (SCR3 ~ SCR0)

These bits are used to select a transfer clock rate. After reset, the bits are cleared to "0".

SCR3	SCR2	SCR1	SCR0	Transfer clock rate	
				4.00 MHz	4.194 MHz
0	0	0	0	1 μs	0.95 μs
0	0	0	1	2 μs	1.91 μs
0	0	1	0	4 μs	3.82 μs
0	0	1	1	8 μs	7.64 μs
≀	≀	≀	≀	≀	≀
1	1	1	1	32768 μs	1/32 s

•SCI Data Register (SDR; $0012)

A serial-parallel conversion register that is used for transfer of data.

•SCI Status Register (SSR; $0011)

7	6	5	4	3	2	1	0
SSR7	SSR6	SSR5	SSR4	SSR3	☒	☒	☒

Bit 7 (SSR7)

Bit 7 is the SCI interrupt request bit which is set upon completion of transmitting or receiving 8-bit data. It is cleared when reset or data is written to or read from the SCI data register with the SCR5="1". The bit can also be cleared by writing "0" in it.

Bit 6 (SSR6)

Bit 6 is the TIMER_2 interrupt request bit. TIMER_2 is used commonly with the serial clock generator, and SSR6 is set each time the internal transfer clock falls. When reset, the bit is cleared. It also be cleared by writing "0" in it. (For details, see TIMER_2.)

Bit 5 (SSR5)

Bit 5 is the SCI interrupt mask bit which can be set or cleared by software. When it is "1", the SCI interrupt (SSR7) is masked. When reset, it is set to "1"

Bit 4 (SSR4)

Bit 4 is the TIMER_2 interrupt mask bit which can be set or cleared by software. When the bit is "1", the TIMER_2 interrupt (SSR6) is masked. When reset, it is set to "1".

Bit 3 (SSR3)

When "1" is written in this bit, the prescaler of the transfer clock generator is initialized. When read, the bit always is "0".

Bits 2 ~ 0

Not used.

SSR7	SCI interrupt request
0	Absent
1	Present

SSR6	TIMER_2 interrupt request
0	Absent
1	Present

SSR5	SCI interrupt mask
0	Enabled
1	Disabled

SSR4	TIMER_2 interrupt mask
0	Enabled
1	Disabled

• Data Transmission

By writing the desired control bits into the SCI control registers, a transfer rate and a source of transfer clock are determined and bits 7 and 5 of port C are set at the serial data output terminal and the serial clock terminal, respectively. The transmit data should be stored from the accumulator or index register into the SCI data register. The data written in the SCI data register is output from the C_7/Tx terminal, starting with the LSB, synchronously with the falling edge of the serial clock. (See Fig. 16.) When 8 bit of data have been transmitted, the interrupt request bit is set in bit 7 of the SCI status register with the rising edge of the last serial clock. This request can be masked by setting bit 5 of the SCI status register. Once the data has been sent, the 8th bit data (MSB) stays at the C_7/Tx terminal. If an external clock source has been selected, the transfer rate determined by bits 0 ~ 3 of the SCI control register is ignored, and the $C_5/\overline{CK}$ terminal is set as input. If the internal clock has been selected, the $C_5/\overline{CK}$ terminal is set as output and clocks are output at the transfer rate selected by bits 0 ~ 3 of the SCI control register.

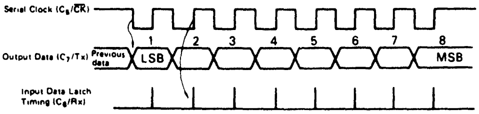

Serial Clock ($C_6/\overline{CK}$)
Output Data (C_7/Tx)
Previous data
1
2
3
4
5
6
7
8
LSB
MSB
Input Data Latch Timing (C_6/Rx)

Figure 16 SCI Timing Chart

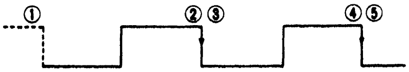
• **Data Reception**

By writing the desired control bits into the SCI control register, a transfer rate and a source of transfer clock are determined and bit 6 and 5 of port C are set at the serial data input terminal and the serial clock terminal, respectively. Then dummy-writing or -reading the SCI data register, the system is ready for receiving data. (This procedure is not needed after reading subsequent received data. It must be taken after reset and after not reading subsequent received data.)

The data from the C_6/Rx terminal is input to the SCI data register synchronously with the rising edge of the serial clock (see Fig. 16). When 8 bits of data have been received, the interrupt request bit is set in bit 7 of the SCI status register. This request can be masked by setting bit 5 of the SCI status register. If an external clock source have been selected, the transfer rate determined by bits 0 ~ 3 of the SCI control register is ignored and the data is received synchronously with the clock from the $C_5/\overline{CK}$ terminal. If the internal clock has been selected, the $C_5/\overline{CK}$ terminal is set as output and clocks are output at the transfer rate selected by bits 0 ~ 3 of the SCI control register.

• **TIMER$_2$**

The SCI transfer clock generator can be used as a timer. The clock selected by bits 3 ~ 0 of the SCI control register (4 μs ~ approx. 32 ms (for oscillation at 4 MHz)) is input to bit 6 of the SCI status register and the TIMER$_2$ interrupt request bit is set at each falling edge of the clock. Since interrupt requests occur periodically, TIMER$_2$ can be used as a reload counter or clock.



① : Transfer clock generator is reset and mask bit (bit 4 of SCI status register) is cleared.

②, ④ : TIMER$_2$ interrupt request

③, ⑤ : TIMER$_2$ interrupt request bit cleared

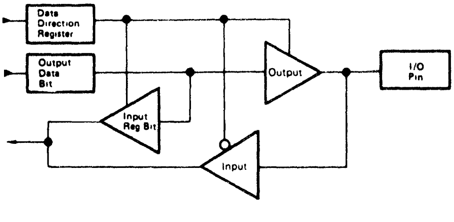
TIMER$_2$ is commonly used with the SCI transfer clock generator. If wanting to use TIMER$_2$ independently of the SCI, specify "External" (SCR5 = 1, SCR4 = 1) as the SCI clock source.

If "Internal" is selected as the clock source, reading or writing the SDR causes the prescaler of the transfer clock generator to be initialized.

■ **I/O PORTS**

There are 24 input/output terminals (ports A, B, C). Each I/O terminal can be selected for either input or output by the data direction register. More specifically, an I/O port will be input if "0" is written in the data direction register, and output if "1" is written in the data direction register. Port A, B or C reads latched data if it has been programmed as output, even with the output level being fluctuated by the output load. (See Fig. 17.)

When reset, the data direction register and data register go to "0" and all the input/output terminals are used as input



Bit of data direction register	Bit of output data	Status of output	Input to CPU
1	0	0	0
1	1	1	1
0	X	3-state	Pin

Figure 17 Input/Output Port Diagram

Seven input-only terminals are available (port D). Writing to an input terminal is invalid.

All input/output terminals and input terminals are TTL compatible and CMOS compatible in respect of both input and output.

If I/O ports or input ports are not used, they should be connected to V_{SS} via resistors. With none connected to these terminals, there is the possibility of power being consumed despite that they are not used.

■ **RESET**

The MCU can be reset either by external reset input ($\overline{RES}$) or power-on reset. (See Fig. 18.) On power up, the reset input must be held "Low" for at least t_{OSC} to assure that the internal oscillator is stabilized. A sufficient time of delay can be obtained by connecting a capacitance to the $\overline{RES}$ input as shown in Fig. 19.

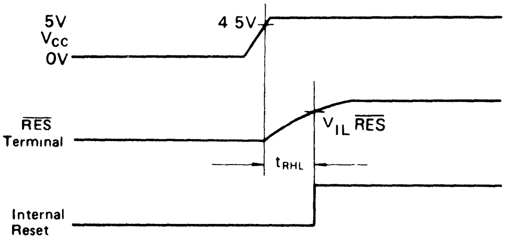


Figure 18 Power On and Reset Timing

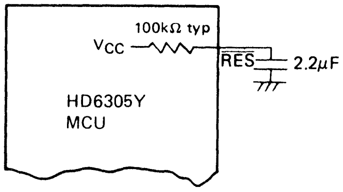
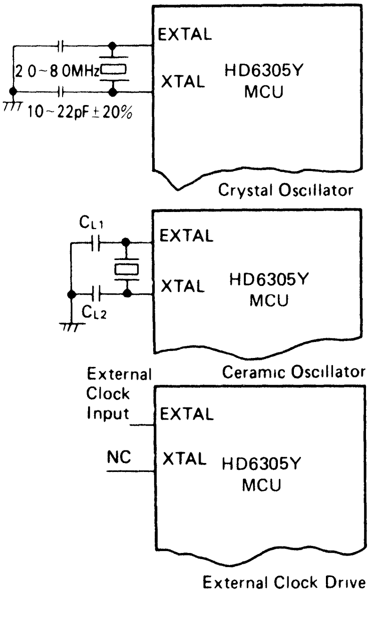


Figure 19 Input Reset Delay Circuit

■INTERNAL OSCILLATOR

The internal oscillator circuit is designed to meet the requirement for minimum external configurations. It can be driven by connecting a crystal (AT cut 2.0 ~ 8.0MHz) or ceramic oscillator between pins 5 and 6 depending on the required oscillation frequency stability.

Three different terminal connections are shown in Fig. 20. Figs. 21 and 22 illustrate the specifications and typical arrangement of the crystal, respectively.



Crystal Oscillator

Ceramic Oscillator

External Clock Drive

Figure 20 Internal Oscillator Circuit

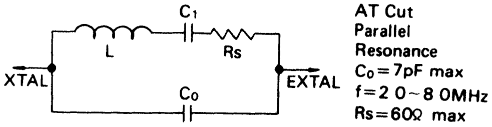
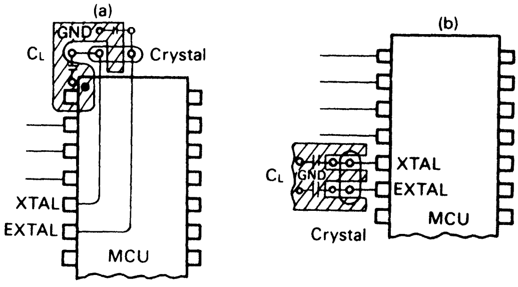


Figure 21 Parameters of Crystal



[NOTE] Use as short wirings as possible for connection of the crystal with the EXTAL and XTAL terminals Do not allow these wirings to cross others

Figure 22 Typical Crystal Arrangement

■LOW POWER DISSIPATION MODE

The HD6305Y has three low power dissipation modes: wait, stop and standby.

• Wait Mode

When WAIT instruction being executed, the MCU enters into the wait mode. In this mode, the oscillator stays active but the internal clock stops. The CPU stops but the peripheral functions – the timer and the serial communication interface – stay active. (NOTE: Once the system has entered the wait mode, the serial communication interface can no longer be retriggered.) In the wait mode, the registers, RAM and I/O terminals hold their condition just before entering into the wait mode.

The escape from this mode can be done by interrupt ($\overline{\text{INT}}$, TIMER/$\overline{\text{INT}_2}$ or SCI/TIMER2), $\overline{\text{RES}}$ or $\overline{\text{STBY}}$. The $\overline{\text{RES}}$ resets the MCU and the $\overline{\text{STBY}}$ brings it into the standby mode. (This will be mentioned later.)

When interrupt is requested to the CPU and accepted, the wait mode escapes, then the CPU is brought to the operation mode and vectors to the interrupt routine. If the interrupt is masked by the I bit of the condition code register, after releasing from the wait mode the MCU executes the instruction next to the WAIT. If an interrupt other than the $\overline{\text{INT}}$ (i.e., TIMER/$\overline{\text{INT}_2}$ or SCI/TIMER2) is masked by the timer control register, miscellaneous register or serial status register, there is no interrupt request to the CPU, so the wait mode cannot be released

Fig. 23 shows a flowchart for the wait function.

- **Stop Mode**

When STOP instruction being executed, MCU enters into the stop mode. In this mode, the oscillator stops and the CPU and peripheral functions become inactive but the RAM, registers and I/O terminals hold their condition just before entering into the stop mode.

The escape from this mode can be done by an external interrupt ($\overline{\text{INT}}$ or $\overline{\text{INT}_2}$), $\overline{\text{RES}}$ or $\overline{\text{STBY}}$. The $\overline{\text{RES}}$ resets the MCU and the $\overline{\text{STBY}}$ brings into the standby mode.

When interrupt is requested to the CPU and accepted, the stop mode escapes, then the CPU is brought to the operation mode and vectors to the interrupt routine. If the interrupt is masked by the I bit of the condition code register, after releasing from the stop mode, the MCU executes the instruction next to the STOP If the $\overline{\text{INT}_2}$ interrupt is masked by the miscellaneous register, there is no interrupt request to the MCU, so the stop mode cannot be released

Fig. 24 shows a flowchart for the stop function. Fig. 25 shows a timing chart of return to the operation mode from the stop mode

For releasing from the stop mode by an interrupt, oscillation starts upon input of the interrupt and, after the internal delay time for stabilized oscillation, the CPU becomes active. For restarting by $\overline{\text{RES}}$, oscillation starts when the $\overline{\text{RES}}$ goes "0" and the CPU restarts when the $\overline{\text{RES}}$ goes "1". The duration of $\overline{\text{RES}}$="0" must exceed 30 ms to assure stabilized oscillation

- **Standby Mode**

The MCU enters into the standby mode when the $\overline{\text{STBY}}$ terminal goes "Low" In this mode, all operations stop and the internal condition is reset but the contents of the RAM are hold. The I/O terminals turn to high-impedance state The standby mode should escape by bringing $\overline{\text{STBY}}$ "High" The CPU must be restarted by reset The timing of input signals at the $\overline{\text{RES}}$ and $\overline{\text{STBY}}$ terminals is shown in Fig 26

Table 4 lists the status of each parts of the MCU in each low power dissipation modes Transitions between each mode are shown in Fig 27

(Note)

When I bit of condition code register is "1" and interrupt ($\overline{\text{INT}}$, TIMER/$\overline{\text{INT}_2}$, SCI/TIMER$_2$) is held, MCU does not enter WAIT mode by the execution of WAIT instruction.

In that case, after the 4 dummy cycles MCU executes the next instruction

In the same way, when external interrupts ($\overline{\text{INT}}$, $\overline{\text{INT}_2}$) are held at the bit I set, MCU does not enter STOP mode by the execution of STOP instruction In that case, also, MCU executes the next instruction after the 4 dummy cycles.

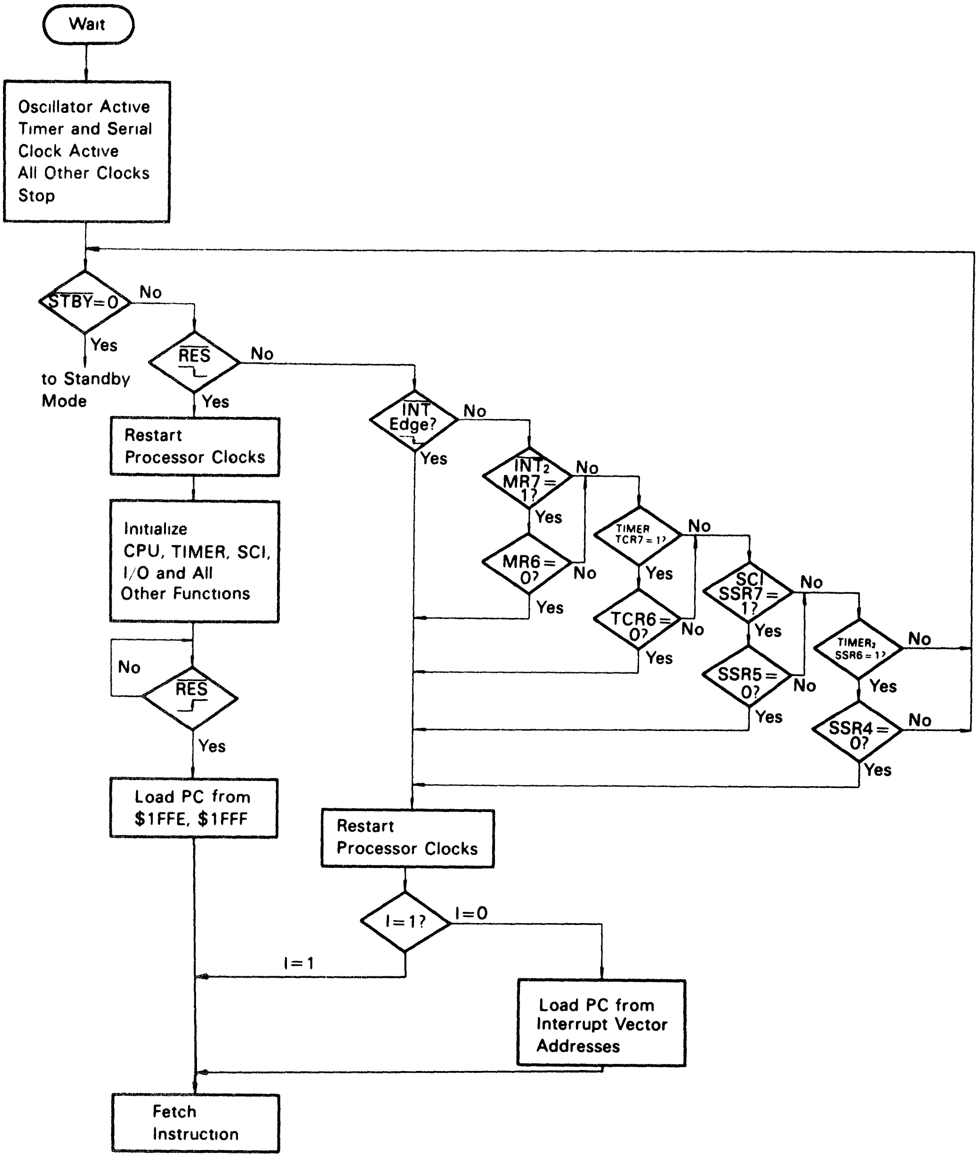


Figure 23 Wait Mode Flow Chart

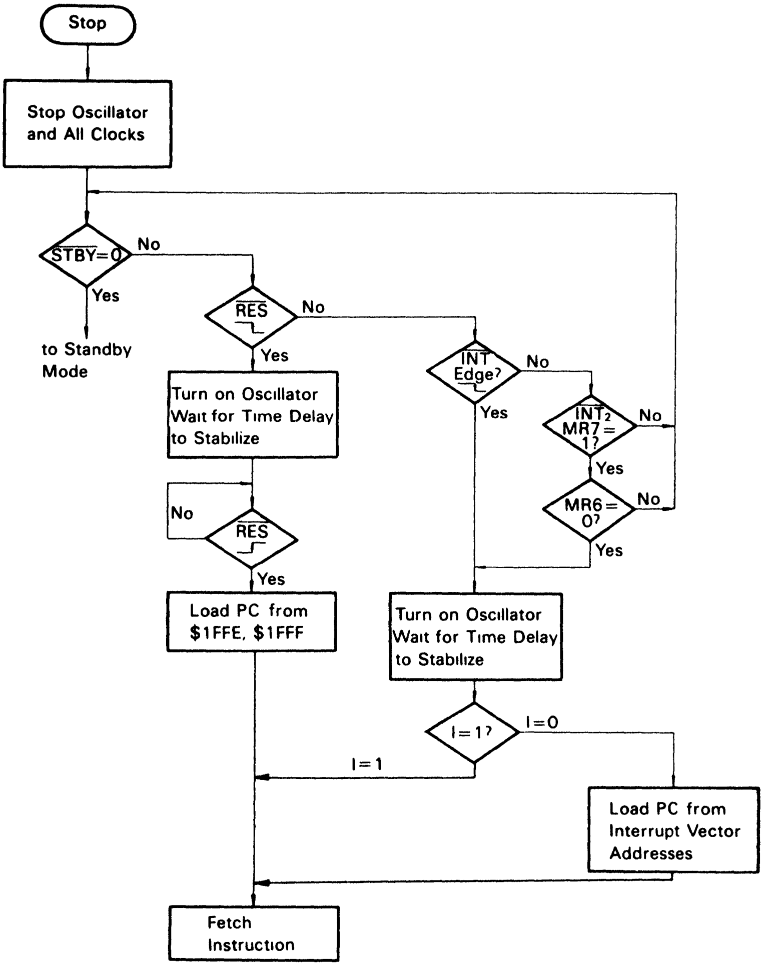
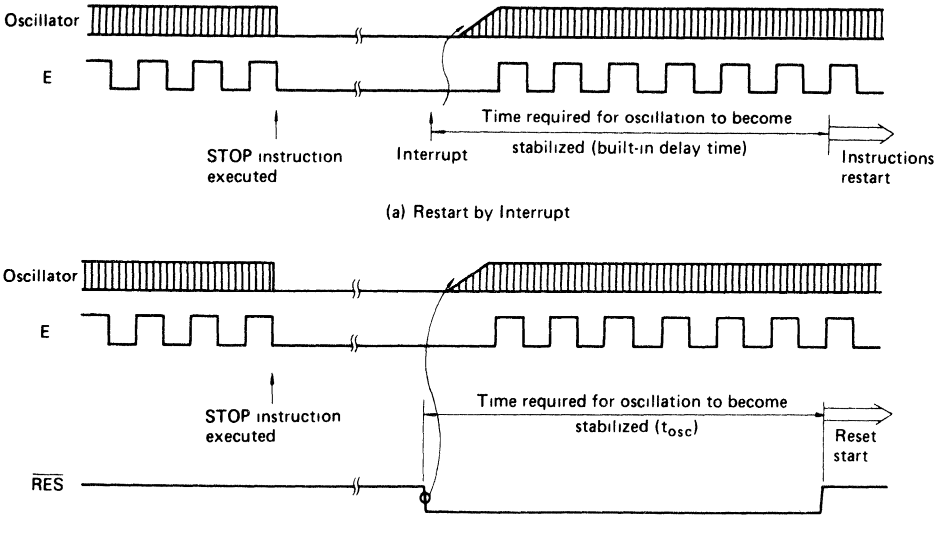

Stop
Stop Oscillator and All Clocks
STBY=0
No
Yes
to Standby Mode
RES
No
Yes
Turn on Oscillator Wait for Time Delay to Stabilize
No
RES
Yes
Load PC from $1FFE, $1FFF
INT Edge?
No
Yes
INT2 MR7=1?
No
Yes
MR6=0?
No
Yes
Turn on Oscillator Wait for Time Delay to Stabilize
I=1?
I=0
I=1
Load PC from Interrupt Vector Addresses
Fetch Instruction

Figure 24 Stop Mode Flow Chart


Oscillator
E
STOP instruction executed
Interrupt
Time required for oscillation to become stabilized (built-in delay time)
Instructions restart
Oscillator
E
STOP instruction executed
Time required for oscillation to become stabilized (t_{OSC})
Reset start
$\overline{\text{RES}}$

(a) Restart by Interrupt

(b) Restart by Reset

Figure 25 Timing Chart of Releasing from Stop Mode

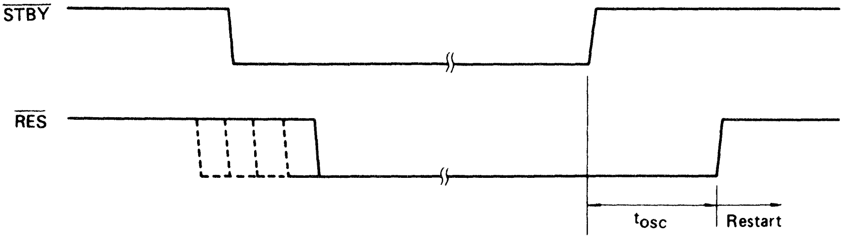


Figure 26 Timing Chart of Releasing from Standby Mode

Table 4 Status of Each Part of MCU in Low Power Dissipation Modes

Mode	Start		Condition						Escape
			Oscillator	CPU	Timer, Serial	Register	RAM	I/O terminal	
WAIT	Software	WAIT instruction	Active	Stop	Active	Keep	Keep	Keep	$\overline{\text{STBY}}$, $\overline{\text{RES}}$, $\overline{\text{INT}}$, $\overline{\text{INT}_2}$, each interrupt request of TIMER, TIMER$_2$, SCI
STOP		STOP instruction	Stop	Stop	Stop	Keep	Keep	Keep	$\overline{\text{STBY}}$, $\overline{\text{RES}}$, $\overline{\text{INT}}$, $\overline{\text{INT}_2}$
Standby	Hardware	$\overline{\text{STBY}}$="Low"	Stop	Stop	Stop	Reset	Keep	High impedance	$\overline{\text{STBY}}$="High"

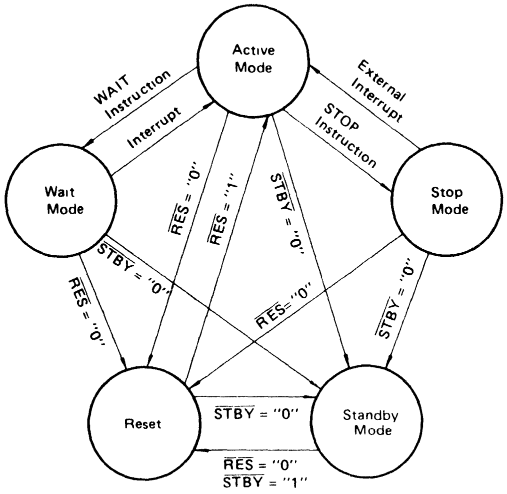


Figure 27 Transitions among Active Mode, Wait Mode, Stop Mode, Standby Mode and Reset

■BIT MANIPULATION

The MCU can use a single instruction (BSET or BCLR) to set or clear one bit of the RAM within page 0 or an I/O port (except the write-only registers such as the data direction register) Every bit of memory or I/O within page 0 ($00 ~ $FF) can be tested by the BRSET or BRCLR instruction, depending on the result of the test, the program can branch to required destinations Since bits in the RAM on page 0, or I/O can be manipulated, the user may use a bit within the RAM on page 0 as a flag or handle a single I/O bit as an independent I/O terminal Fig 28 shows an example of bit manipulation and the validity of test instructions In the example, the program is configured assuming that bit 0 of port A is connected to a zero cross detector circuit and bit 1 of the same port to the trigger of a triac

The program shown can activate the triac within a time of 10μs from zero-crossing through the use of only 7 bytes on the ROM The on-chip timer provides a required time of delay and pulse width modulation of power is also possible

```
              ⋮
SELF 1    BRCLR 0, PORT A, SELF 1
          BSET 1, PORT A
          BCLR 1, PORT A
              ⋮
```

Figure 28 Example of Bit Manipulation

■ADDRESSING MODES

Ten different addressing modes are available to the MCU.

• Immediate

See Fig 29. The immediate addressing mode provides access to a constant which does not vary during execution of the program.

This access requires an instruction length of 2 bytes. The effective address (EA) is PC and the operand is fetched from the byte that follows the operation code

• Direct

See Fig 30 In the direct addressing mode, the address of the operand is contained in the 2nd byte of the instruction. The user can gain direct access to memory up to the lower 255th address 192 byte RAM and I/O registers are on page 0 of address space so that the direct addressing mode may be utilized

• Extended

See Fig 31. The extended addressing is used for referencing to all addresses of memory. The EA is the contents of the 2 bytes that follow the operation code. An extended addressing instruction requires a length of 3 bytes

• Relative

See Fig 32. The relative addressing mode is used with branch instructions only. When a branch occurs, the program counter is loaded with the contents of the byte following the operation code. EA = (PC) + 2 + Rel., where Rel. indicates a signed 8-bit data following the operation code. If no branch occurs, Rel. = 0. When a branch occurs, the program jumps to any byte in the range +129 to −127 A branch instruction requires a length of 2 bytes

• Indexed (No Offset)

See Fig 33 The indexed addressing mode allows access up to the lower 255th address of memory. In this mode, an instruction requires a length of one byte. The EA is the contents of the index register

• Indexed (8-bit Offset)

See Fig. 34. The EA is the contents of the byte following the operation code, plus the contents of the index register. This mode allows access up to the lower 511th address of memory. Each instruction when used in the index addressing mode (8-bit offset) requires a length of 2 bytes.

• Indexed (16-bit Offset)

See Fig. 35. The contents of the 2 bytes following the operation code are added to content of the index register to compute the value of EA. In this mode, the complete memory can be accessed. When used in the indexed addressing mode (16-bit offset), an instruction must be 3 bytes long.

• Bit Set/Clear

See Fig. 36. This addressing mode is applied to the BSET and BCLR instructions that can set or clear any bit on page 0. The lower 3 bits of the operation code specify the bit to be set or cleared. The byte that follows the operation code indicates an address within page 0.

• Bit Test and Branch

See Fig. 37. This addressing mode is applied to the BRSET and BRCLR instructions that can test any bit within page 0 and can be branched in the relative addressing mode. The byte to be tested is addressed depending on the contents of the byte following the operation code. Individual bits within the byte to be tested are specified by the lower 3 bits of the operation code. The 3rd byte represents a relative value which will be added to the program counter when a branch condition is established. Each of these instructions should be 3 bytes long. The value of the test bit is written in the carry bit of the condition code register.

• Implied

See Fig. 38. This mode involves no EA. All information needed for execution of an instruction is contained in the operation code. Direct manipulation on the accumulator and index register is included in the implied addressing mode. Other instructions such as SWI and RTI are also used in this mode. All instructions used in the implied addressing mode should have a length of one byte.

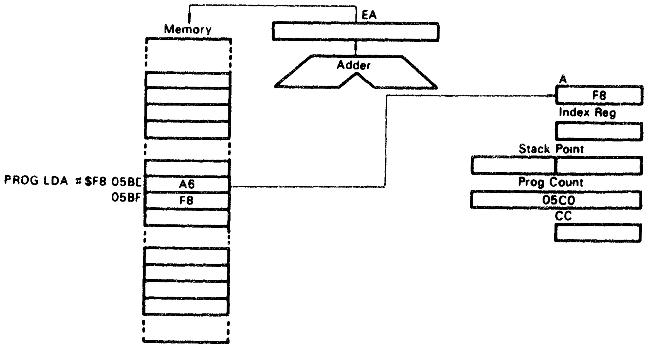


Figure 29 Example of Immediate Addressing

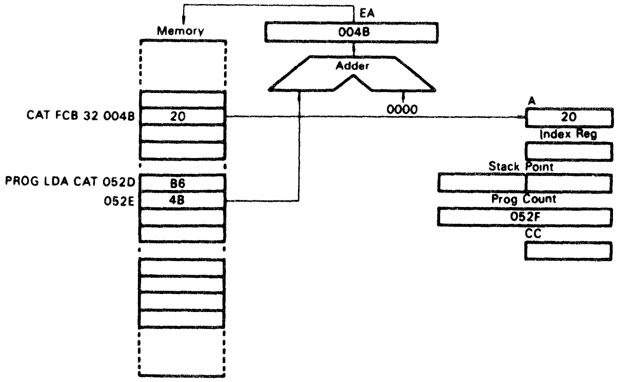


Figure 30 Example of Direct Addressing

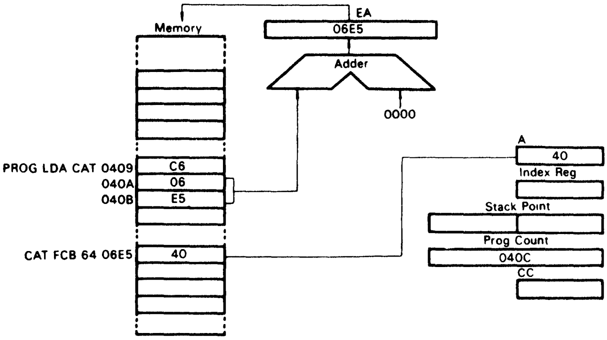


Figure 31 Example of Extended Addressing

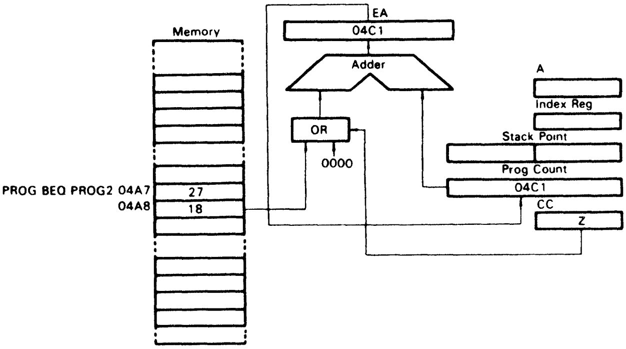


Figure 32 Example of Relative Addressing

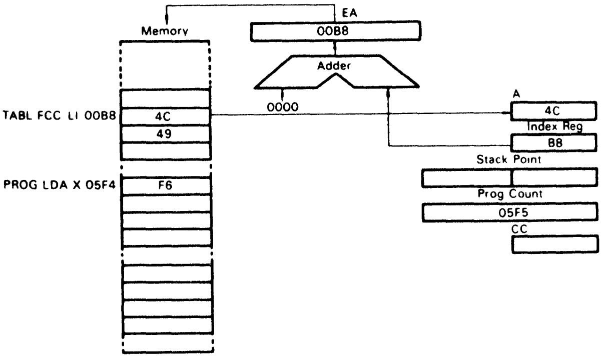


Figure 33 Example of Indexed (No Offset) Addressing

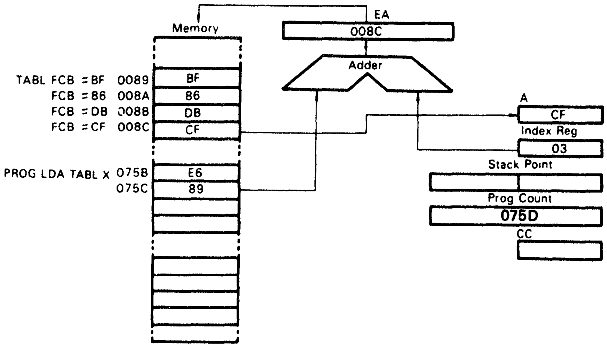


Figure 34 Example of Index (8-bit Offset) Addressing

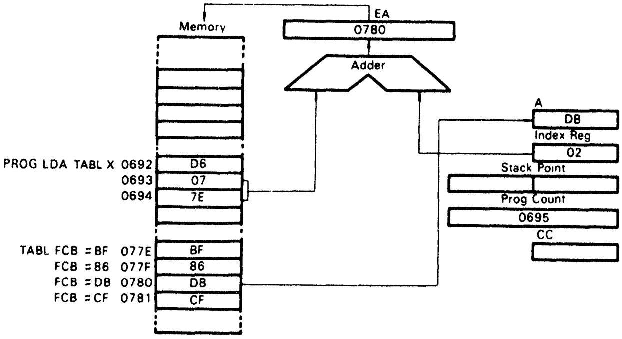


Figure 35 Example of Index (16-bit Offset) Addressing

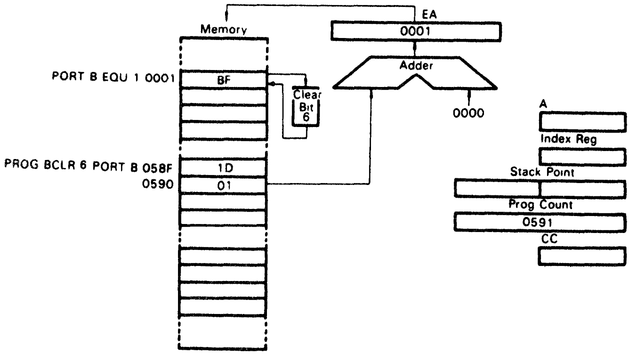


Figure 36 Example of Bit Set/Clear Addressing

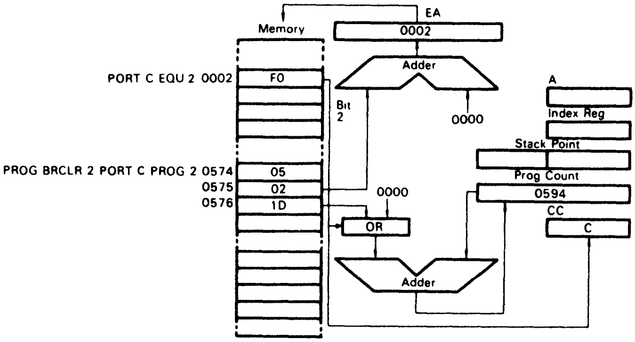


Figure 37 Example of Bit Test and Branch Addressing

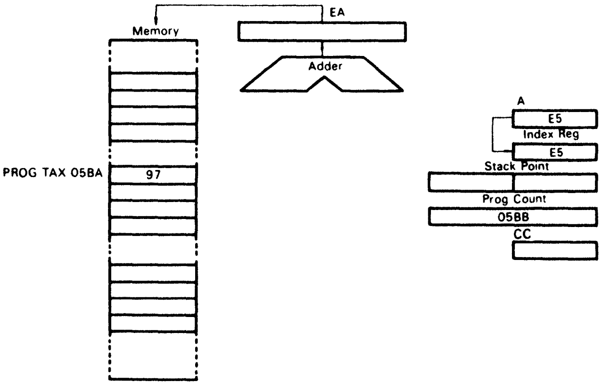


Figure 38 Example of Implied Addressing

■ INSTRUCTION SET

There are 62 basic instructions available to the HD6305Y MCU. They can be classified into five categories: register/memory, read/modify/write, branch, bit manipulation, and control The details of each instruction are described in Tables 5 through 11.

• Register/Memory Instructions

Most of these instructions use two operands. One operand is either an accumulator or index register. The other is derived from memory using one of the addressing modes used on the HD6305Y MCU. There is no register operand in the unconditional jump instruction (JMP) and the subroutine jump instruction (JSR). See Table 5.

• Read/Modify/Write Instructions

These instructions read a memory or register, then modify or test its contents, and write the modified value into the memory or register. Zero test instruction (TST) does not write data, and is handled as an exception in the read/modify/write group. See Table 6.

• Branch Instructions

A branch instruction branches from the program sequence in progress if a particular condition is established. See Table 7.

• Bit Manipulation Instructions

These instructions can be used with any bit located up to the lower 255th address of memory. Two groups are available; one for setting or clearing and the other for bit testing and branching. See Table 8.

• Control Instructions

The control instructions control the operation of the MCU which is executing a program. See Table 9.

• List of Instructions in Alphabetical Order

Table 10 lists all the instructions used on the HD6305Y MCU in the alphabetical order.

• Operation Code Map

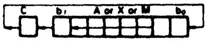
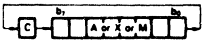
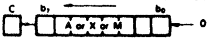
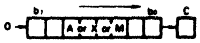
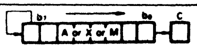
Table 11 shows the operation code map for the instructions used on the MCU.

Table 5 Register/Memory Instructions

Operations	Mnemonic	Addressing Modes																		Boolean/ Arithmetic Operation	Condition Code				
		Immediate			Direct			Extended			Indexed (No Offset)			Indexed (8-Bit Offset)			Indexed (16-Bit Offset)								
		OP	#	~	OP	#	~	OP	#	~	OP	#	~	OP	#	~	OP	#	~		H	I	N	Z	C
Load A from Memory	LDA	A6	2	2	B6	2	3	C6	3	4	F6	1	3	E6	2	4	D6	3	5	M→A	●	●	∧	∧	●
Load X from Memory	LDX	AE	2	2	BE	2	3	CE	3	4	FE	1	3	EE	2	4	DE	3	5	M→X	●	●	∧	∧	●
Store A in Memory	STA	-	-	-	B7	2	3	C7	3	4	F7	1	4	E7	2	4	D7	3	5	A→M	●	●	∧	∧	●
Store X in Memory	STX		-	-	BF	2	3	CF	3	4	FF	1	4	EF	2	4	DF	3	5	X→M	●	●	∧	∧	●
Add Memory to A	ADD	AB	2	2	BB	2	3	CB	3	4	FB	1	3	EB	2	4	DB	3	5	A+M→A	∧	●	∧	∧	∧
Add Memory and Carry to A	ADC	A9	2	2	B9	2	3	C9	3	4	F9	1	3	E9	2	4	D9	3	5	A+M+C→A	∧	●	∧	∧	∧
Subtract Memory	SUB	A0	2	2	B0	2	3	C0	3	4	F0	1	3	E0	2	4	D0	3	5	A−M→A	●	●	∧	∧	∧
Subtract Memory from A with Borrow	SBC	A2	2	2	B2	2	3	C2	3	4	F2	1	3	E2	2	4	D2	3	5	A−M−C→A	●	●	∧	∧	∧
AND Memory to A	AND	A4	2	2	B4	2	3	C4	3	4	F4	1	3	E4	2	4	D4	3	5	A · M→A	●	●	∧	∧	●
OR Memory with A	ORA	AA	2	2	BA	2	3	CA	3	4	FA	1	3	EA	2	4	DA	3	5	A+M→A	●	●	∧	∧	●
Exclusive OR Memory with A	EOR	A8	2	2	B8	2	3	C8	3	4	F8	1	3	E8	2	4	D8	3	5	A+M→A	●	●	∧	∧	●
Arithmetic Compare A with Memory	CMP	A1	2	2	B1	2	3	C1	3	4	F1	1	3	E1	2	4	D1	3	5	A−M	●	●	∧	∧	∧
Arithmetic Compare X with Memory	CPX	A3	2	2	B3	2	3	C3	3	4	F3	1	3	E3	2	4	D3	3	5	X−M	●	●	∧	∧	∧
Bit Test Memory with A (Logical Compare)	BIT	A5	2	2	B5	2	3	C5	3	4	F5	1	3	E5	2	4	D5	3	5	A M	●	●	∧	∧	●
Jump Unconditional	JMP				BC	2	2	CC	3	3	FC	1	2	EC	2	3	DC	3	4		●	●	●	●	●
Jump to Subroutine	JSR				BD	2	5	CD	3	6	FD	1	5	ED	2	5	DD	3	6		●	●	●	●	●

Symbols. Op = Operation
= Number of bytes
~ = Number of cycles

Table 6 Read/Modify/Write Instructions

Operations	Mnemonic	Addressing Modes															Boolean/Arithmetic Operation	Condition Code				
		Implied(A)			Implied(X)			Direct			Indexed (No Offset)			Indexed (8 Bit Offset)								
		OP	#	~	OP	#	~	OP	#	~	OP	#	~	OP	#	~		H	I	N	Z	C
Increment	INC	4C	1	2	5C	1	2	3C	2	5	7C	1	5	6C	2	6	A+1→A or X+1→X or M+1→M	●	●	∧	∧	●
Decrement	DEC	4A	1	2	5A	1	2	3A	2	5	7A	1	5	6A	2	6	A−1→A or X−1→X or M−1→M	●	●	∧	∧	●
Clear	CLR	4F	1	2	5F	1	2	3F	2	5	7F	1	5	6F	2	6	00→A or 00→X or 00→M	●	●	0	1	●
Complement	COM	43	1	2	53	1	2	33	2	5	73	1	5	63	2	6	$\overline{A}$→A or $\overline{X}$→X or $\overline{M}$→M	●	●	∧	∧	1
Negate (2's Complement)	NEG	40	1	2	50	1	2	30	2	5	70	1	5	60	2	6	00−A→A or 00−X→X or 00−M→M	●	●	∧	∧	∧
Rotate Left Thru Carry	ROL	49	1	2	59	1	2	39	2	5	79	1	5	69	2	6		●	●	∧	∧	∧
Rotate Right Thru Carry	ROR	46	1	2	56	1	2	36	2	5	76	1	5	66	2	6		●	●	∧	∧	∧
Logical Shift Left	LSL	48	1	2	58	1	2	38	2	5	78	1	5	68	2	6		●	●	∧	∧	∧
Logical Shift Right	LSR	44	1	2	54	1	2	34	2	5	74	1	5	64	2	6		●	●	0	∧	∧
Arithmetic Shift Right	ASR	47	1	2	57	1	2	37	2	5	77	1	5	67	2	6		●	●	∧	∧	∧
Arithmetic Shift Left	ASL	48	1	2	58	1	2	38	2	5	78	1	5	68	2	6	Equal to LSL	●	●	∧	∧	∧
Test for Negative or Zero	TST	4D	1	2	5D	1	2	3D	2	4	7D	1	4	6D	2	5	A−00 or X−00 or M−00	●	●	∧	∧	●

Symbols Op = Operation
= Number of bytes
~ = Number of cycles

Table 7 Branch Instructions

Operations	Mnemonic	Addressing Modes			Branch Test	Condition Code				
		Relative								
		OP	#	~		H	I	N	Z	C
Branch Always	BRA	20	2	3	None	●	●	●	●	●
Branch Never	BRN	21	2	3	None	●	●	●	●	●
Branch IF Higher	BHI	22	2	3	C+Z=0	●	●	●	●	●
Branch IF Lower or Same	BLS	23	2	3	C+Z=1	●	●	●	●	●
Branch IF Carry Clear	BCC	24	2	3	C=0	●	●	●	●	●
(Branch IF Higher or Same)	(BHS)	24	2	3	C=0	●	●	●	●	●
Branch IF Carry Set	BCS	25	2	3	C=1	●	●	●	●	●
(Branch IF Lower)	(BLO)	25	2	3	C=1	●	●	●	●	●
Branch IF Not Equal	BNE	26	2	3	Z=0	●	●	●	●	●
Branch IF Equal	BEQ	27	2	3	Z=1	●	●	●	●	●
Branch IF Half Carry Clear	BHCC	28	2	3	H=0	●	●	●	●	●
Branch IF Half Carry Set	BHCS	29	2	3	H=1	●	●	●	●	●
Branch IF Plus	BPL	2A	2	3	N=0	●	●	●	●	●
Branch IF Minus	BMI	2B	2	3	N=1	●	●	●	●	●
Branch IF Interrupt Mask Bit is Clear	BMC	2C	2	3	I=0	●	●	●	●	●
Branch IF Interrupt Mask Bit is Set	BMS	2D	2	3	I=1	●	●	●	●	●
Branch IF Interrupt Line is Low	BIL	2E	2	3	INT=0	●	●	●	●	●
Branch IF Interrupt Line is High	BIH	2F	2	3	INT=1	●	●	●	●	●
Branch to Subroutine	BSR	AD	2	5	——	●	●	●	●	●

Symbols· Op = Operation
# = Number of bytes
~ = Number of cycles

Table 8 Bit Manipulation Instructions

Operations	Mnemonic	Addressing Modes						Boolean/ Arithmetic Operation	Branch Test	Condition Code				
		Bit Set/Clear			Bit Test and Branch									
		OP	#	~	OP	#	~			H	I	N	Z	C
Branch IF Bit n is set	BRSET n(n=0·· 7)	–	–	–	2·n	3	5	——	Mn=1	●	●	●	●	^
Branch IF Bit n is clear	BRCLR n(n=0···7)	–	–	–	01+2·n	3	5	——	Mn=0	●	●	●	●	^
Set Bit n	BSET n(n=0···7)	10+2·n	2	5	–	–	–	1→Mn	——	●	●	●	●	●
Clear Bit n	BCLR n(n=0 · 7)	11+2·n	2	5	–	–	–	0→Mn	——	●	●	●	●	●

Symbols: Op = Operation
# = Number of bytes
~ = Number of cycles

Table 9 Control Instructions

Operations	Mnemonic	Addressing Modes			Boolean Operation	Condition Code				
		Implied								
		OP	#	~		H	I	N	Z	C
Transfer A to X	TAX	97	1	2	A→X	●	●	●	●	●
Transfer X to A	TXA	9F	1	2	X→A	●	●	●	●	●
Set Carry Bit	SEC	99	1	1	1→C	●	●	●	●	1
Clear Carry Bit	CLC	98	1	1	0→C	●	●	●	●	0
Set Interrupt Mask Bit	SEI	9B	1	2	1→I	●	1	●	●	●
Clear Interrupt Mask Bit	CLI	9A	1	2	0→I	●	0	●	●	●
Software Interrupt	SWI	83	1	10		●	1	●	●	●
Return from Subroutine	RTS	81	1	5		●	●	●	●	●
Return from Interrupt	RTI	80	1	8		?	?	?	?	?
Reset Stack Pointer	RSP	9C	1	2	$FF→SP	●	●	●	●	●
No-Operation	NOP	9D	1	1	Advance Prog Cntr Only	●	●	●	●	●
Decimal Adjust A	DAA	8D	1	2	Converts binary add of BCD charcters into BCD format	●	●	∧	∧	∧*
Stop	STOP	8E	1	4		●	●	●	●	●
Wait	WAIT	8F	1	4		●	●	●	●	●

Symbols. Op = Operation
= Number of bytes
~ = Number of cycles

* Are BCD characters of upper byte 10 or more? (They are not cleared if set in advance.)

Table 10 Instruction Set (in Alphabetical Order)

Mnemonic	Addressing Modes										Condition Code				
	Implied	Immediate	Direct	Extended	Relative	Indexed (No Offset)	Indexed (8-Bit)	Indexed (16-Bit)	Bit Set/ Clear	Bit Test & Branch	H	I	N	Z	C
ADC		×	×	×		×	×	×			∧	●	∧	∧	∧
ADD		×	×	×		×	×	×			∧	●	∧	∧	∧
AND		×	×	×		×	×	×			●	●	∧	∧	●
ASL	×		×			×	×				●	●	∧	∧	∧
ASR	×		×			×	×				●	●	∧	∧	∧
BCC					×						●	●	●	●	●
BCLR									×		●	●	●	●	●
BCS					×						●	●	●	●	●
BEQ					×						●	●	●	●	●
BHCC					×						●	●	●	●	●
BHCS					×						●	●	●	●	●
BHI					×						●	●	●	●	●
(BHS)					×						●	●	●	●	●
BIH					×						●	●	●	●	●
BIL					×						●	●	●	●	●
BIT		×	×	×		×	×	×			●	●	∧	∧	●
(BLO)					×						●	●	●	●	●
BLS					×						●	●	●	●	●
BMC					×						●	●	●	●	●
BMI					×						●	●	●	●	●
BMS					×						●	●	●	●	●
BNE					×						●	●	●	●	●
BPL					×						●	●	●	●	●
BRA					×						●	●	●	●	●

(to be continued)

Condition Code Symbols.

H	Half Carry (From Bit 3)		C	Carry/Borrow
I	Interrupt Mask		∧	Test and Set if True, Cleared Otherwise
N	Negative (Sign Bit)		●	Not Affected
Z	Zero		?	Load CC Register From Stack

Table 10 Instruction Set (in Alphabetical Order)

	Addressing Modes										Condition Code				
Mnemonic	Implied	Immediate	Direct	Extended	Relative	Indexed (No Offset)	Indexed (8-Bit)	Indexed (16-Bit)	Bit Set Clear	Bit Test & Branch	H	I	N	Z	C
BRN					×						●	●	●	●	●
BRCLR										×	●	●	●	●	∧
BRSET										×	●	●	●	●	∧
BSET									×		●	●	●	●	●
BSR					×						●	●	●	●	●
CLC	×										●	●	●	●	0
CLI	×										●	0	●	●	●
CLR	×		×			×	×				●	●	0	1	●
CMP		×	×	×		×	×	×			●	●	∧	∧	∧
COM	×		×			×	×				●	●	∧	∧	1
CPX		×	×	×		×	×	×			●	●	∧	∧	∧
DAA	×										●	●	∧	∧	∧
DEC	×		×			×	×				●	●	∧	∧	●
EOR		×	×	×		×	×	×			●	●	∧	∧	●
INC	×		×			×	×				●	●	∧	∧	●
JMP			×	×		×	×	×			●	●	●	●	●
JSR			×	×		×	×	×			●	●	●	●	●
LDA		×	×	×		×	×	×			●	●	∧	∧	●
LDX		×	×	×		×	×	×			●	●	∧	∧	●
LSL	×		×			×	×				●	●	∧	∧	∧
LSR	×		×			×	×				●	●	0	∧	∧
NEG	×		×			×	×				●	●	∧	∧	∧
NOP	×										●	●	●	●	●
ORA		×	×	×		×	×	×			●	●	∧	∧	●
ROL	×		×			×	×				●	●	∧	∧	∧
ROR	×		×			×	×				●	●	∧	∧	∧
RSP	×										●	●	●	●	●
RTI	×										?	?	?	?	?
RTS	×										●	●	●	●	●
SBC		×	×	×		×	×	×			●	●	∧	∧	∧
SEC	×										●	●	●	●	1
SEI	×										●	1	●	●	●
STA			×	×		×	×	×			●	●	∧	∧	●
STOP	×										●	●	●	●	●
STX			×	×		×	×	×			●	●	∧	∧	●
SUB		×	×	×		×	×	×			●	●	∧	∧	∧
SWI	×										●	1	●	●	●
TAX	×										●	●	●	●	●
TST	×		×			×	×				●	●	∧	∧	●
TXA	×										●	●	●	●	●
WAIT	×										●	●	●	●	●

Condition Code Symbols

H	Half Carry (From Bit 3)	C	Carry Borrow
I	Interrupt Mask	∧	Test and Set if True, Cleared Otherwise
N	Negative (Sign Bit)	●	Not Affected
Z	Zero	?	Load CC Register From Stack

Table 11 Operation Code Map

	Bit Manipulation		Branch	Read/Modify/Write					Control		Register/Memory							
	Test & Branch	Set Clear	Rel	DIR	A	X	,X1	,X0	IMP	IMP	IMM	DIR	EXT	,X2	,X1	,X0		
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	← HIGH	
0	BRSET0	BSET0	BRA	NEG					RTI*		SUB						0	
1	BRCLR0	BCLR0	BRN	–					RTS*	–	CMP						1	
2	BRSET1	BSET1	BHI	–						-	SBC						2	
3	BRCLR1	BCLR1	BLS	COM					SWI*	-	CPX						3	L
4	BRSET2	BSET2	BCC	LSR					–	--	AND						4	O
5	BRCLR2	BCLR2	BCS	–					--		BIT						5	W
6	BRSET3	BSET3	BNE	ROR					-	-	LDA						6	
7	BRCLR3	BCLR3	BEQ	ASR					-	TAX*	---	STA				STA(+1)	7	
8	BRSET4	BSET4	BHCC	LSL/ASL					-	CLC	EOR						8	
9	BRCLR4	BCLR4	BHCS	ROL					–	SEC	ADC						9	
A	BRSET5	BSET5	BPL	DEC					-	CLI*	ORA						A	
B	BRCLR5	BCLR5	BMI	–					---	SEI*	ADD						B	
C	BRSET6	BSET6	BMC	INC					--	RSP*	-	JMP(−1)					C	
D	BRCLR6	BCLR6	BMS	TST(−1)	TST		TST(−1)		DAA*	NOP	BSR*	JSR(+2)		JSR(+1)		JSR(+2)	D	
E	BRSET7	BSET7	BIL	–					STOP*	–	LDX						E	
F	BRCLR7	BCLR7	BIH	CLR					WAIT*	TXA*	-	STX				STX(+1)	F	
	3/5	2/5	2/3	2/5	1/2	1/2	2/6	1 5	1/*	1/1	2/2	2/3	3/4	3/5	2/4	1/3		

(NOTES) 1 "–" is an undefined operation code

2. The lowermost numbers in each column represent a byte count and the number of cycles required (byte count/number of cycles)
The number of cycles for the mnemonics asterisked (*) is as follows.

RTI	8	TAX	2
RTS	5	RSP	2
SWI	10	TXA	2
DAA	2	BSR	5
STOP	4	CLI	2
WAIT	4	SEI	2

3. The parenthesized numbers must be added to the cycle count of the particular instruction

- **Additional Instructions**

The following new instructions are used on the HD6305Y

DAA Converts the contents of the accumulator into BCD code.

WAIT Causes the MCU to enter the wait mode. For this mode, see the topic, Wait Mode.

STOP Causes the MCU to enter the stop mode. For this mode, see the topic, Stop Mode.

■ PRECAUTION 1—BOARD DESIGN OF OSCILLATION CIRCUIT

When connecting crystal and ceramic resonator with the XTAL and EXTAL pins to oscillate, observe the following in designing the board.

(1) Locate crystal, ceramic resonator, and load capacity C_1 and C_2 as near the LSI as possible. (Induction of noise from outside to the XTAL and EXTAL pins may cause trouble in oscillation.)

(2) Wire the signal lines to the neighbouring XTAL and EXTAL pins as far apart as possible.

(3) Board design of situating signal lines or power supply lines near the oscillator circuit as shown in Fig. 40, should not be used because of trouble in oscillation by induction. The resistor between the XTAL and EXTAL, and pins close to them should be 10M Ω or more. The circuit in Fig 39 is an example of good board design

■ PRECAUTION 2—PROGRAM OF WRITE ONLY REGISTER

Read/Modify/Write instructions are unavailable for changing the contents of Write Only Register (e g. DDR, Data Direction Register of I/O port) of HD6305X, HD6305Y and HD63P05Y.

(1) Data cannot be read from write only register. (e.g. DDR of I/O port)

While read/modify/write instructions are executed in the following sequence.

(i) Reads the contents from appointed address.

(ii) Changes the data which has been read.

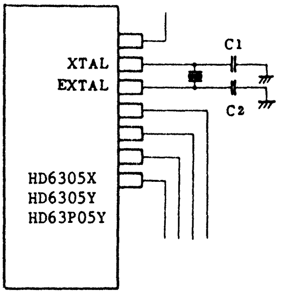


Figure 39 Design of Oscillation Circuit Board

(iii) Turn the data back to the original address.
Thus, read/modify/write instructions cannot be applied to write only register such as DDR.

(2) For the same reason, do not set DDR of I/O port using BSET and BCLR instructions.

(3) Stored instructions (e.g. STA and STX, etc.) are available for writing into the write only register.

■ PRECAUTION 3—SENDING/RECEIVING PROGRAM OF SERIAL DATA

Be careful that malfunction may occur if SDR (SERIAL DATA REGISTER: $0012) is read or written durıng transmitting or receiving serial data.

■ PRECAUTION 4—WAIT/STOP INSTRUCTIONS PROGRAM

When I bit of condition code register is "1" and an interrupt ($\overline{INT}_2$, TIMER/$\overline{INT}_2$) is held, the MCU does not enter into WAID mode by executing the WAIT instruction.

In that case, after the 4 dummy cycles, the MCU executes the next instruction.

In the same way, when external interrupts ($\overline{INT}$, $\overline{INT}_2$) are held at the bit I set, the MCU does not enter into the STOP mode by executing STOP instruction. In that case the MCU executes the next instruction after the 4 dummy cycles.

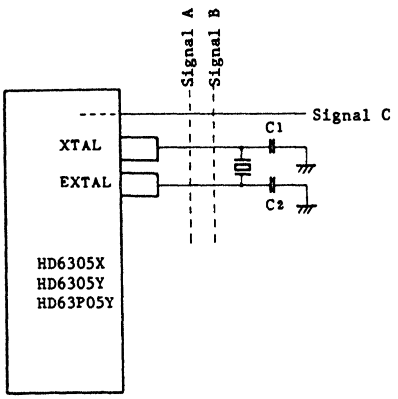


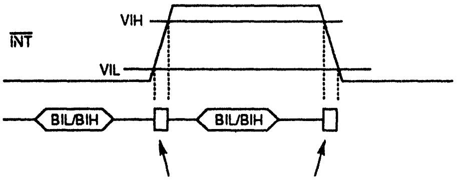
Figure 40 Example of Circuit Causing Trouble in Oscillation

■ PRECAUTION WHEN USING BIL/BIH INSTRUCTION

(1) Execute Instruction after the $\overline{INT}$ Voltage level has stabilized above V_{IH} or below V_{IL}.

(2) $\overline{INT}$ voltage level needs to be stabilized while BIL/BIH Instruction Execution.

There may be a malfunction by glitch on control signal if BIL/BIH Instruction Execution has exercized in unstablized $\overline{INT}$ signal level.



Avoid BIL/BIH Instruction Execution

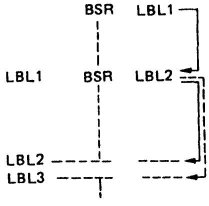
■ PRECAUTION TO USE BSR

If there is 2nd BSR programmed on the address which is directed by first BSR, 2nd BSR may not be executed correctly. For this reason, BSR should not be programmed on the address which is directed by first BSR.

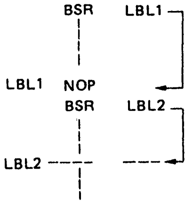
If necessary, please program as following.

(1) On the address which first BSR directed, NOP instruction should be ınserted before second BSR.

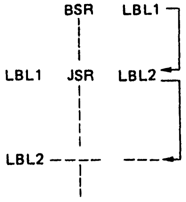
(2) On the address which first BSR directed, JSR instruction should be programmed instead of 2nd BSR.



example of malfunction of 2nd BSR execution



example of counter measure (NOP ıs inserted)



example of counter measure (JSR is used instead of BSR)

HD63B09, HD63C09
CMOS MPU (Micro Processing Unit)

Description

The HD6309 is the highest 8-bit microprocessor of HMCS6800 family, which is compatible with the conventional HD6809.

The HD6309 has hardware and software features which make it an ideal processor for higher level language execution or standard controller applications.

The HD6309 is complete CMOS device and its power dissipation is extremely low. Moreover, the SYNC and CWAI instruction makes low power application possible.

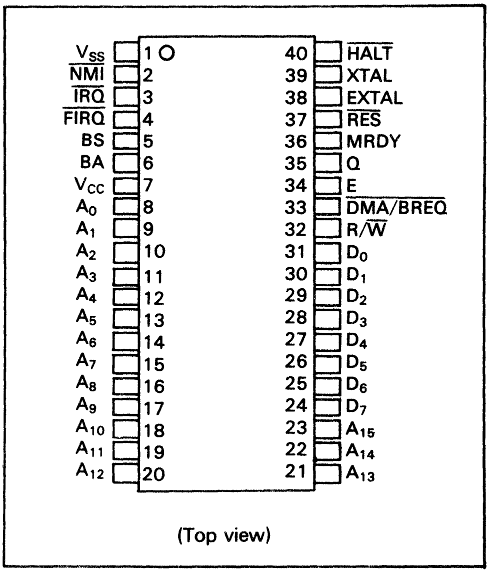
Features

- Hardware
  - Interfaces with all HMCS6800 peripherals
  - DMA transfer with no auto-refresh cycle
- Software: object code compatible with the HD6809
- Low power consumption mode (Sleep mode)
  - SYNC state of SYNC Instruction
  - WAIT state of CWAI Instruction
- On chip oscillator
- Wide operation range: f = 0.5 to 3 MHz (V_{CC} = 5 V±10%)

Type of Products

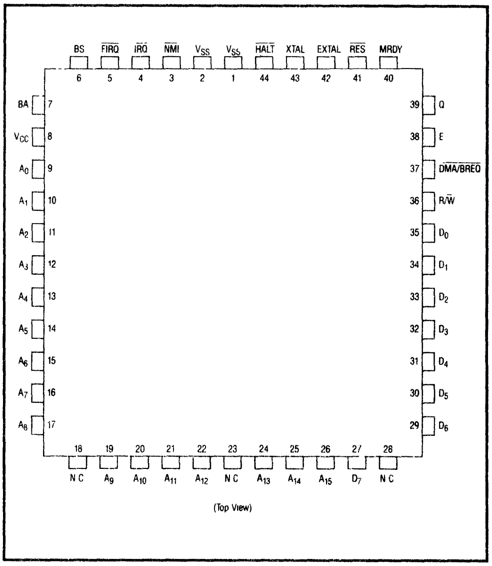
Type No.	Bus Timing
HD63B09	2.0 MHz
HD63C09	3.0 MHz

Pin Arrangement



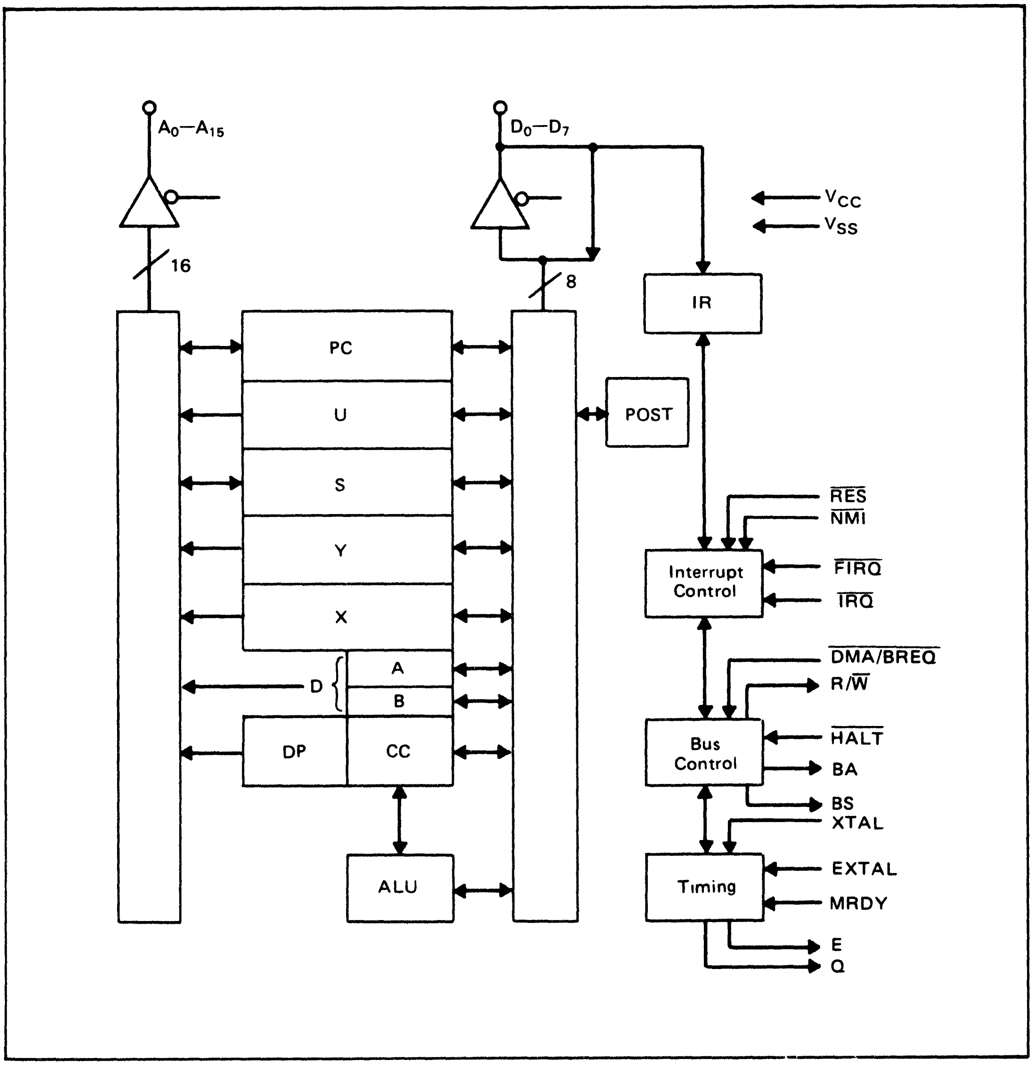
(Top view)

- PLCC package available



(Top View)

Block Diagram



Programming Model

As shown in figure 1, the HD6309 adds three registers to the set available in the HD6800. The added registers are a direct page register, the user stack pointer and a second index register.

Accumulators (A, B, D)

The A and B registers are general purpose accumulators which are used for arithmetic calculations and manipulation of data.

Certain instructions concatenate the A and B registers to form a single 16-bit accumulator. This is referred to as the D register. It is formed with the A register as the most significant byte.

Direct Page Register (DP)

The direct page register of the HD6309 serves to enhance the direct addressing mode. The contents of this register appears at the higher address outputs (A_8 $-A_{15}$) during direct addressing instruction execution. This allows the direct mode to be used at any place in memory, under program control. To ensure HD6800 compatibility, all bits of this register are cleared during processor reset.

Index Registers (X, Y)

The index registers are used in indexed mode addressing. The 16-bit address in this register takes part in the calculation of effective addresses. This address may be used to point to data directly or may be modified by an optional constant or register offset. In some indexed modes, the contents of the index register are incremented or decremented to point to the next item of tabular data. All four pointer registers (X, Y, U, S) may be used as index registers.

Stack Pointer (U, S)

The hardware stack pointer (S) is used automatically by the processor during subroutine calls and interrupts. The stack pointers of the HD6309 point to the top of the stack, in contrast to the HD6800 stack pointer, which pointed to the next free location on the stack. The user stack pointer (U) is controlled exclusively by the programmer thus allowing arguments to be passed to and from subroutines with ease. Both stack pointers have the same indexed mode addressing capabilities as the X and Y registers, but also support push and pull instructions. This allows the HD6309 to be used efficiently as a stack processor, greatly enhancing its ability to support higher level languages and modular programming.

Note: The stack pointers of the HD6309 point to the top of the stack, in contrast to the HD6800 stack pointer, which pointed to the next free location on stack.

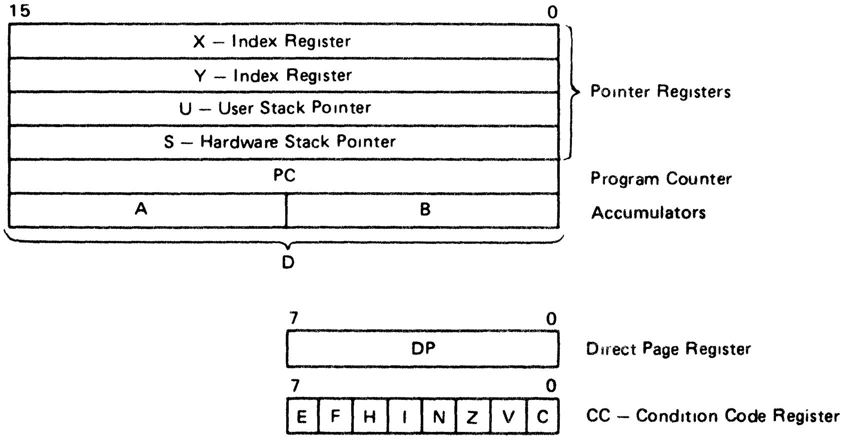


Figure 1. Programming Model of The Microprocessing Unit

Program Counter (PC)

The program counter is used by the processor to point to the address of the next instruction to be executed by the processor. Relative addressing is provided allowing the program counter to be used like an index register in some situations.

Condition Code Register (CC)

The condition code register defines the state of the processor at any given time. See figure 2.

Condition Code Register Description

Bit 0 (C)

Bit 0 is the carry flag. It is usually the carry from the binary ALU. C is also used to represent a 'borrow' from subtract-like instructions (CMP, NEG, SUB, SBC). Then, it is the complement of the carry from the binary ALU.

Bit 1 (V)

Bit 1 is the overflow flag. It is set to a one by an operation which causes a signed two's complement arithmetic overflow. This overflow is detected in an operation in which the carry from the MSB in the ALU does not match the carry from the MSB minus 1.

Bit 2 (Z)

Bit 2 is the zero flag. It is set to one if the result of the previous operation was identically zero.

Bit 3 (N)

Bit 3 is the negative flag. It contains exactly the value of the MSB of the result of the preceding operation. Thus, a negative two's-complement result will leave N set to one.

Bit 4 (I)

Bit 4 is the $\overline{\text{IRQ}}$ mask bit. The processor will not recognize interrupts from the $\overline{\text{IRQ}}$ line if this bit is set to one. $\overline{\text{NMI}}$, $\overline{\text{FIRQ}}$, $\overline{\text{IRQ}}$, $\overline{\text{RES}}$, and SWI all set I to one; SWI2 and SWI3 do not affect I.

Bit 5 (H)

Bit 5 is the half-carry bit. It is used to indicate a carry from bit 3 in the ALU as a result of an 8-bit addition only (ADC or ADD). This bit is used by the DAA instruction to perform a BCD decimal add adjust operation. The state of this flag is undefined in all subtract-like instructions.

Bit 6 (F)

Bit 6 is the $\overline{\text{FIRQ}}$ mask bit. The processor will not recognize interrupts from the $\overline{\text{FIRQ}}$ line if this bit is a one. $\overline{\text{NMI}}$, $\overline{\text{FIRQ}}$, SWI, and $\overline{\text{RES}}$ all set F to one. $\overline{\text{IRQ}}$, SWI2 and SWI3 do not affect F.

Bit 7 (E)

Bit 7 is the entire flag. Set to one, it indicates that the complete machine state (all the registers) was stacked, as opposed to the subset state (PC and CC). The E bit of the stacked CC is used on a return from interrupt (RTI) to determine the extent of the unstacking. Therefore, the current E left in the condition code register represents past action.

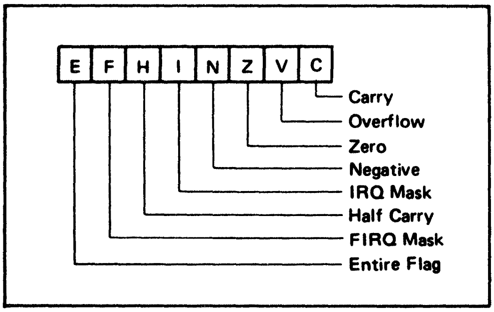


Figure 2. Condition Code Register Format

Signal Description

Power (Vss, V_{CC})

Two pins supply power to the part: Vss is ground or 0 volts, while V_{CC} is +5.0 V ±10%.

Address Bus (A_0 $-A_{15}$)

Sixteen pins output address information from the MPU onto the address bus. When the processor does not require the bus for a data transfer, it will output address $FFFF_{16}$, R/$\overline{W}$=high, and BS=low. This is a "dummy access" or $\overline{VMA}$ cycle (see figures 25 and 26). All address bus drivers are made high impedance when the bus available output (BA) is high. Each pin will drive one Schottky TTL load or four LS TTL loads, and typically 90 pF.

Data Bus (D_0 $-D_7$)

These eight pins provide communication with the system bi-directional data bus. Each pin will drive one Schottky TTL load or four LS TTL loads, and typically 130 pF.

Read/Write (R/$\overline{W}$)

This signal indicates the direction of data transfer on the data bus. A low indicates that the MPU is writing data onto the data bus. R/$\overline{W}$ is made high impedance when BA is high. Refer to figures 25 and 26.

Reset ($\overline{RES}$)

A low level on this Schmitt-trigger input for greater than one bus cycle will reset the MPU, as shown in figure 3. The reset vectors are fetched from locations $FFFE_{16}$ and $FFFF_{16}$ (table 2) when interrupt acknowledge is true, ($\overline{BA}$ · BS=1). During initial power-on, the reset line should be held low until the clock oscillator is fully operational. See figure 4.

Because the HD6309 reset pin has a Schmitt-trigger input with a threshold voltage higher than that of standard peripherals, a simple R/C network may be used to reset the entire system. This higher

Table 1. Pin Description

Symbol	Pin No.	I/O	Function
Vss	1		Ground
$\overline{NMI}$	2	I	Non maskable interrupt
$\overline{IRQ}$	3	I	Interrupt request
$\overline{FIRQ}$	4	I	Fast interrupt request
BS, BA	5, 6	O	Bus status, Bus available
Vcc	7		+5 V power supply
A_0 $-A_{15}$	8−23	O	Address bus, bits 0-15
D_7 $-D_0$	24−31	I/O	Data bus, bits 0-7
R/$\overline{W}$	32	O	Read / Write output
$\overline{DMA/BREQ}$	33	I	DMA Bus request
E, Q	34, 35	O	Clock signal
MRDY	36	I	Memory ready
$\overline{RES}$	37	I	Reset input
EXTAL, XTAL	38, 39	I	Oscillator connection
$\overline{HALT}$	40	I	Halt input

threshold voltage ensures that all peripherals are out of the reset state before the processor.

Halt ($\overline{\text{HALT}}$)

A low level on this input pin will cause the MPU to stop running at the end of the present instruction and remain halted indefinitely without loss of data. When halted, the BA output is driven high indicating the buses are high impedance. BS is also high which indicates the processor is in the halt or bus grant state. While halted, the MPU will not respond to external realtime requests ($\overline{\text{FIRQ}}$, $\overline{\text{IRQ}}$) although $\overline{\text{DMA/BREQ}}$ will always be accepted, and $\overline{\text{NMI}}$ or $\overline{\text{RES}}$ will be latched for later response. During the halt state, Q and E continue to run normally. If the MPU is not running ($\overline{\text{RES}}$), a halted state (BA • BS = 1) can be achieved by pulling $\overline{\text{HALT}}$ low while $\overline{\text{RES}}$ is still low. See figure 5.

Bus Available, Bus Status (BA, BS)

The BA output is an indication of an internal control signal which makes the MOS buses of the MPU high impedance. This signal does not imply that the bus will be available for more than one cycle. When BA goes low, an additional dead cycle will elapse before the MPU acquires the bus.

The BS output signal, when decoded with BA, represents the MPU state.

Interrupt Acknowledge is indicated during both cycles of a hardware vector fetch ($\overline{\text{RES}}$, $\overline{\text{NMI}}$, $\overline{\text{FIRQ}}$, $\overline{\text{IRQ}}$, SWI, SWI2, SWI3). This signal, plus decoding of the lower four address lines, can provide the user with an indication of which interrupt level is being serviced and allow vectoring by device. See Table 2.

Sync Acknowledge is indicated while the MPU is waiting for external synchronization on an interrupt line.

Halt/Bus Grant is true when the HD6309 is in a halt or bus grant condition.

Non Maskable Interrupt ($\overline{\text{NMI}}$)

A negative edge on $\overline{\text{NMI}}$ requests that a nonmaskable interrupt sequence be generated. A nonmaskable interrupt cannot be inhibited by the program, and also has a higher priority than $\overline{\text{FIRQ}}$, $\overline{\text{IRQ}}$ or software interrupts. During recognition of an $\overline{\text{NMI}}$, the entire machine state is saved on the hardware stack. After reset, an $\overline{\text{NMI}}$ will not be recognized until the first program load of the hardware stack pointer (S). The pulse width of $\overline{\text{NMI}}$ low must be at least one E cycle. If the $\overline{\text{NMI}}$ input does not meet the minimum set up with respect to Q, the interrupt will not be recognized until the next cycle See figure 6.

Table 2. Memory Map for Interrupt Vectors

Memory Map for Vector Locations		
MS	**LS**	**Interrupt Vector Description**
FFFE	FFFF	$\overline{\text{RES}}$
FFFC	FFFD	$\overline{\text{NMI}}$
FFFA	FFFB	SWI
FFF8	FFF9	$\overline{\text{IRQ}}$
FFF6	FFF7	$\overline{\text{FIRQ}}$
FFF4	FFF5	SWI2
FFF2	FFF3	SWI3
FFF0	FFF1	Reserved

Table 3. MPU State Definition

BA	BS	MPU State
0	0	Normal (Running)
0	1	Interrupt or RESET Acknowledge
1	0	SYNC Acknowledge
1	1	HALT or Bus Grant

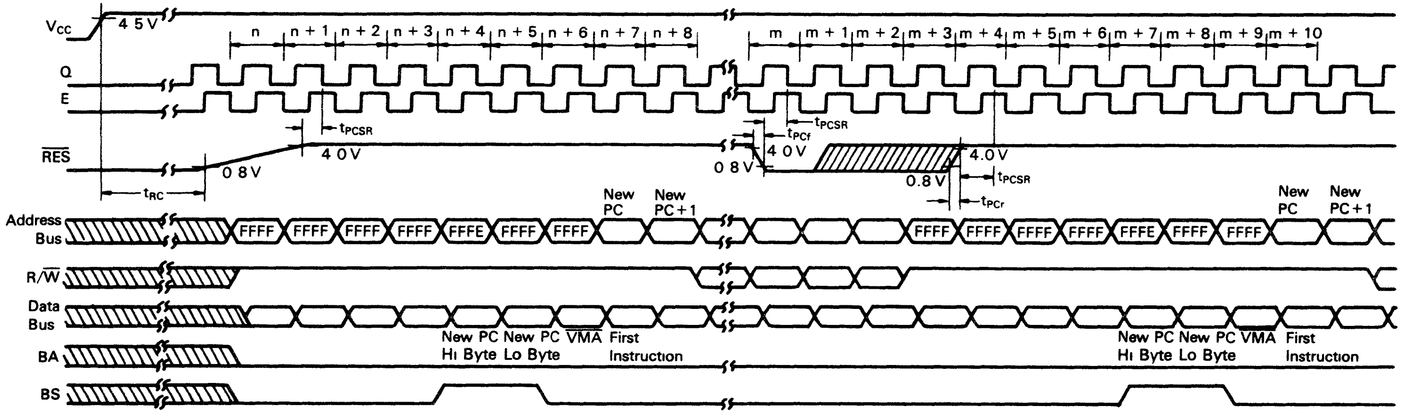
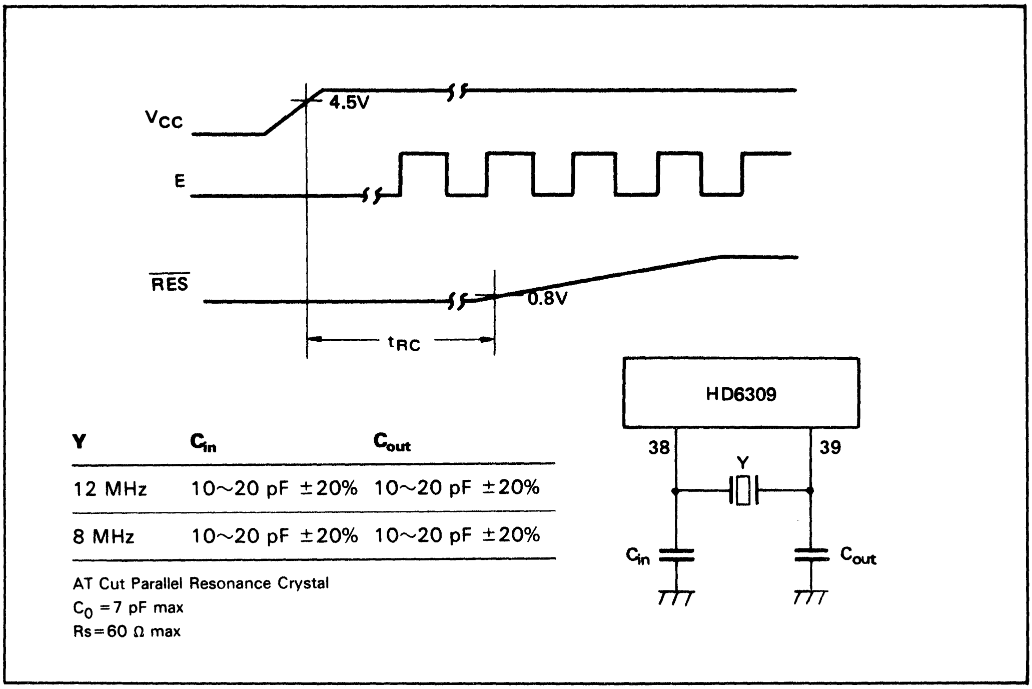


Figure 3. RES Timing



Y	C_{in}	C_{out}
12 MHz	10~20 pF ±20%	10~20 pF ±20%
8 MHz	10~20 pF ±20%	10~20 pF ±20%

AT Cut Parallel Resonance Crystal
C_0 = 7 pF max
Rs = 60 Ω max

Figure 4. Crystal Connections and Oscillator Start Up

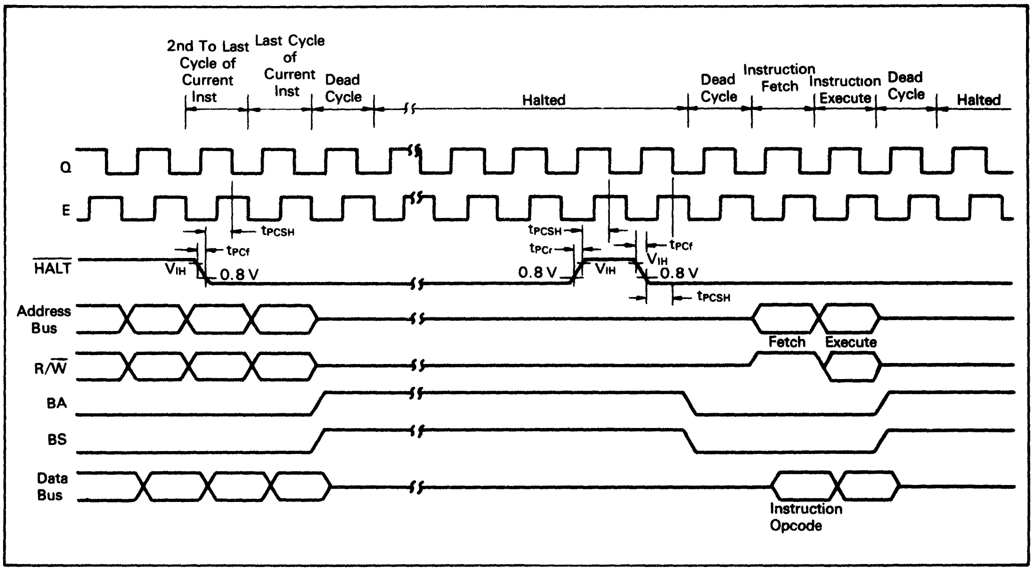


Figure 5. $\overline{\text{HALT}}$ and Single Instruction Execution for System Debug

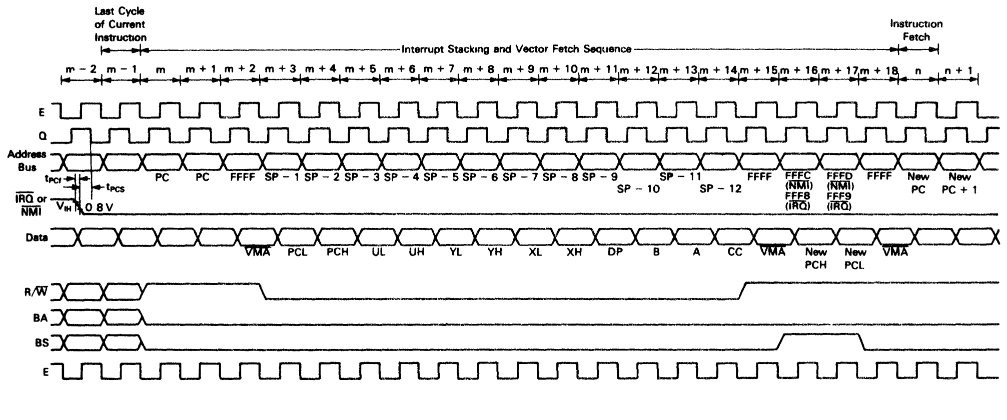


Figure 6. $\overline{IRQ}$ and $\overline{NMI}$ Interrupt Timing

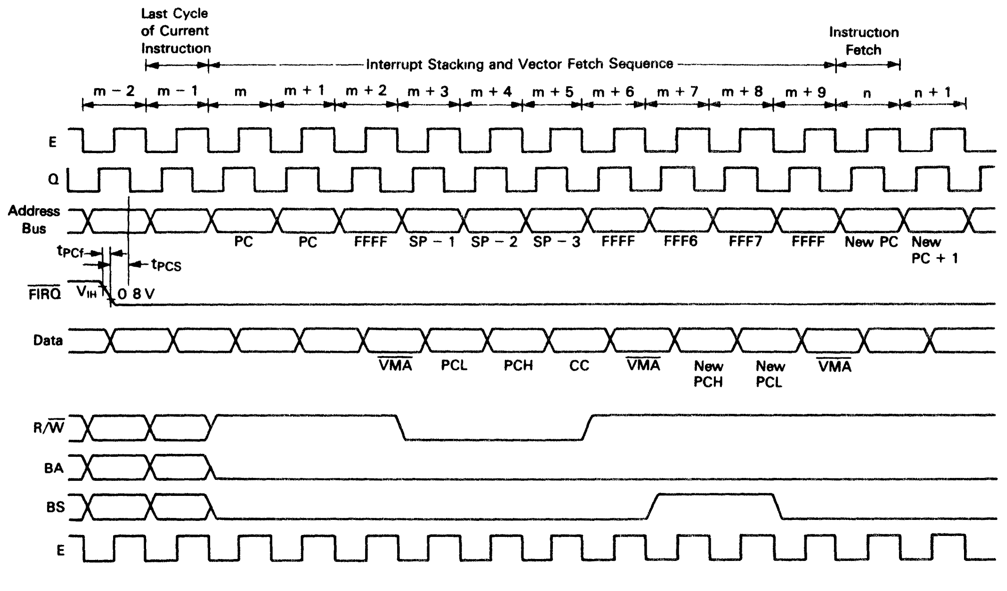


Figure 7. $\overline{FIRQ}$ Interrupt Timing

Fast Interrupt Request (FIRQ)

A low level on FIRQ input will initiate a fast interrupt sequence provided its mask bit (F) in the CC is clear. This sequence has priority over the standard interrupt request (IRQ). It is fast in the sense that it stacks only the contents of the condition code register and the program counter. The interrupt service routine should clear the source of the interrupt before doing an RTI. See figure 7.

Interrupt Request (IRQ)

A low level input on IRQ will initiate an interrupt request sequence provided the mask bit (I) in the CC is clear. Since IRQ stacks the entire machine state it provides a slower response to interrupts than FIRQ. IRQ also has a lower priority than FIRQ. Again, the interrupt service routine should clear the source of the interrupt before doing an RTI. See figure 6.

Note: NMI, FIRQ, and IRQ requests are sampled on the falling edge of Q. One cycle is required for synchronization before these interrupts are recognized. The pending interrupt(s) will not be serviced until completion of the current instruction unless a SYNC or CWAI condition is present. IfIRQ and FIRQ do not remain low until completion of the current instruction they may not be recognized. However, NMI is latched and need only remain low for one cycle.

XTAL, EXTAL

These two pins are connected with parallel resonant fundamental crystal, AT cut. Alternately, the pin EXTAL may be used as a TTL level input for external timing with XTAL floating. The crystal or external frequency is four times the bus frequency. See figure 4. Proper RF layout techniques should be observed in the layout of printed circuit boards.

Note for Board Design of the Oscillation Circuit: In designing the board, the following notes should be taken when the crystal oscillator is used. See figure 8.

1. Crystal oscillator and load capacity Cin, Cout must be placed near the LSI as much as possible. (Normal oscillation may be disturbed when external noise is induced to pin 38 and 39.)
2. Pin 38 and 39 signal line should be wired apart from other signal line as much as possible. Don't wire them in parallel with other lines. (Normal oscillation may be disturbed when E or Q signal feeds back to pin 38 and 39.)

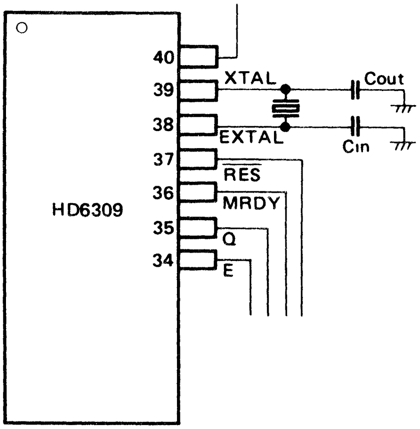


Figure 8. Board Design of the Oscillation Circuit

Designs to be Avoided: A signal line or a power source line must not cross or go near the oscillation circuit line as shown in figure 9 to prevent induction from these lines. The resistance between XTAL, EXTAL and other pins should be over 10 MΩ.

E, Q

E is similar to the HD6800 bus timing signal ϕ_2: Q is a quadrature clock signal which leads E. Q has no parallel on the HD6800. Data is latched on the falling edge of E. Timing for E and Q is shown in figure 10.

Memory Ready (MRDY)

This input control signal allows stretching of E and Q to extend data-access time. E and Q operate normally while MRDY is high. When MRDY is low, E and Q may be stretched in integral multiples of half (1/2) bus cycles, thus allowing interface to slow memories, as shown in figure 11. The maximum stretch is 5 microseconds.

During nonvalid memory access ($\overline{\text{VMA}}$ cycles) MRDY has no effect on stretching E and Q: this inhibits slowing the processor during "don't care" bus accesses. MRDY may also be used to stretch clocks (for slow memory) when bus control has been transferred to an external device (through the use of $\overline{\text{HALT}}$ and $\overline{\text{DMA/BREQ}}$).

MRDY also stretches E and Q during dead cycles.

DMA Bus Request ($\overline{\text{DMA/BREQ}}$)

The $\overline{\text{DMA/BREQ}}$ input provides a method of suspending execution and acquiring the MPU bus for another use, as shown in figure 12. Typical uses include DMA and dynamic memory refresh.

Transition of $\overline{\text{DMA/BREQ}}$ should occur during Q. A low level on this pin will stop instruction execution at the end of the current cycle. The MPU will acknowledge $\overline{\text{DMA/BREQ}}$ by setting BA and BS to high level. The HD6309 does not perform the auto-refresh executed in the HD6809. See figure 13.

Typically, the DMA controller will request to use the bus by asserting $\overline{\text{DMA/BREQ}}$ pin low on the leading edge of E. When the MPU replies by setting BA and BS to one, that cycle will be a dead cycle used to transfer bus mastership to the DMA controller.

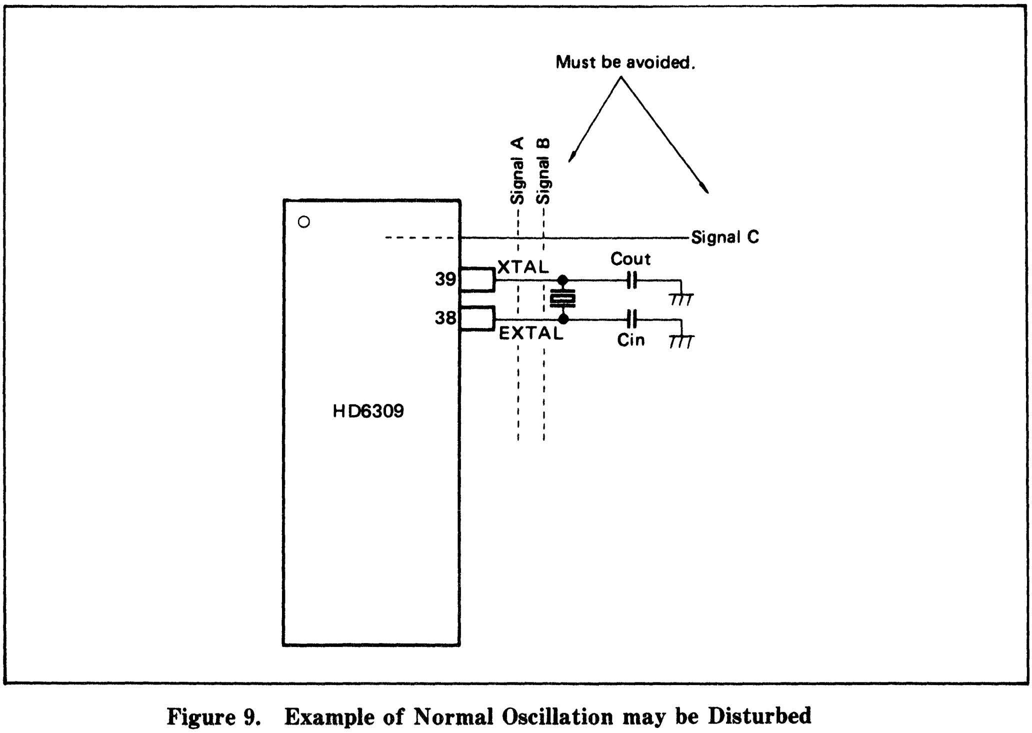


Figure 9. Example of Normal Oscillation may be Disturbed

False memory accesses may be prevented during dead cycles by developing a system $\overline{\text{DMAVMA}}$ signal which is low in any cycle when BA has changed.

When BA goes low (a result of $\overline{\text{DMA/BREQ}}$= high), another dead cycle will elapse before the MPU accesses memory, to allow transfer of bus mastership without contention.

The $\overline{\text{DMA/BREQ}}$ input should be tied high during reset state.

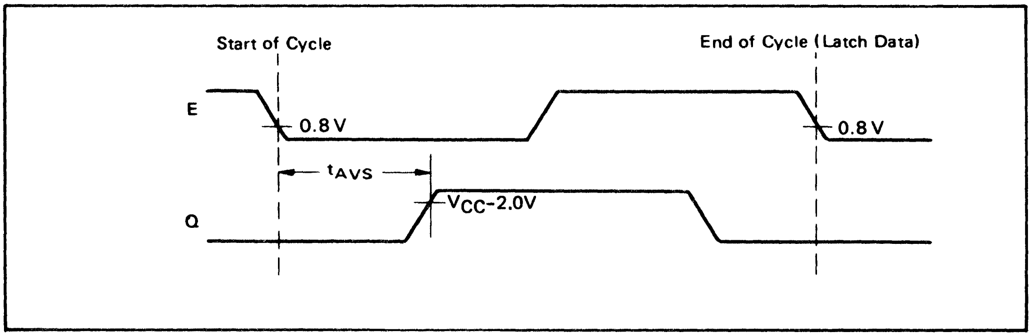


Figure 10. E/Q Relationship

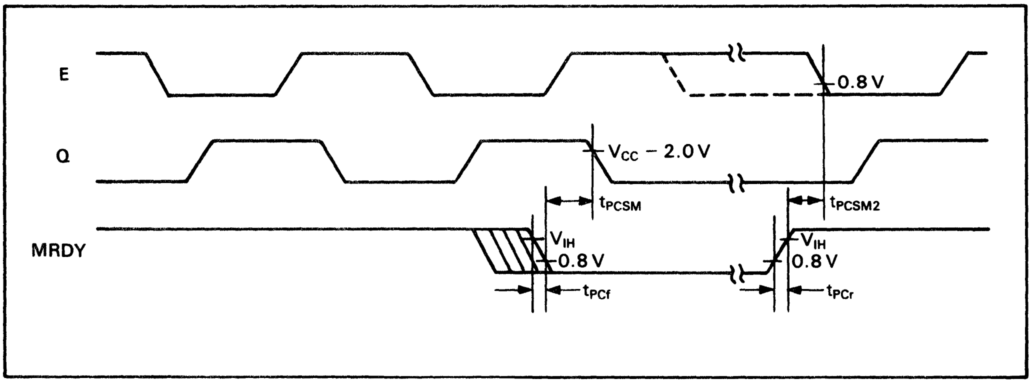
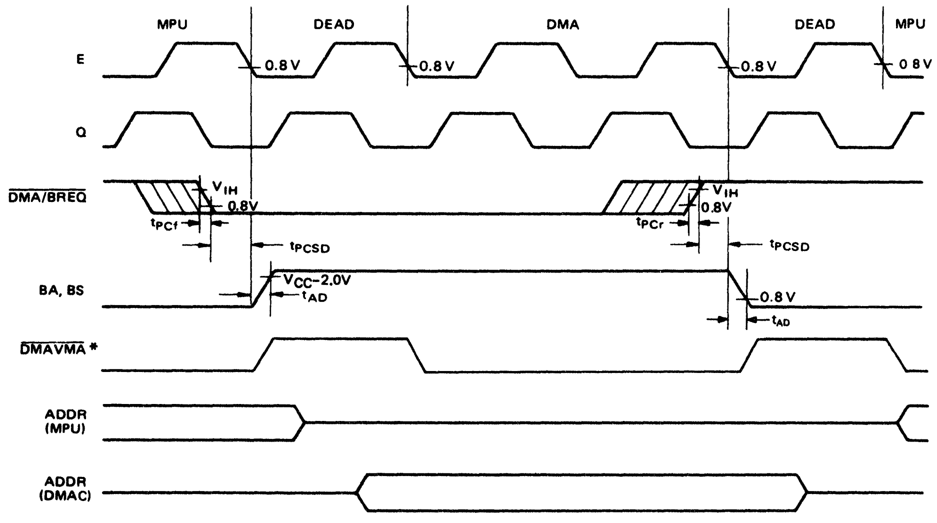


Figure 11. MRDY Clock Stretching

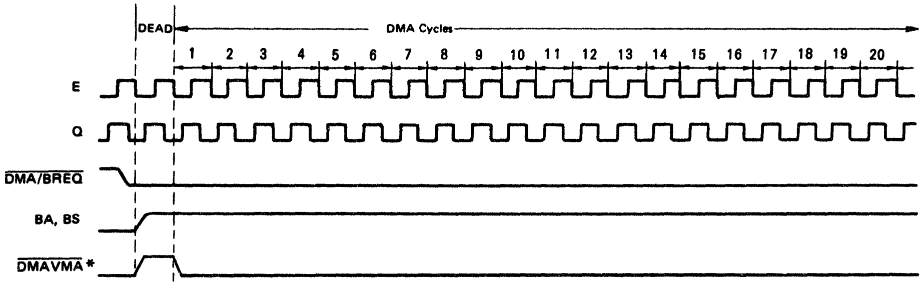
MPU Operation

During normal operation, the MPU fetches an instruction from memory and then executes the requested function. This sequence begins at RES and is repeated indefinitely unless altered by a special instruction or hardware occurrence. Software instructions that alter normal MPU operation are: SWI, SWI2, SWI3, CWAI, RTI and SYNC. An interrupt, HALT or DMA/BREQ can also alter the normal execution of instructions. Figure 14 illustrates the flow chart for the HD6309.



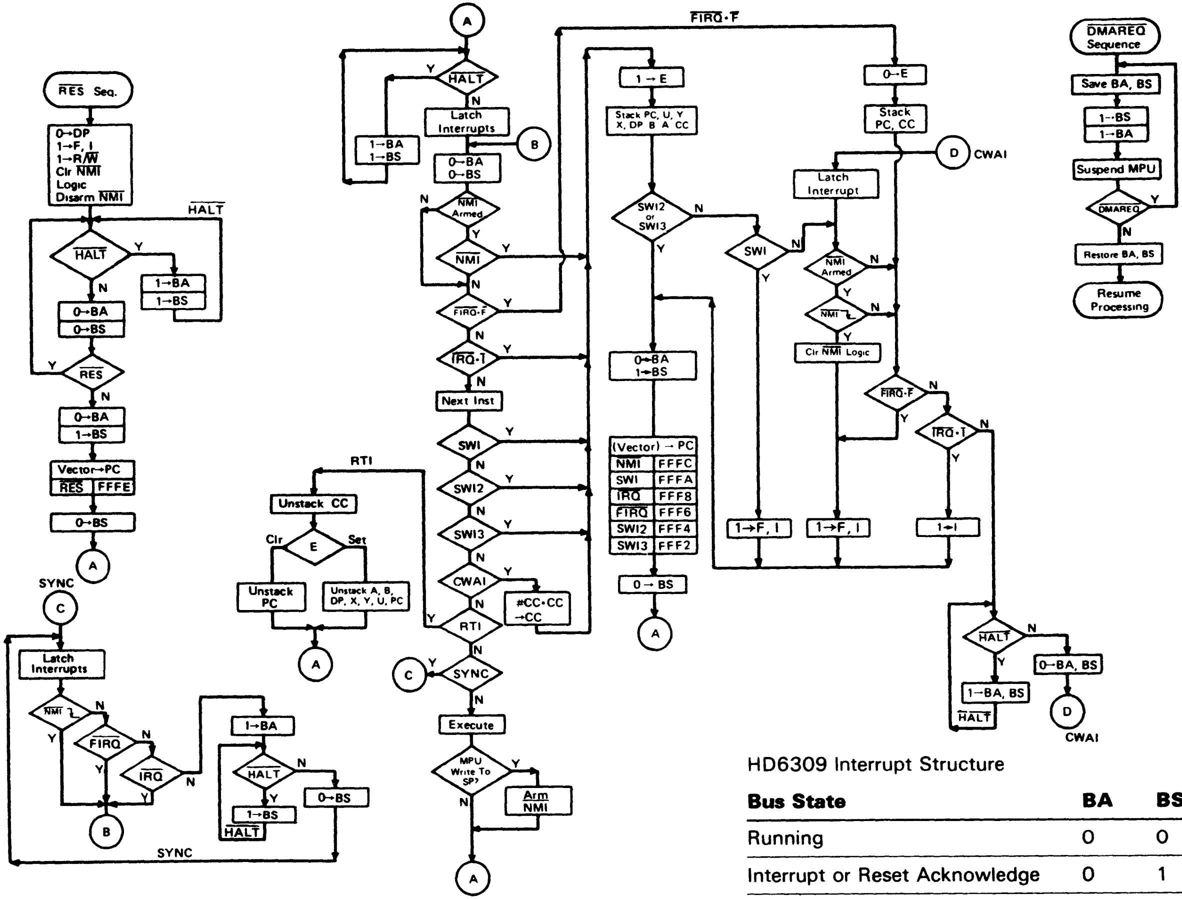
* DMAVMA is developed externally, but it is a system requirement for DMA.

Figure 12. Typical DMA Timing



* DMAVMA is developed externally, but it is a system requirement for DMA.
The HD6309 does not perform the auto-refresh executed in the HD6809.

Figure 13. DMA Timing



Note: Asserting RES will result in entering the reset sequence from any point in the flow chart.

HD6309 Interrupt Structure

Bus State	BA	BS
Running	0	0
Interrupt or Reset Acknowledge	0	1
Sync	1	0
Halt/Bus Grant	1	1

Figure 14. Flowchart for HD6309 Instruction

Addressing Modes

The basic instructions of any computer are greatly enhanced by the presence of powerful addressing modes. The HD6309 has the most complete set of addressing modes available on any microcomputer today. For example, the HD6309 has 59 basic instructions, however, it recognizes 1464 different variations of instructions and addressing modes. The addressing modes support modern programming techniques. The following addressing modes are available on the HD6309:

- Implied (includes accumulator)
- Immediate
- Extended
- Extended indirect
- Direct
- Register
- Indexed
  - Zero-offset
  - Constant offset
  - Accumulator offset
  - Auto increment/decrement
- Indexed indirect
- Relative
- Program counter relative

Implied (Includes Accumulator)

In this addressing mode, the opcode of the instruction contains all the address information necessary. Examples of implied addressing are: ABX, DAA, SWI, ASRA, and CLRB.

Immediate Addressing

In immediate addressing, the effective address of the data is the location immediately following the opcode (i.e., the data to be used in the instruction immediately follows the opcode of the instruction). The HD6309 uses both 8-and 16-bit immediate values depending on the size of the argument specified by the opcode. Examples of instructions with immediate addressing are:

```
LDA #$20
LDX #$F000
LDY #CAT
```

Note: # signifies immediate addressing, $ signifies hexadecimal value.

Extended Addressing

In extended addressing, the contents of the two bytes immediately following the opcode fully specify the 16-bit effective address used by the instruction. Note that the address generated by an extended instruction defines an absolute address and is not position independent. Examples of extended addressing include:

```
LDA   CAT
STX   MOUSE
LDD   $2000
```

Extended Indirect

As a special case of indexed addressing (discussed below), one level of indirection may be added to extended addressing. In extended indirect, the two bytes following the postbyte of an indexed instruction contain the address of the data.

```
LDA   [CAT]
LDX   [$FFFE]
STU   [DOG]
```

Direct Addressing

Direct addressing is similar to extended addressing except that only one byte of address follows the opcode. This byte specifies the lower 8 bits of the address to be used. The upper 8 bits of the address are supplied by the direct page register. Since only one byte of address is required in direct addressing, this mode requires less memory and executes faster than extended addressing. Of course, only 256 locations (one page) can be accessed without redefining the contents of the DP register. Since the DP register is set to $00 on reset, direct addressing on the HD6309 is compatible with direct addressing on the HD6800. Indirection is not allowed in direct addressing. Some examples of direct addressing are:

```
LDA     $30
SETDP   $10 (Assembler directive)
LDB     $1030
LDD     <CAT
```

Note: < is an assembler directive which forces direct addressing.

Register Addressing

Some opcodes are followed by a byte that defines a register or set of registers to be used by the instruction. This is called a postbyte. Some examples of register addressing are:

```
TFR    X,Y          Transfers X into Y
EXG    A, B         Exchanges A with B
PSHS   A, B, X, Y   Push Y, X, B, and A onto S
PULU   X, Y, D      Pull D, X, and Y from U
```

Indexed Addressing

In all indexed addressing, one of the pointer registers (X, Y, U, S, and sometimes PC) is used in a calculation of the effective address of the operand to be used by the instruction. Five basic types of indexing are available and are discussed below. The postbyte of an indexed instruction specifies the basic type and variation of the addressing mode as well as the pointer register to be used. Figure 15 lists the legal formats for the postbyte. Table 4 gives the assembler form and the number of cycles and bytes added to the basic values for indexed addressing for each variation.

Zero-Offset Indexed: In this mode, the selected pointer register contains the effective address of the data to be used by the instruction. This is the fastest indexing mode.

Examples are:

```
LDD     0, X
LDA     S
```

Post-byte Register Bit								Indexed Addressing Mode
7	6	5	4	3	2	1	0	
0	R	R	d	d	d	d	d	EA = ,R +5 Bit Offset
1	R	R	0	0	0	0	0	,R+
1	R	R	0/1	0	0	0	1	,R++
1	R	R	0	0	0	1	0	,−R
1	R	R	0/1	0	0	1	1	,−−R
1	R	R	0/1	0	1	0	0	EA = ,R + 0 Offset
1	R	R	0/1	0	1	0	1	EA = ,R + B Offset
1	R	R	0/1	0	1	1	0	EA = ,R + A Offset
1	R	R	0/1	1	0	0	0	EA = ,R + 8 Bit Offset
1	R	R	0/1	1	0	0	1	EA = ,R + 16 Bit Offset
1	R	R	0/1	1	0	1	1	EA = ,R + D Offset
1	×	×	0/1	1	1	0	0	EA = ,PC + 8 Bit Offset
1	×	×	0/1	1	1	0	1	EA = ,PC + 16 Bit Offset
1	0	0	1	1	1	1	1	EA = [, Address]

Addressing Mode Field

Indirect Field
(Sign bit when b7 = 0)
0········· Non Indirect
1········· Indirect

Register Field : RR
00 = X
01 = Y
10 = U
11 = S

d = Offset Bit
X = Don't Care

Figure 15. Indexed Addressing Postbyte Register Bit Assignments

Constant Offset Indexed: In this mode, a two's-complement offset and the contents of one of the pointer registers are added to form the effective address of the operand. The pointer register's initial content is unchanged by the addition.

Three sizes of offsets are available:

5-bit (−16 to +15)
8-bit (−128 to+127)
16-bit (−32768 to+32767)

The two's complement 5-bit offset is included in the postbyte and, therefore, is most efficient in use of bytes and cycles. The two's complement 8-bit offset is contained in a single byte following the postbyte. The two's complement 16-bit offset is in the two bytes following the postbyte. In most cases the programmer need not be concerned with the size of this offset since the assembler will select the optimal size automatically.

Examples of constant-offset indexing are:

LDA	23, X
LDX	−2, S
LDY	300, X
LDU	CAT, Y

Accumulator Offset Indexed: This mode is similar to constant offset indexed except that the two's-complement value in one of the accumulators (A, B or D) and the contents of one of the pointer registers are added to form

Table 4. Indexed Addressing Mode

		Non Indirect			Indirect		
Type	**Forms**	**Assembler Form**	**Postbyte OP Code**	**+ + ~ #**	**Assembler Form**	**Postbyte OP Code**	**+ + ~ #**
Constant Offset From R (2's Complement Offsets)	No Offset	,R	1RR00100	0 0	[,R]	1RR10100	3 0
	5 Bit Offset	n,R	0RRnnnnn	1 0	defaults to 8-bit		
	8 Bit Offset	n,R	1RR01000	1 1	[n, R]	1RR11000	4 1
	16 Bit Offset	n,R	1RR01001	4 2	[n, R]	1RR11001	7 2
Accumulator Offset From R (2's Complement Offsets)	A Register Offset	A,R	1RR00110	1 0	[A, R]	1RR10110	4 0
	B Register Offset	B,R	1RR00101	1 0	[B, R]	1RR10101	4 0
	D Register Offset	D,R	1RR01011	4 0	[D, R]	1RR11011	7 0
Auto Increment/Decrement R	Increment By 1	,R+	1RR00000	2 0	not allowed		
	Increment By 2	,R++	1RR00001	3 0	[,R ++]	1RR10001	6 0
	Decrement By 1	,−R	1RR00010	2 0	not allowed		
	Decrement By 2	,−−R	1RR00011	3 0	[,−−R]	1RR10011	6 0
Constant Offset From PC (2's Complement Offsets)	8 Bit Offset	n, PCR	1xx01100	1 1	[n, PCR]	1xx11100	4 1
	16 Bit Offset	n, PCR	1xx01101	5 2	[n, PCR]	1xx11101	8 2
Extended Indirect	16 Bit Address				[n]	10011111	5 2

R = X, Y, U or S	RR:
x = Don't Care	00=X
	01=Y
	10=U
	11=S

$\overset{+}{\sim}$and $\overset{+}{\#}$ indicate the number of additional cycles and bytes for the particular variation.

the effective address of the operand. The contents of both the accumulator and the pointer register are unchanged by the addition. The postbyte specifies which accumulator to use as an offset and no additional bytes are required. The advantage of an accumulator offset is that the value of the offset can be calculated by a program at run-time.

Some examples are:

```
LDA     B, Y
LDX     D, Y
LEAX    B, X
```

Auto Increment/Decrement Indexed: In the auto increment addressing mode, the pointer register contains the address of the operand. Then, after the pointer register is used it is incremented by one or two. This addressing mode is useful in stepping through tables, moving data, or for the creation of software stacks. In auto decrement, the pointer register is decremented prior to use as the address of the data. The use of auto decrement is similar to that of auto increment; but the tables, etc, are scanned from high to low addresses. The size of the increment/decrement can be either one or two to allow for tables of either 8-or 16-bit data to be accessed, selectable by the programmer. The pre-decrement, post-increment nature of these modes allow them to be used to create additional software stacks that behave identically to the U and S stacks.

Some examples of the auto increment/decrement addressing modes are:

```
LDA     ,X+
STD     ,Y++
LDB     ,-Y
LDX     ,--S
```

Care should be taken in performing operations on 16-bit pointer registers (X, Y, U, S) where the same register is used to calculate the effective address.

Consider the following instruction:

STX 0, X++ (X initialized to 0)

The desired result is to store a 0 in locations $0000 and $0001 then increment X to point to $0002. In reality, the following occurs:

0→temp	calculate the EA; temp is a holding register
X+2→X	perform autoincrement
X→(temp)	do store operation

Indexed Indirect

All of the indexing modes with the exception of auto increment/decrement by one, or a ±4-bit offset may have an additional level of indirection specified. In indirect addressing, the effective address is contained at the location specified by the contents of the index register plus any offset. In the example below, the A accumulator is loaded indirectly using an effective address calculated from the index register and an offset.

Before Execution:

A=×× (don't care)
X=$F000

$0100	LDA [$10, X]	EA is now $F010
$F010	$F1	$F150 is now the new EA
$F011	$50	
$F150	$AA	

After Execution:

A=$AA (Actual Data Loaded)
X=$F000

All modes of indexed indirect are included except those which are meaningless (e.g., auto increment/decrement by 1 indirect). Some examples of indexed indirect are:

```
LDA     [,X ]
LDD     [10,S]
LDA     [B,Y]
LDD     [,X++]
```

Relative Addressing

The byte(s) following the branch opcode is (are) treated as a signed offset which may be added to the program counter. If the branch condition is true then the calculated address (PC + signed offset) is loaded into the program counter. Program execution continues at the new location as indicated by the PC. Short (1 byte offset) and long (2 bytes offset) relative addressing modes are available. All of memory can be reached in long relative addressing as an effective address is interpreted modulo 2^{16}. Some examples of relative addressing are:

```
        BEQ     CAT       (short)
        BGT     DOG       (short)
CAT     LBEQ    RAT       (long)
DOG     LBGT    RABBIT    (long)
        .
        .
        .
RAT     NOP
RABBIT  NOP
```

Program Counter Relative

The PC can be used as the pointer register with 8 -or 16-bit signed offsets. As in relative addressing, the offset is added to the current PC to create the effective address. The effective address is then used as the address of the operand or data. Program counter relative addressing is used for writing position independent programs. Tables related to a particular routine will maintain the same relationship after the routine is moved, if referenced relative to the program counter. Examples are:

```
LDA     CAT, PCR
LEAX    TABLE, PCR
```

Since program counter relative is a type of indexing, an additional level of indirection is available.

```
LDA     [CAT, PCR]
LDU     [DOG, PCR]
```

HD6309 Instruction Set

The instruction set of the HD6309 is similar to that of the HD6800 and is upward compatible at the source code level. The number of opcodes has been reduced from 72 to 59, but because of the expanded architecture and additional addressing modes, the number of available opcodes (with different addressing modes) has risen from 197 to 1464.

Some of the instructions and addressing modes are described in detail below:

PSHU/PSHS

The push instructions can push onto either the hardware stack (S) or user stack (U) any single register, or set of registers with a single instruction.

PULU/PULS

The pull instructions have the same capability of the push instruction, in reverse order. The byte immediately following the push or pull opcode determines which register or registers are to be pushed or pulled. The actual PUSH/PULL sequence is fixed: each bit defines a unique register to push or pull, as shown in figure 16.

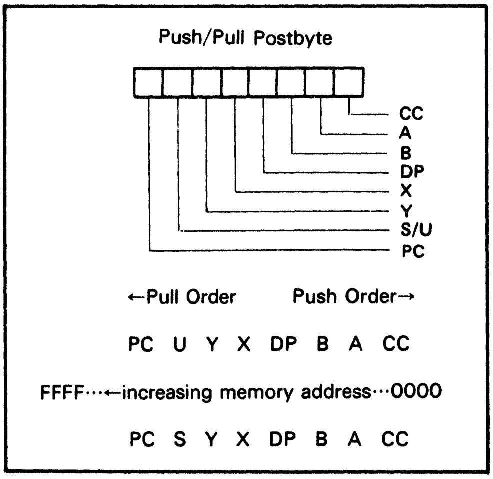


Figure 16. Push and Pull Order

TFR/EXG

Within the HD6309, any register may be transferred to or exchanged with another of like-size: i. e., 8-bit to 8-bit or 16-bit to 16-bit. Bits 4-7 of the postbyte define the source register, while bits 0-3 represent the destination register (figure 17). They are denoted as follows:

0000—D
0001—X
0010—Y
0011—U
0100—S
0101—PC
1000—A
1001—B
1010—CC
1011—DP

Note: All other combinations are undefined and invalid.

LEAX/LEAY/LEAU/LEAS

The LEA (load effective address) works by calculating the effective address used in an indexed instruction and stores that address value, rather than the data at that address, in a pointer register. This makes all the features of the internal addressing hardware available to the programmer. Some of the implications of this instruction are illustrated in table 5.

The LEA instruction also allows the user to access data in a position independent manner. For example:

```
        LEAX  MSG1, PCR
        LBSR  PDATA(Print message routine)

    MSG1 FCC  'MESSAGE'
```

This sample program prints: 'MESSAGE'. By writing MSG1, PCR, the assembler computes the distance between the present address and MSG1. This result is placed as a constant into the LEAX instruction which will be indexed from the PC value at the time of execution. No matter where the code is located, when it is executed, the computed offset from the PC will put the absolute address of MSG1 into the X pointer register. This code is totally position independent.

The LEA instructions are very powerful and use an internal holding register (temp). Care must be exercised when using the LEA instructions with the autôincrement and autodecrement addressing modes due to the sequence of internal operations. The LEA internal sequence is outlined as follows:

LEAa ,b+	(any of the 16-bit pointer registers X, Y, U, or S may be substituted for a and b)
1. b→temp	(calculate the EA)
2. b + 1→b	(modify b, postincrement)
3. temp →a	(load a)
LEAa ,—b	
1. b —1 →temp	(calculate EA with predecrement)
2. b —1 →b	(modify b, predecrement)
3. temp →a	(load a)

Autoincrement-by-two and autodecrement-by-two instructions work similarly. Note that LEAX, X+ does not change X, however LEAX,—X does decrement X. LEAX 1, X should be used to increment X by one.

MUL

Multiplies the unsigned binary numbers in the A

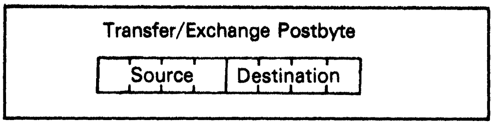


Figure 17. TFR/EXG Format

Table 5. LEA Examples

Instruction	Operation	Comment
LEAX 10, X	X+10→X	Adds 5-bit constant 10 to X
LEAX 500, X	X+500→X	Adds 16-bit constant 500 to X
LEAY A, Y	Y+A→Y	Adds 8-bit A accumulator to Y
LEAY D, Y	Y+D→Y	Adds 16-bit D accumulator to Y
LEAU-10, U	U-10→U	Subtracts 10 from U
LEAS-10, S	S-10→S	Used to reserve area on stack
LEAS 10, S	S+10→S	Used to 'clean up' stack
LEAX 5, S	S+5→X	Transfers as well as adds

and B accumulator and places the unsigned result into the 16-bit D accumulator. This unsigned multiply also allows multiple-precision multiplications.

Long And Short Relative Branches

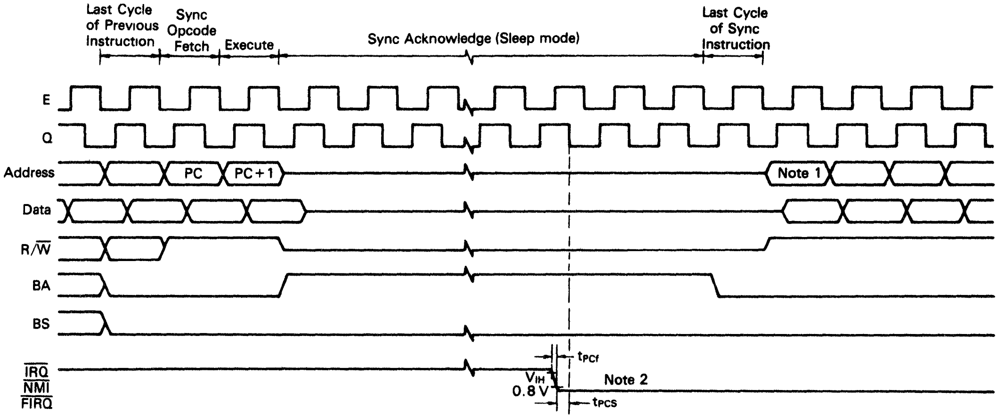
The HD6309 has the capability of program counter relative branching throughout the entire memory map. In this mode, if the branch is to be taken, the 8-or 16-bit signed offset is added to the value of the program counter to be used as the effective address. This allows the program to branch anywhere in the 64k memory map. Position independent code can be easily generated through the use of relative branching. Both short (8-bit) and long (16-bit) branches are available.

SYNC

After encountering a sync instruction, the MPU enters a sync state, stops processing instructions, and waits for an interrupt. If the pending interrupt is non-maskable ($\overline{\text{NMI}}$) or maskable ($\overline{\text{FIRQ}}$, $\overline{\text{IRQ}}$) with its mask bit (F or I) clear, the processor will clear the sync state and perform the normal interrupt stacking and service routine. Since $\overline{\text{FIRQ}}$ and $\overline{\text{IRQ}}$ are not edge-triggered, a low level with a minimum duration of three bus cycles is required to assure that the interrupt will be taken. If the pending interrupt is maskable ($\overline{\text{FIRQ}}$, $\overline{\text{IRQ}}$) with its mask bit (F or I) set, the processor will clear the sync state and continue processing by executing the next inline instruction. Figure 18 depicts sync timing.

Software Interrupt

A software interrupt instruction will cause an interrupt, and its associated vector fetch. These software interrupts are useful in operating system calls, software debugging, trace operations, memory mapping, and software development systems. Three levels of SWI are available on this HD6309, and are prioritized in the following order: SWI, SWI2, SWI3.


Last Cycle of Previous Instruction
Sync Opcode Fetch
Execute
Sync Acknowledge (Sleep mode)
Last Cycle of Sync Instruction
E
Q
Address
PC
PC+1
Note 1
Data
R/W
BA
BS
IRQ
NMI
FIRQ
tPCf
VIH
0.8V
Note 2
tPCS

Notes: 1 If the associated mask bit is set when the interrupt is requested, this cycle will be an instruction fetch from address location PC + 1.However if the interrupt is accepted ($\overline{\text{NMI}}$ or an unmasked $\overline{\text{FIRQ}}$ or $\overline{\text{IRQ}}$) interrupt processing continues with this cycle as (m) on figures 6 and 7 (interrupt timing).

2.If mask bits are clear, $\overline{\text{IRQ}}$ and $\overline{\text{FIRQ}}$ must be held low for three cycles to guarantee that interrupt will be taken, although only one cycle is necessary to bring the processor out of SYNC.

Figure 18. Sync Timing

16-Bit Operation

The HD6309 has the capability of processing 16-bit data. These instructions include loads, stores, compares, adds, subtracts, transfers, exchanges, pushes and pulls.

Cycle-by-Cycle Operation

The address bus cycle-by-cycle performance chart illustrates the memory-access sequence corresponding to each possible instruction and address ing mode in the HD6309. Each instruction begins with an opcode fetch. While that opcode is being internally decoded, the next program byte is always fetched. (Most instructions will use the next byte, so this technique considerably speeds throughput.) Next, the operation of each opcode will follow the flow chart.$\overline{VMA}$ is an indication of $FFFF_{16}$ on the address bus, R/$\overline{W}$ = high and BS = low. The following examples illustrate the use of the chart : see figure 19.

Example 1: LBSR (Branch Taken)

Before Execution SP = F000

```
                        .
                        .
                        .
$8000                   LBSR    CAT
                        .
                        .
                        .
$A000          CAT      .
```

Cycle-by-Cycle Flow

Cycle #	Address	Data	R/$\overline{W}$	Description
1	8000	17	1	Opcode Fetch
2	8001	1F	1	Offset High Byte
3	8002	FD	1	Offset Low Byte
4	FFFF	*	1	$\overline{VMA}$ Cycle
5	FFFF	*	1	$\overline{VMA}$ Cycle
6	FFFF	*	1	$\overline{VMA}$ Cycle
7	FFFF	*	1	$\overline{VMA}$ Cycle
8	EFFF	03	0	Stack Low Order Byte of Return Address
9	EFFE	80	0	Stack High Order Byte of Return Address

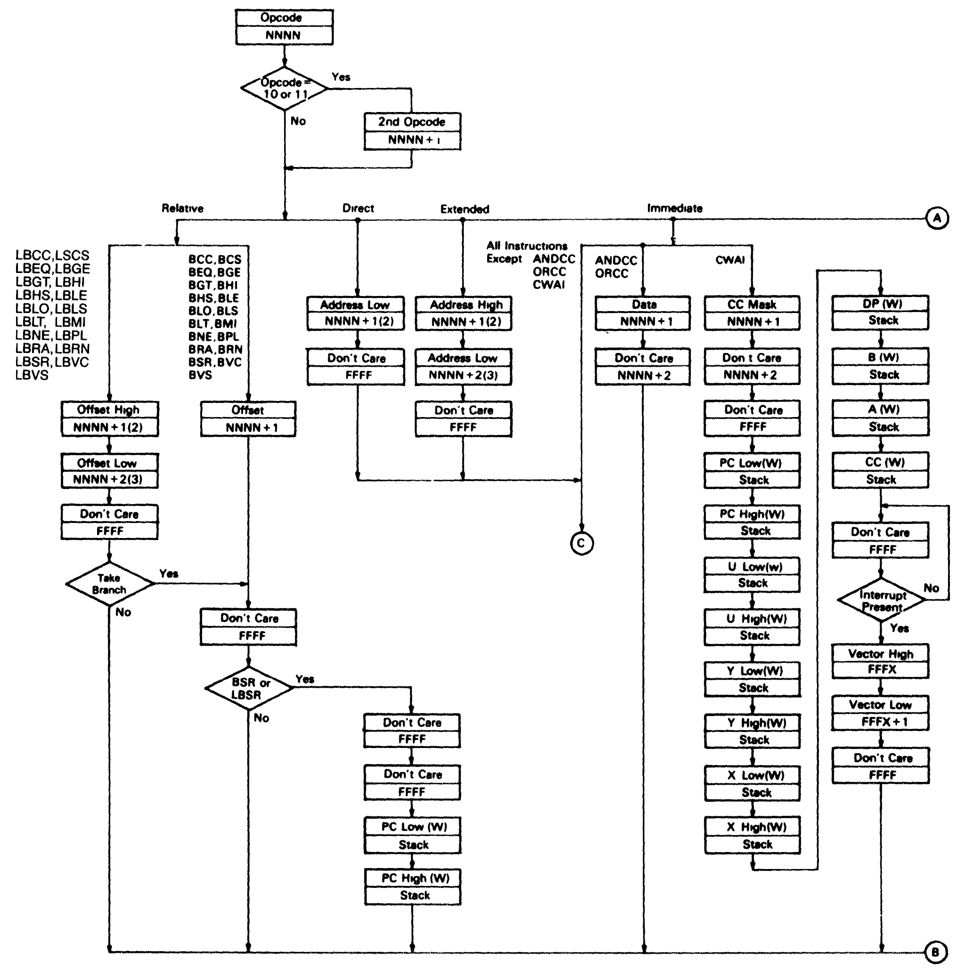
Example 2: DEC (Extended)

```
$8000          DEC          $A000
$A000          FCB          $80
```

Cycle-by-Cycle Flow

Cycle #	Address	Data	R/$\overline{W}$	Description
1	8000	7A	1	Opcode Fetch
2	8001	A0	1	Operand Address, High Byte
3	8002	00	1	Operand Address, Low Byte
4	FFFF	*	1	$\overline{VMA}$ Cycle
5	A000	80	1	Read the Data
6	FFFF	*	1	$\overline{VMA}$ Cycle
7	A000	7F	0	Store the Decremented Data

* The data bus has the data at that particular address.



Notes: 1. Each state shows
Data Bus | Offset High |
Address Bus | NNNN + 1(2) |
2. Address NNNN is location of opcode.
3. If opcode is two byte opcode subsequent addresses are in parenthesis ( ).
4. Two-byte opcodes are highlighted.

Figure 19. Cycle-by-Cycle Performance

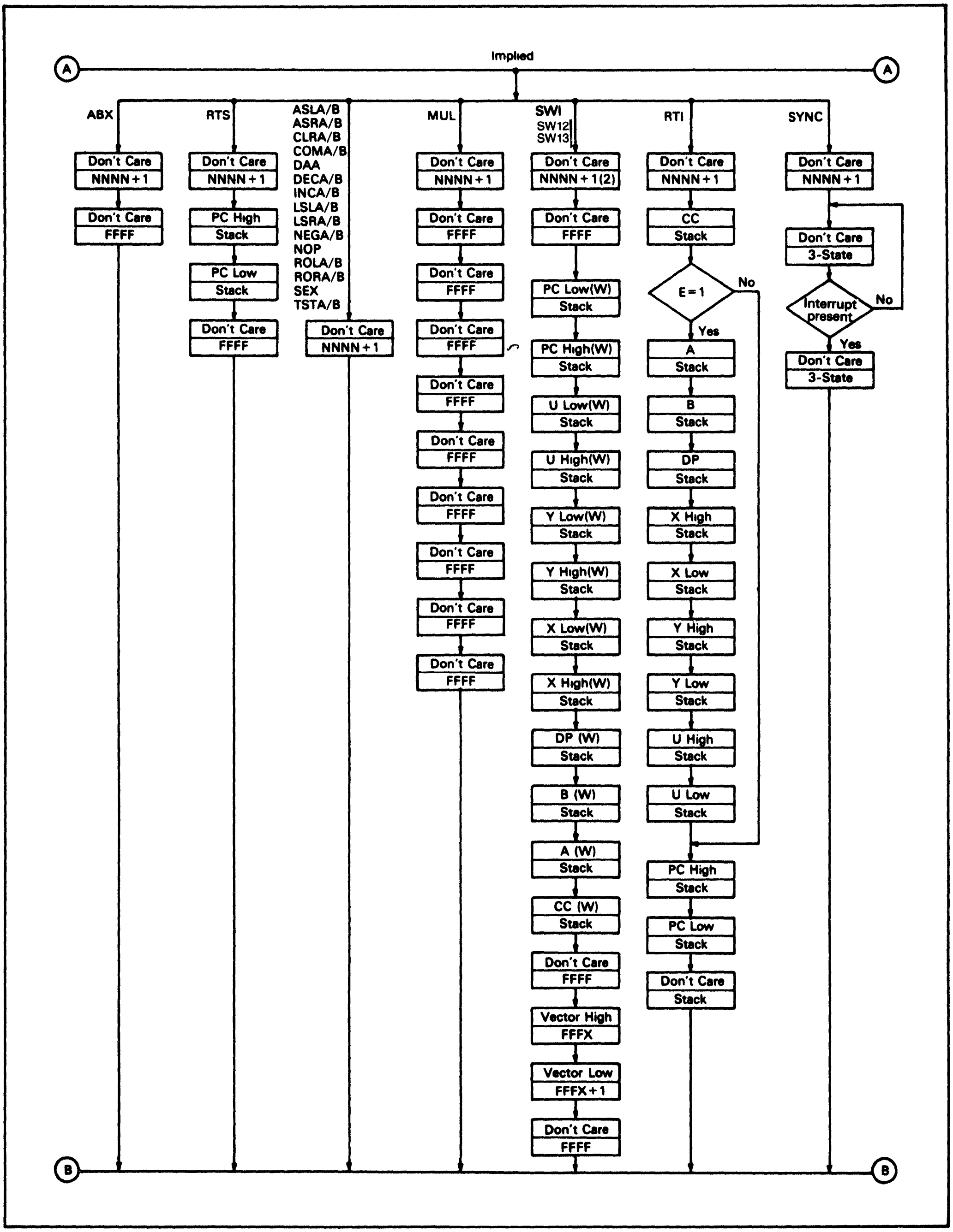


Figure 19. Cycle-by-Cycle Performance (Cont.)

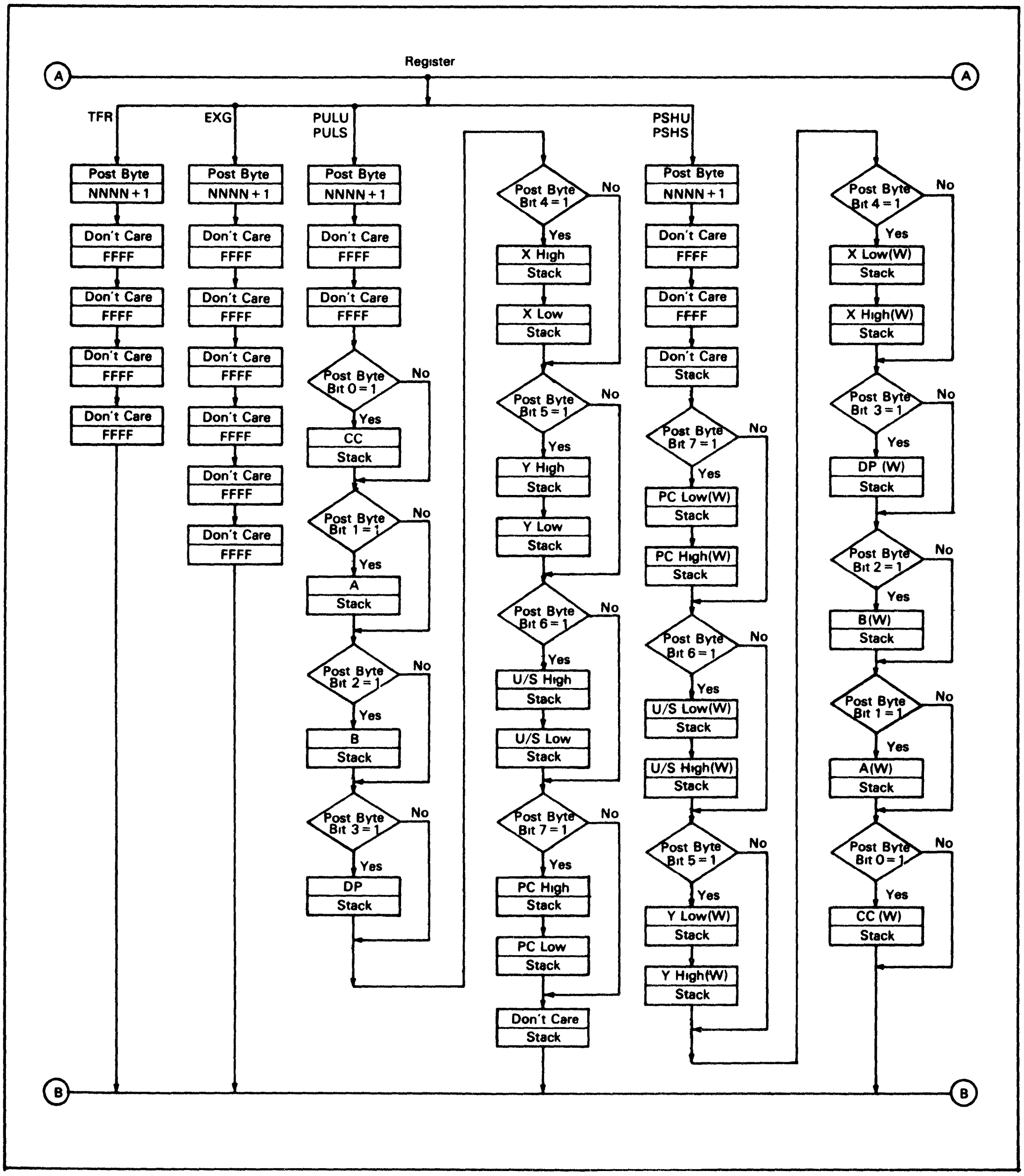
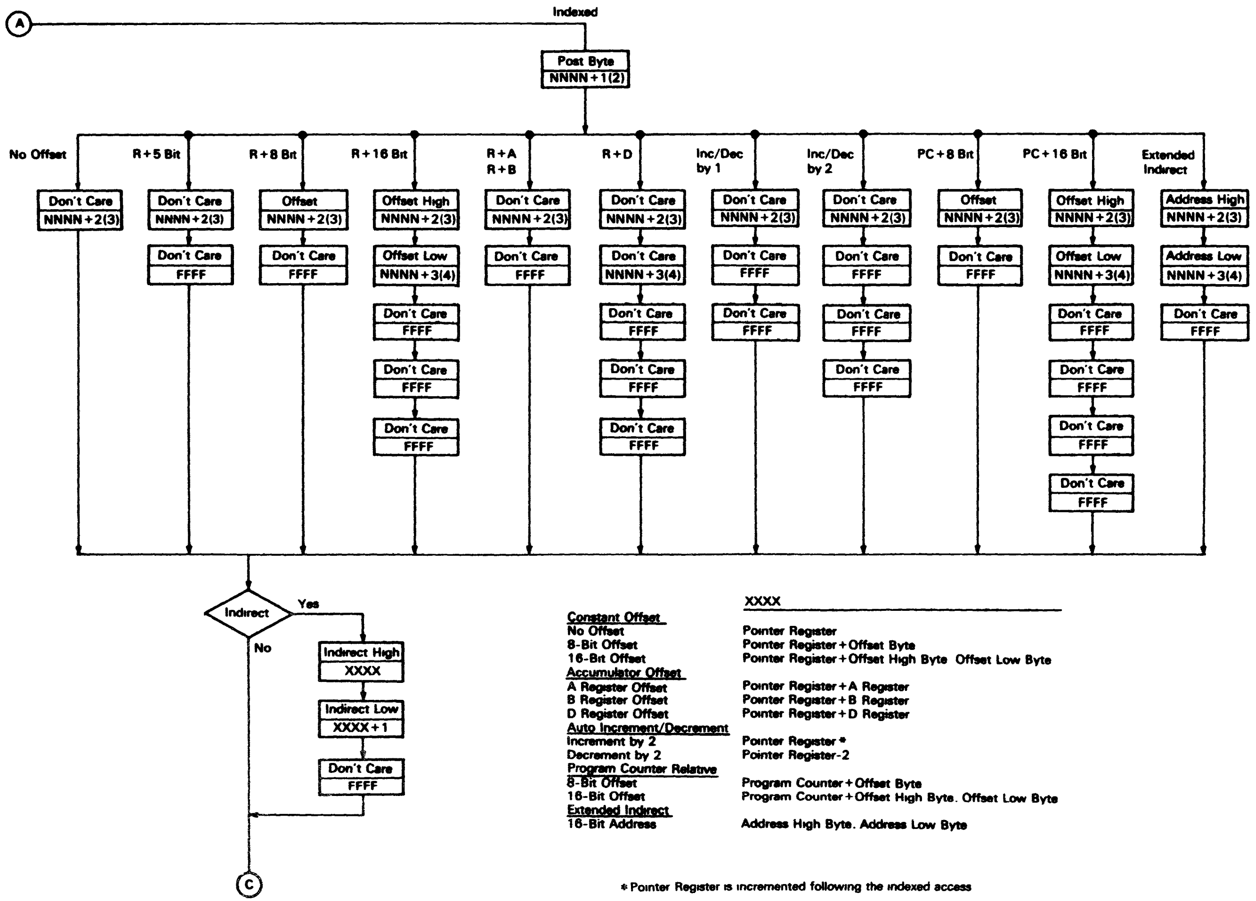


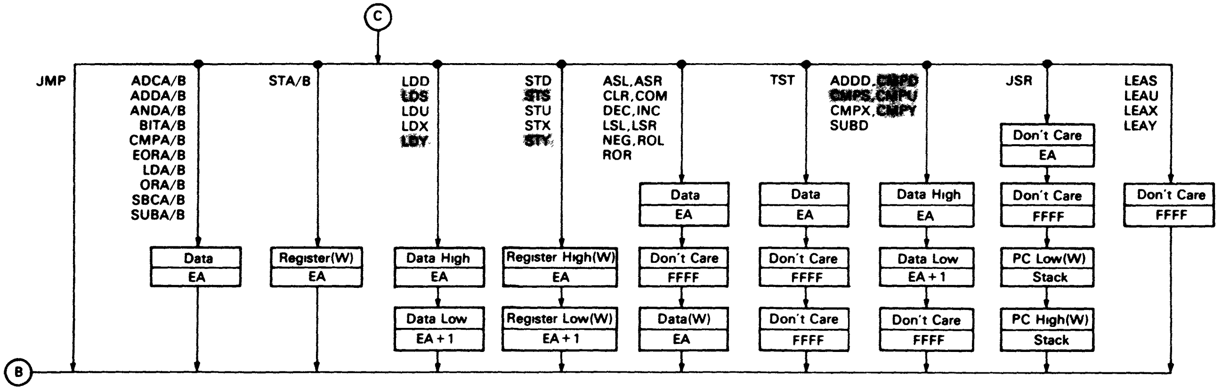
Figure 19. Cycle-by-Cycle Performance (Cont.)



	XXXX
Constant Offset	
No Offset	Pointer Register
8-Bit Offset	Pointer Register+Offset Byte
16-Bit Offset	Pointer Register+Offset High Byte Offset Low Byte
Accumulator Offset	
A Register Offset	Pointer Register+A Register
B Register Offset	Pointer Register+B Register
D Register Offset	Pointer Register+D Register
Auto Increment/Decrement	
Increment by 2	Pointer Register *
Decrement by 2	Pointer Register-2
Program Counter Relative	
8-Bit Offset	Program Counter+Offset Byte
16-Bit Offset	Program Counter+Offset High Byte. Offset Low Byte
Extended Indirect	
16-Bit Address	Address High Byte. Address Low Byte

* Pointer Register is incremented following the indexed access

Figure 19. Cycle-by-Cycle Performance (Cont.)



Constant Offset	Effective Address (EA)
No Offset	Pointer Register
5-Bit Offset	Pointer Register + Post Byte
8-Bit Offset	Pointer Register + Offset Byte
16-Bit Offset	Pointer Register + Offset High Byte Offset Low Byte
Accumulator Offset	
A Register Offset	Pointer Register + A Register
B Register Offset	Pointer Register + B Register
D Register Offset	Pointer Register + D Register
Auto Increment/Decrement	
Increment by 1	Pointer Register*
Increment by 2	Pointer Register*
Decrement by 1	Pointer Register − 1
Decrement by 2	Pointer Register − 2
Program Counter Relative	
8-Bit Offset	Program Counter + Offset Byte
16-Bit Offset	Program Counter + Offset High Byte Offset Low Byte
Indirect	Indirect High Byte Indirect Low Byte
Direct	Direct Page Register Address Low Byte
Extended	Address High Byte Address Low Byte
Immediate	NNNN + 1(2)

*Pointer Register is incremented following the indexed access

Figure 19. Cycle-by-Cycle Performance (Cont.)

Sleep Mode

During the interrupt wait period in the SYNC instruction (the sync state) and in the CWAI instruction (the wait state), MPU operation is halted and goes to the sleep mode. However, the state of I/O pins is the same as that of the HD6809 in this mode.

HD6309 Instruction set Tables

The instructions of the HD6309 have been broken down into five different categories. They are as follows:

- 8-Bit operation (table 6)
- 16-Bit operation (table 7)
- Index register/stack pointer instructions (table 8)
- Relative branches (long or short) (table 9)
- Miscellaneous instructions (table 10)

HD6309 instruction set tables and Hexadecimal Values of instructions are shown in table 11 and table 12.

Table 6. 8-Bit'Accumulator and Memory Instructions

Mnemonic(s)	Operation
ADCA, ADCB	Add memory to accumulator with carry
ADDA, ADDB	Add memory to accumulator
ANDA, ANDB	AND memory with accumulator
ASL, ASLA, ASLB	Arithmetic shift of accumulator or memory left
ASR, ASRA, ASRB	Arithmetic shift of accumulator or memory right
BITA, BITB	Bit test memory with accumulator
CLR, CLRA, CLRB	Clear accumulator or memory location
CMPA, CMPB	Compare memory from accumulator
COM, COMA, COMB	Complement accumulator or memory location
DAA	Decimal adjust A accumulator
DEC, DECA, DECB	Decrement accumulator or memory location
EORA, EORB	Exclusive OR memory with accumulator
EXG R1, R2	Exchange R1 with R2 (R1, R2=A, B, CC, DP)
INC, INCA, INCB	Increment accumulator or memory location
LDA, LDB	Load accumulator from memory
LSL, LSLA, LSLB	Logical shift left accumulator or memory location
LSR, LSRA, LSRB	Logical shift right accumulator or memory location
MUL	Unsigned multiply (A×B→D)
NEG, NEGA, NEGB	Negate accumulator or memory
ORA, ORB	OR memory with accumulator
ROL, ROLA, ROLB	Rotate accumulator or memory left
ROR, RORA, RORB	Rotate accumulator or memory right
SBCA, SBCB	Subtract memory from accumulator with borrow
STA, STB	Store accumulator to memory
SUBA, SUBB	Subtract memory from accumulator
TST, TSTA, TSTB	Test accumulator or memory location
TFR R1, R2	Transfer R1 or R2 (R1, R2=A, B, CC, DP)

Note: A, B, CC or DP may be pushed to (pulled from) either stack with PSHS, PSHU (PULS, PULU) instructions.

Table 7. 16-Bit Accumulator and Memory Instructions

Mnemonic(s)	Operation
ADDD	Add memory to D accumulator
CMPD	Compare memory from D accumulator
EXG D, R	Exchange D with X, Y, S, U or PC
LDD	Load D accumulator from memory
SEX	Sign Extend B accumulator into A accumulator
STD	Store D accumulator to memory
SUBD	Subtract memory from D accumulator
TFR D, R	Transfer D to X, Y, S, U or PC
TFR R, D	Transfer X, Y, S, U or PC to D

Note: D may be pushed (pulled) to either stack with PSHS, PSHU (PULS, PULU) instructions.

Table 8. Index Register/Stack Pointer Instructions

Mnemonic(s)	Operation
CMPS, CMPU	Compare memory from stack pointer
CMPX, CMPY	Compare memory from index register
EXG R1, R2	Exchange D, X, Y, S, U or PC with D, X, Y, S, U or PC
LEAS, LEAU	Load effective address into stack pointer
LEAX, LEAY	Load effective address into index register
LDS, LDU	Load stack pointer from memory
LDX, LDY	Load index register from memory
PSHS	Push A, B, CC, DP, D, X, Y, U or PC onto hardware stack
PSHU	Push A, B, CC, DP, D, X, Y, S or PC onto user stack
PULS	Pull A, B, CC, DP, D, X, Y, U or PC from hardware stack
PULU	Pull A, B, CC, DP, D, X, Y, S or PC from user stack
STS, STU	Store stack pointer to memory
STX, STY	Store index register to memory
TFR R1, R2	Transfer D, X, Y, S, U or PC to D, X, Y, S, U or PC
ABX	Add B accumulator to X (unsigned)

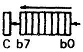
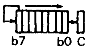
Table 9. Branch Instructions

Mnemonic(s)	Operation
	Simple Branches
BEQ, LBEQ	Branch if equal
BNE, LBNE	Branch if not equal
BMI, LBMI	Branch if minus
BPL, LBPL	Branch if plus
BCS, LBCS	Branch if carry set
BCC, LBCC	Branch if carry clear
BVS, LBVS	Branch if overflow set
BVC, LBVC	Branch if overflow clear
	Signed Branches
BGT, LBGT	Branch if greater (signed)
BGE, LBGE	Branch if greater than or equal (signed)
BEQ, LBEQ	Branch if equal
BLE, LBLE	Branch if less than or equal (signed)
BLT, LBLT	Branch if less than (signed)
	Unsigned Branches
BHI, LBHI	Branch if higher (unsigned)
BHS, LBHS	Branch if higher or same (unsigned)
BEQ, LBEQ	Branch if equal
BLS, LBLS	Branch if lower or same (unsigned)
BLO, LBLO	Branch if lower (unsigned)
	Other Branches
BSR, LBSR	Branch to subroutine
BRA, LBRA	Branch always
BRN, LBRN	Branch never

Table 10. Miscellaneous Instructions

Mnemonic(s)	Operation
ANDCC	AND condition code register
CWAI	AND condition code register, then wait for interrupt
NOP	No operation
ORCC	OR condition code register
JMP	Jump
JSR	Jump to subroutine
RTI	Return from interrupt
RTS	Return from subroutine
SWI, SWI2, SWI3	Software interrupt (absolute indirect)
SYNC	Synchronize with interrupt line

Table 11. HD6309 Instruction Set Table

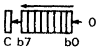
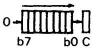
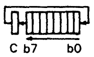
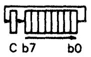
INSTRUCTIONS/ FORMS		IMP ACCM REG			DIRECT			EXTND			IMMED			INDEX①			RELATIVE			DESCRIPTION	7	6	5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~⑤	#		E	F	H	I	N	Z	V	C
ABX		3A	3	1																B+X→X (Unsigned)	●	●	●	●	●	●	●	●
ADC	ADCA				99	4	2	B9	5	3	89	2	2	A9	4+	2+				A+M+C→A	●	●	↕	●	↕	↕	↕	↕
	ADCB				D9	4	2	F9	5	3	C9	2	2	E9	4+	2+				B+M+C→B	●	●	↕	●	↕	↕	↕	↕
ADD	ADDA				9B	4	2	BB	5	3	8B	2	2	AB	4+	2+				A+M→A	●	●	↕	●	↕	↕	↕	↕
	ADDB				DB	4	2	FB	5	3	CB	2	2	EB	4+	2+				B+M→B	●	●	↕	●	↕	↕	↕	↕
	ADDD				D3	6	2	F3	7	3	C3	4	3	E3	6+	2+				D+M . M+1→D	●	●	●	●	↕	↕	↕	↕
AND	ANDA				94	4	2	B4	5	3	84	2	2	A4	4+	2+				A∧M→A	●	●	●	●	↕	↕	R	●
	ANDB				D4	4	2	F4	5	3	C4	2	2	E4	4+	2+				B∧M→B	●	●	●	●	↕	↕	R	●
	ANDCC										1C	3	2							CC∧IMM→CC	(			⑦				)
ASL	ASLA	48	2	1																A	●	●	⑧	●	↕	↕	↕	↕
	ASLB	58	2	1																B 	●	●	⑧	●	↕	↕	↕	↕
	ASL				08	6	2	78	7	3				68	6+	2+				M	●	●	⑧	●	↕	↕	↕	↕
ASR	ASRA	47	2	1																A	●	●	⑧	●	↕	↕	●	↕
	ASRB	57	2	1																B 	●	●	⑧	●	↕	↕	●	↕
	ASR				07	6	2	77	7	3				67	6+	2+				M	●	●	⑧	●	↕	↕	●	↕
BCC	BCC																24	3	2	Branch C=0	●	●	●	●	●	●	●	●
	LBCC																10 24	5(6)	4	Long Branch C=0	●	●	●	●	●	●	●	●
BCS	BCS																25	3	2	Branch C=1	●	●	●	●	●	●	●	●
	LBCS																10 25	5(6)	4	Long Branch C=1	●	●	●	●	●	●	●	●
BEQ	BEQ																27	3	2	Branch Z=1	●	●	●	●	●	●	●	●
	LBEQ																10 27	5(6)	4	Long Branch Z=1	●	●	●	●	●	●	●	●
BGE	BGE																2C	3	2	Branch N⊕V=0	●	●	●	●	●	●	●	●
	LBGE																10 2C	5(6)	4	Long Branch N⊕V=0	●	●	●	●	●	●	●	●
BGT	BGT																2E	3	2	Branch Z∨(N⊕V)=0	●	●	●	●	●	●	●	●
	LBGT																10 2E	5(6)	4	Long Branch Z∨(N⊕V)=0	●	●	●	●	●	●	●	●
BHI	BHI																22	3	2	Branch C∨Z=0	●	●	●	●	●	●	●	●
	LBHI																10 22	5(6)	4	Long Branch C∨Z=0	●	●	●	●	●	●	●	●
BHS	BHS																24	3	2	Branch C=0	●	●	●	●	●	●	●	●
	LBHS																10 24	5(6)	4	Long Branch C=0	●	●	●	●	●	●	●	●
BIT	BITA				95	4	2	B5	5	3	85	2	2	A5	4+	2+				Bit Test A (M∧A)	●	●	●	●	↕	↕	R	●
	BITB				D5	4	2	F5	5	3	C5	2	2	E5	4+	2+				Bit Test B (M∧B)	●	●	●	●	↕	↕	R	●
BLE	BLE																2F	3	2	Branch Z∨(N⊕V)=1	●	●	●	●	●	●	●	●
	LBLE																10 2F	5(6)	4	Long Branch Z∨(N⊕V)=1	●	●	●	●	●	●	●	●
BLO	BLO																25	3	2	Branch C=1	●	●	●	●	●	●	●	●
	LBLO																10 25	5(6)	4	Long Branch C=1	●	●	●	●	●	●	●	●
BLS	BLS																23	3	2	Branch C∨Z=1	●	●	●	●	●	●	●	●
	LBLS																10 23	5(6)	4	Long Branch C∨Z=1	●	●	●	●	●	●	●	●
BLT	BLT																2D	3	2	Branch N⊕V=1	●	●	●	●	●	●	●	●
	LBLT																10 2D	5(6)	4	Long Branch N⊕V=1	●	●	●	●	●	●	●	●
BMI	BMI																2B	3	2	Branch N=1	●	●	●	●	●	●	●	●
	LBMI																10 2B	5(6)	4	Long Branch N=1	●	●	●	●	●	●	●	●

(Continued)

INSTRUCTIONS/ FORMS		IMP ACCM REG			DIRECT			EXTND			IMMED			INDEX①			RELATIVE			DESCRIPTION	7	6	5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~⑤	#		E	F	H	I	N	Z	V	C
BNE	BNE																26	3	2	Branch Z=0	●	●	●	●	●	●	●	●
	LBNE																10	5(6)	4	Long Branch	●	●	●	●	●	●	●	●
																	26			Z=0								
BPL	BPL																2A	3	2	Branch N=0	●	●	●	●	●	●	●	●
	LBPL																10	5(6)	4	Long Branch	●	●	●	●	●	●	●	●
																	2A			N=0								
BRA	BRA																20	3	2	Branch Always	●	●	●	●	●	●	●	●
	LBRA																16	5	3	Long Branch Always	●	●	●	●	●	●	●	●
BRN	BRN																21	3	2	Branch Never	●	●	●	●	●	●	●	●
	LBRN																10	5	4	Long Branch Never	●	●	●	●	●	●	●	●
																	21											
BSR	BSR																8D	7	2	Branch to	●	●	●	●	●	●	●	●
																				Subroutine								
	LBSR																17	9	3	Long Branch to	●	●	●	●	●	●	●	●
																				Subroutine								
BVC	BVC																28	3	2	Branch V=0	●	●	●	●	●	●	●	●
	LBVC																10	5(6)	4	Long Branch	●	●	●	●	●	●	●	●
																	28			V=0								
BVS	BVS																29	3	2	Branch V=1	●	●	●	●	●	●	●	●
	LBVS																10	5(6)	4	Long Branch	●	●	●	●	●	●	●	●
																	29			V=1								
CLR	CLRA	4F	2	1																0→A	●	●	●	●	R	S	R	R
	CLRB	5F	2	1																0→B	●	●	●	●	R	S	R	R
	CLR				0F	6	2	7F	7	3				6F	6+	2+				0→M	●	●	●	●	R	S	R	R
CMP	CMPA				91	4	2	B1	5	3	81	2	2	A1	4+	2+				Compare M from A	●	●	⑧	●	↕	↕	↕	↕
	CMPB				D1	4	2	F1	5	3	C1	2	2	E1	4+	2+				Compare M from B	●	●	⑧	●	↕	↕	↕	↕
	CMFD				10	7	3	10	8	4	10	5	4	10	7+	3+				Compare M : M+1	●	●	●	●	↕	↕	↕	↕
					93			B3			83			A3						from D								
	CMPS				11	7	3	11	8	4	11	5	4	11	7+	3+				Compare M : M+1	●	●	●	●	↕	↕	↕	↕
					9C			BC			8C			AC						from S								
	CMPU				11	7	3	11	8	4	11	5	4	11	7+	3+				Compare M : M+1	●	●	●	●	↕	↕	↕	↕
					93			B3			83			A3						from U								
	CMPX				9C	6	2	BC	7	3	8C	4	3	AC	6+	2+				Compare M : M+1	●	●	●	●	↕	↕	↕	↕
																				from X								
	CMPY				10	7	3	10	8	4	10	5	4	10	7+	3+				Compare M : M+1	●	●	●	●	↕	↕	↕	↕
					9C			BC			8C			AC						from Y								
COM	COMA	43	2	1																$\overline{A}$→A	●	●	●	●	↕	↕	R	S
	COMB	53	2	1																$\overline{B}$→B	●	●	●	●	↕	↕	R	S
	COM				03	6	2	73	7	3				63	6+	2+				$\overline{M}$→M	●	●	●	●	↕	↕	R	S
CWAI											3C	≧20	2							CC∧IMM→CC :	S	(			⑦			)
																				Wait for Interrupt								
DAA		19	2	1																Decimal Adjust A	●	●	●	●	↕	↕	⑧	↕
DEC	DECA	4A	2	1																A−1→A	●	●	●	●	↕	↕	↕	●
	DECB	5A	2	1																B−1→B	●	●	●	●	↕	↕	↕	●
	DEC				0A	6	2	7A	7	3				6A	6+	2+				M−1→M	●	●	●	●	↕	↕	↕	●
EOR	EORA				98	4	2	B8	5	3	88	2	2	A8	4+	2+				A⊕M→A	●	●	●	●	↕	↕	R	●
	EORB				D8	4	2	F8	5	3	C8	2	2	E8	4+	2+				B⊕M→B	●	●	●	●	↕	↕	R	●
EXG	R1, R2	1E	8	2																R1↔R2②	(			⑩				)
INC	INCA	4C	2	1																A+1→A	●	●	●	●	↕	↕	↕	●
	INCB	5C	2	1																B+1→B	●	●	●	●	↕	↕	↕	●
	INC				0C	6	2	7C	7	3				6C	6+	2+				M+1→M	●	●	●	●	↕	↕	↕	●
JMP					0E	3	2	7E	4	3				6E	3+	2+				EA③→PC	●	●	●	●	●	●	●	●
JSR					9D	7	2	BD	8	3				AD	7+	2+				Jump to Subroutine	●	●	●	●	●	●	●	●

(Continued)

2

INSTRUCTIONS/ FORMS		IMP ACCM REG			DIRECT			EXTND			IMMED			INDEX①			RELATIVE			DESCRIPTION	7	6	5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~⑤	#		E	F	H	I	N	Z	V	C
LD	LDA				96	4	2	B6	5	3	86	2	2	A6	4+	2+				M→A	●	●	●	●	↕	↕	R	●
	LDB				D6	4	2	F6	5	3	C6	2	2	E6	4+	2+				M→B	●	●	●	●	↕	↕	R	●
	LDD				DC	5	2	FC	6	3	CC	3	3	EC	5+	2+				M M+1→D	●	●	●	●	↕	↕	R	●
	LDS				10	6	3	10	7	4	10	4	4	10	6+	3+				M : M+1→S	●	●	●	●	↕	↕	R	●
					DE			FE			CE			EE														
	LDU				DE	5	2	FE	6	3	CE	3	3	EE	5+	2+				M . M+1→U	●	●	●	●	↕	↕	R	●
	LDX				9E	5	2	BE	6	3	8E	3	3	AE	5+	2+				M . M+1→X	●	●	●	●	↕	↕	R	●
	LDY				10	6	3	10	7	4	10	4	4	10	6+	3+				M . M+1→Y	●	●	●	●	↕	↕	R	●
					9E			BE			8E			AE														
LEA	LEAS													32	4+	2+				EA③→S	●	●	●	●	●	●	●	●
	LEAU													33	4+	2+				EA③→U	●	●	●	●	●	●	●	●
	LEAX													30	4+	2+				EA③→X	●	●	●	●	●	↕	●	●
	LEAY													31	4+	2+				EA③→Y	●	●	●	●	●	↕	●	●
LSL	LSLA	48	2	1																A	●	●	●	●	↕	↕	↕	↕
	LSLB	58	2	1																B 	●	●	●	●	↕	↕	↕	↕
	LSL				08	6	2	78	7	3				68	6+	2+				M	●	●	●	●	↕	↕	↕	↕
LSR	LSRA	44	2	1																A	●	●	●	●	R	↕	●	↕
	LSRB	54	2	1																B 	●	●	●	●	R	↕	●	↕
	LSR				04	6	2	74	7	3				64	6+	2+				M	●	●	●	●	R	↕	●	↕
MUL		3D	11	1																A×B→D	●	●	●	●	●	↕	●	⑨
																				(Unsigned)								
NEG	NEGA	40	2	1																$\overline{A}$+1→A	●	●	⑧	●	↕	↕	↕	↕
	NEGB	50	2	1																$\overline{B}$+1→B	●	●	⑧	●	↕	↕	↕	↕
	NEG				00	6	2	70	7	3				60	6+	2+				$\overline{M}$+1→M	●	●	⑧	●	↕	↕	↕	↕
NOP		12	2	1																No Operation	●	●	●	●	●	●	●	●
OR	ORA				9A	4	2	BA	5	3	8A	2	2	AA	4+	2+				A∨M→A	●	●	●	●	↕	↕	R	●
	ORB				DA	4	2	FA	5	3	CA	2	2	EA	4+	2+				B∨M→B	●	●	●	●	↕	↕	R	●
	ORCC										1A	3	2							CC∨IMM→CC	(			⑦				)
PSH	PSHS	34	5+④	2																Push Registers on	●	●	●	●	●	●	●	●
																				S Stack								
	PSHU	36	5+④	2																Push Registers on	●	●	●	●	●	●	●	●
																				U Stack								
PUL	PULS	35	5+④	2																Pull Registers	(			⑩				)
																				from S Stack								
	PULU	37	5+④	2																Pull Registers	(			⑩				)
																				from U Stack								
ROL	ROLA	49	2	1																A	●	●	●	●	↕	↕	↕	↕
	ROLB	59	2	1																B 	●	●	●	●	↕	↕	↕	↕
	ROL				09	6	2	79	7	3				69	6+	2+				M	●	●	●	●	↕	↕	↕	↕
ROR	RORA	46	2	1																A	●	●	●	●	↕	↕	●	↕
	RORB	56	2	1																B 	●	●	●	●	↕	↕	●	↕
	ROR				06	6	2	76	7	3				66	6+	2+				M	●	●	●	●	↕	↕	●	↕
RTI		3B	6/15	1																Return from	(			⑦				)
																				Interrupt								
RTS		39	5	1																Return from	●	●	●	●	●	●	●	●
																				Subroutine								
SBC	SBCA				92	4	2	B2	5	3	82	2	2	A2	4+	2+				A−M−C→A	●	●	⑧	●	↕	↕	↕	↕
	SBCB				D2	4	2	F2	5	3	C2	2	2	E2	4+	2+				B−M−C→B	●	●	⑧	●	↕	↕	↕	↕
SEX		1D	2	1																Sign Extend B into A	●	●	●	●	↕	↕	●	●
																				{ Bのビット7=1 FF→A								
																				Bのビット7=0 0→A								

(Continued)

INSTRUCTIONS/ FORMS		IMP ACCM REG			DIRECT			EXTND			IMMED			INDEX①			RELATIVE			DESCRIPTION	7	6	5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~⑤	#		E	F	H	I	N	Z	V	C
ST	STA				97	4	2	B7	5	3				A7	4+	2+				A→M	●	●	●	●	↕	↕	R	●
	STB				D7	4	2	F7	5	3				E7	4+	2+				B→M	●	●	●	●	↕	↕	R	●
	STD				DD	5	2	FD	6	3				ED	5+	2+				D→M M+1	●	●	●	●	↕	↕	R	●
	STS				10	6	3	10	7	4				10	6+	3+				S→M M+1	●	●	●	●	↕	↕	R	●
					DF			FF						EF														
	STU				DF	5	2	FF	6	3				EF	5+	2+				U→M M+1	●	●	●	●	↕	↕	R	●
	STX				9F	5	2	BF	6	3				AF	5+	2+				X→M M+1	●	●	●	●	↕	↕	R	●
	STY				10	6	3	10	7	4				10	6+	3+				Y→M M+1	●	●	●	●	↕	↕	R	●
					9F			BF						AF														
SUB	SUBA				90	4	2	B0	5	3	80	2	2	A0	4+	2+				A−M→A	●	●	⑧	●	↕	↕	↕	↕
	SUBB				D0	4	2	F0	5	3	C0	2	2	E0	4+	2+				B−M→B	●	●	⑧	●	↕	↕	↕	↕
	SUBD				93	6	2	B3	7	3	83	4	3	A3	6+	2+				D−M M+1→D	●	●	●	●	↕	↕	↕	↕
SWI	SWI⑥	3F	19	1																Software interrupt 1	S	S	●	S	●	●	●	●
	SWI2⑥	10	20	2																Software interrupt 2	S	●	●	●	●	●	●	●
		3F																										
	SWI3⑥	11	20	2																Software interrupt 3	S	●	●	●	●	●	●	●
		3F																										
SYNC		13	≧4	1																Synchronize to interrupt	●	●	●	●	●	●	●	●
TFR	R1,R2	1F	6	2																R1→R2②	(	—	—	⑩	—	—	—	)
TST	TSTA	4D	2	1																Test A	●	●	●	●	↕	↕	R	●
	TSTB	5D	2	1																Test B	●	●	●	●	↕	↕	R	●
	TST				0D	6	2	7D	7	3				6D	6+	2+				Test M	●	●	●	●	↕	↕	R	●

(NOTES)

① This column gives a base cycle and byte count To obtain total count, and the values obtained from the INDEXED ADDRESSING MODES table
② R1 and R2 may be any pair of 8 bit or any pair of 16 bit registers
The 8 bit registers are A, B, CC, DP
The 16 bit registers are X, Y, U, S, D, PC
③ EA is the effective address
④ The PSH and PUL instructions require 5 cycle plus 1 cycle for each byte pushed or pulled
⑤ 5(6) means 5 cycles if branch not taken, 6 cycles if taken
⑥ SWI sets 1 and F bits SW12 and SW13 do not affect I and F
⑦ Conditions Codes set as a direct result of the instruction
⑧ Value of half-carry flag is undefined
⑨ Special Case—Carry set if b7 is SET
⑩ Condition Codes set as a direct result of the instruction if CC is specified, and not affected otherwise

LEGEND

OP	Operation Code (Hexadecimal)	Z	Zero (byte)
~	Number of MPU Cycles	V	Overflow, 2's complement
#	Number of Program Bytes	C	Carry from bit 7
+	Arithmetic Plus	↕	Test and set if true, cleared otherwise
−	Arithmetic Minus	●	Not Affected
x	Multiply	CC	Condition Code Register
M̄	Complement of M		Concatenation
→	Transfer Into	∨	Logical or
H	Half-carry (from bit 3)	∧	Logical and
N	Negative (sign bit)	⊕	Logical Exclusive or

Table 12. Hexadecimal Values of Machine Codes

OP	Mnem	Mode	~	#	OP	Mnem	Mode	~	#	OP	Mnem	Mode	~	#
00	NEG	Direct	6	2	30	LEAX	Indexed	4+	2+	60	NEG	Indexed	6+	2+
01	*	↑			31	LEAY	↑	4+	2+	61	*	↑		
02	*				32	LEAS	↓	4+	2+	62	*			
03	COM		6	2	33	LEAU	Indexed	4+	2+	63	COM		6+	2+
04	LSR		6	2	34	PSHS	Implied	5+	2	64	LSR		6+	2+
05	*				35	PULS	↑	5+	2	65	*			
06	ROR		6	2	36	PSHU		5+	2	66	ROR		6+	2+
07	ASR		6	2	37	PULU		5+	2	67	ASR		6+	2+
08	ASL, LSL		6	2	38	*				68	ASL, LSL		6+	2+
09	ROL		6	2	39	RTS		5	1	69	ROL		6+	2+
0A	DEC		6	2	3A	ABX	↓	3	1	6A	DEC		6+	2+
0B	*				3B	RTI	Implied	6, 15	1	6B	*			
0C	INC		6	2	3C	CWAI	Immed	≥20	2	6C	INC		6+	2+
0D	TST		6	2	3D	MUL	Implied	11	1	6D	TST		6+	2+
0E	JMP	↓	3	2	3E	*				6E	JMP	↓	3+	2+
0F	CLR	Direct	6	2	3F	SWI	Implied	19	1	6F	CLR	Indexed	6+	2+
10	See	–	–	–	40	NEGA	Implied	2	1	70	NEG	Extended	7	3
11	Next Page	–	–	–	41	*	↑			71	*	↑		
12	NOP	Implied	2	1	42	*				72	*			
13	SYNC	Implied	≥4	1	43	COMA		2	1	73	COM		7	3
14	*				44	LSRA		2	1	74	LSR		7	3
15	*				45	*				75	*			
16	LBRA	Relative	5	3	46	RORA		2	1	76	ROR		7	3
17	LBSR	Relative	9	3	47	ASRA		2	1	77	ASR		7	3
18	*				48	ASLA, LSLA		2	1	78	ASL, LSL		7	3
19	DAA	Implied	2	1	49	ROLA		2	1	79	ROL		7	3
1A	ORCC	Immed	3	2	4A	DECA		2	1	7A	DEC		7	3
1B	*				4B	*				7B	*			
1C	ANDCC	Immed	3	2	4C	INCA		2	1	7C	INC		7	3
1D	SEX	Implied	2	1	4D	TSTA		2	1	7D	TST		7	3
1E	EXG	↕	8	2	4E	*	↓			7E	JMP	↓	4	3
1F	TFR	Implied	6	2	4F	CLRA	Implied	2	1	7F	CLR	Extended	7	3
20	BRA	Relative	3	2	50	NEGB	Implied	2	1	80	SUBA	Immed	2	2
21	BRN	↑	3	2	51	*	↑			81	CMPA	↑	2	2
22	BHI		3	2	52	*				82	SBCA		2	2
23	BLS		3	2	53	COMB		2	1	83	SUBD		4	3
24	BHS, BCC		3	2	54	LSRB		2	1	84	ANDA		2	2
25	BLO, BCS		3	2	55	*				85	BITA		2	2
26	BNE		3	2	56	RORB		2	1	86	LDA		2	2
27	BEQ		3	2	57	ASRB		2	1	87	*			
28	BVC		3	2	58	ASLB, LSLB		2	1	88	EORA		2	2
29	BVS		3	2	59	ROLB		2	1	89	ADCA		2	2
2A	BPL		3	2	5A	DECB		2	1	8A	ORA		2	2
2B	BMI		3	2	5B	*				8B	ADDA	↓	2	2
2C	BGE		3	2	5C	INCB		2	1	8C	CMPX	Immed	4	3
2D	BLT		3	2	5D	TSTB		2	1	8D	BSR	Relative	7	2
2E	BGT	↓	3	2	5E	*	↓			8E	LDX	Immed	3	3
2F	BLE	Relative	3	2	5F	CLRB	Implied	2	1	8F	*			

Legend: ~ Number of MPU cycles (less possible push pull or indexed-mode cycles)
Number of program bytes
* Denotes unused opcode

Table 12. Hexadecimal Values of Machine Codes (Cont.)

OP	Mnem	Mode	~	#	OP	Mnem	Mode	~	#	OP	Mnem	Mode	~	#
90	SUBA	Direct	4	2	C6	LDB	Immed	2	2	FC	LDD	Extended	6	3
91	CMPA	↑	4	2	C7	*	↑			FD	STD	↑	6	3
92	SBCA		4	2	C8	EORB		2	2	FE	LDU	↓	6	3
93	SUBD		6	2	C9	ADCB		2	2	FF	STU	Extended	6	3
94	ANDA		4	2	CA	ORB		2	2					
95	BITA		4	2	CB	ADDB		2	2					
96	LDA		4	2	CC	LDD		3	3	**2 Bytes Opcode**				
197	STA		4	2	CD	*	↓							
98	EORA		4	2	CE	LDU	Immed	3	3	1021	LBRN	Relative	5	4
99	ADCA		4	2	CF	*				1022	LBHI	↑	5(6)	4
9A	ORA		4	2						1023	LBLS		5(6)	4
9B	ADDA		4	2	D0	SUBB	Direct	4	2	1024	LBHS, LBCC		5(6)	4
9C	CMPX		6	2	D1	CMPB	↑	4	2	1025	LBCS, LBLO		5(6)	4
9D	JSR		7	2	D2	SBCB		4	2	1026	LBNE		5(6)	4
9E	LDX	↓	5	2	D3	ADDD		6	2	1027	LBEQ		5(6)	4
9F	STX	Direct	5	2	D4	ANDB		4	2	1028	LBVC		5(6)	4
					D5	BITB		4	2	1029	LBVS		5(6)	4
A0	SUBA	Indexed	4+	2+	D6	LDB		4	2	102A	LBPL		5(6)	4
A1	CMPA	↑	4+	2+	D7	STB		4	2	102B	LBMI		5(6)	4
A2	SBCA		4+	2+	D8	EORB		4	2	102C	LBGE		5(6)	4
A3	SUBD		6+	2+	D9	ADCB		4	2	102D	LBLT		5(6)	4
A4	ANDA		4+	2+	DA	ORB		4	2	102E	LBGT	↓	5(6)	4
A5	BITA		4+	2+	DB	ADDB		4	2	102F	LBLE	Relative	5(6)	4
A6	LDA		4+	2+	DC	LDD		5	2	103F	SWI2	Implied	20	2
A7	STA		4+	2+	DD	STD		5	2	1083	CMPD	Immed	5	4
A8	EORA		4+	2+	DE	LDU	↓	5	2	108C	CMPY	↕	5	4
A9	ADCA		4+	2+	DF	STU	Direct	5	2	108E	LDY	Immed	4	4
AA	ORA		4+	2+						1093	CMPD	Direct	7	3
AB	ADDA		4+	2+	E0	SUBB	Indexed	4+	2+	109C	CMPY	↑	7	3
AC	CMPX		6+	2+	E1	CMPB	↑	4+	2+	109E	LDY	↓	6	3
AD	JSR		7+	2+	E2	SBCB		4+	2+	109F	STY	Direct	6	3
AE	LDX	↓	5+	2+	E3	ADDD		6+	2+	10A3	CMPD	Indexed	7+	3+
AF	STX	Indexed	5+	2+	E4	ANDB		4+	2+	10AC	CMPY	↑	7+	3+
					E5	BITB		4+	2+	10AE	LDY	↓	6+	3+
B0	SUBA	Extended	5	3	E6	LDB		4+	2+	10AF	STY	Indexed	6+	3+
B1	CMPA	↑	5	3	E7	STB		4+	2+	10B3	CMPD	Extended	8	4
B2	SBCA		5	3	E8	EORB		4+	2+	10BC	CMPY	↑	8	4
B3	SUBD		7	3	E9	ADCB		4+	2+	10BE	LDY	↓	7	4
B4	ANDA		5	3	EA	ORB		4+	2+	10BF	STY	Extended	7	4
B5	BITA		5	3	EB	ADDB		4+	2+	10CE	LDS	Immed	4	4
B6	LDA		5	3	EC	LDD		5+	2+	10DE	LDS	Direct	6	3
B7	STA		5	3	ED	STD		5+	2+	10DF	STS	Direct	6	3
B8	EORA		5	3	EE	LDU	↓	5+	2+	10EE	LDS	Indexed	6+	3+
B9	ADCA		5	3	EF	STU	Indexed	5+	2+	10EF	STS	Indexed	6+	3+
BA	ORA		5	3						10FE	LDS	Extended	7	4
BB	ADDA		5	3	F0	SUBB	Extended	5	3	10FF	STS	Extended	7	4
BC	CMPX		7	3	F1	CMPB	↑	5	3	113F	SWI3	Implied	20	2
BD	JSR		8	3	F2	SBCB		5	3	1183	CMPU	Immed	5	4
BE	LDX	↓	6	3	F3	ADDD		7	3	118C	CMPS	Immed	5	4
BF	STX	Extended	6	3	F4	ANDB		5	3	1193	CMPU	Direct	7	3
					F5	BITB		5	3	119C	CMPS	Direct	7	3
C0	SUBB	Immed	2	2	F6	LDB		5	3	11A3	CMPU	Indexed	7+	3+
C1	CMPB	↑	2	2	F7	STB		5	3	11AC	CMPS	Indexed	7+	3+
C2	SBCB		2	2	F8	EORB		5	3	11B3	CMPU	Extended	8	4
C3	ADDD		4	3	F9	ADCB		5	3	11BC	CMPS	Extended	8	4
C4	ANDB	↓	2	2	FA	ORB	↓	5	3					
C5	BITB	Immed	2	2	FB	ADDB	Extended	5	3					

Note All unused opcodes are both undefined and illegal

Note for Use

Compatibility with NMOS MPU (HD6809)

The difference between HD6309 (CMOS) and HD6809 (NMOS) is shown in table 13.

Execution Sequence of CLR Instruction

Cycle-by-cycle flow of CLR instruction (direct, extended, indexed addressing mode) is shown below. In this sequence the contents of the memory location specified by the operand is read before writing 00 into it. Note that status flags, such as IRQ Flag, will be cleared by this extra data read operation when accessing the control/status register (sharing the same address between read and write) of peripheral devices.

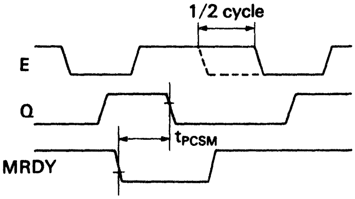
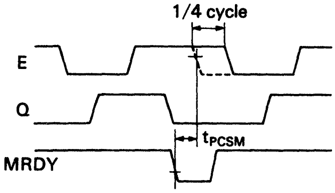
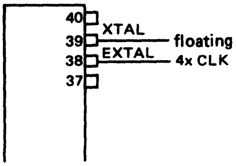
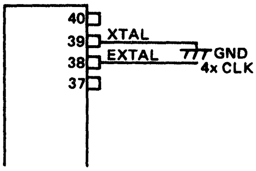
Example: CLR (Extended)

```
$8000    CLR    $A000
$A000    FCB    $80
```

Cycle #	Address	Data	R/$\overline{W}$	Description
1	8000	7F	1	Opcode Fetch
2	8001	A0	1	Operand Address, High Byte
3	8002	00	1	Operand Address, Low Byte
4	FFFF	*	1	$\overline{\text{VMA}}$ Cycle
5	A000	80	1	Read the Data
6	FFFF	*	1	$\overline{\text{VMA}}$ Cycle
7	A000	00	0	Store Fixed 00 into Specified Location

* The data bus has the data at that particular address.

Table 13. Difference between HD6309 and HD6809

Item		HD6309 (CMOS)	HD6809 (NMOS)
MRDY	Stretch Unit	integral multiples of half (1/2) bus cycles  	integral multiples of quarter (1/4) bus cycles 
	Stretch Time	5 µs max	10 µs max
$\overline{\text{DMA/BREQ}}$	Auto-refresh	None	Executed
External Clock Input		XTAL floating  	XTAL grounded 

Application Note for System Design

At the trailing edge of the address bus, the noise pulses may appeare on the output signals in HD6309.

Note the noise pulses and the following measures against them.

Noise Occurrence Condition: As shown in figure 20, the noise pulses which are 0.8 V or over may appear on E and Q clocks when the address bus changes from high to low.

If the address buses (A_0 -A_{15}, and R/$\overline{W}$) change from high to low, the transient current flows through the GND. The noise pulses are generated on the LSI's V_{SS} pins according to the current and to the impedance state of the GND wirings.

Figure 21 shows the noise voltage dependency on the each parameter.

Figure 23 shows the noise voltage dependency on the load capacitance of the address bus.

Note: The noise level should be carefully checked because it depends on the each parameter of actual application system.

Noise Reduction:

1. Control each parameter such as Cd, V $_{CC}$, Zg in figure 21, and the noise level is reduced to be allowable.
2. Insert a bypass capacitor between the V_{CC} and the GND of the HD6309.
3. Connect the CMOS buffer with noise margin to E and Q clocks.
4. Insert the damping registors to the address bus. That is effective for the noise level to reduce less than 0.8 V. The damping resistor is about 40-50 Ω on the higher byte of the address bus (A $_{15}$ − A_8)and about 130-140 Ω on the lower byte of the address bus (A_7 −A_0), and R/$\overline{W}$ as shown in figure 22. Electrical characteristics do not change by inserting the damping resistors.

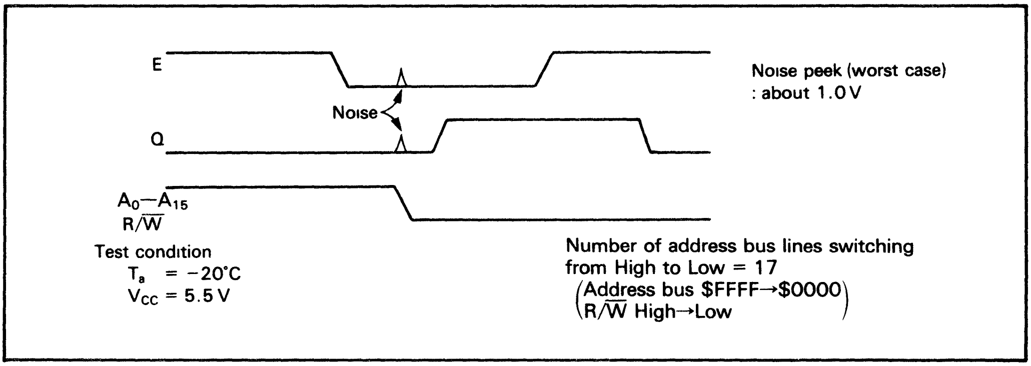


Figure 20. Noise at Address Bus Output Changing

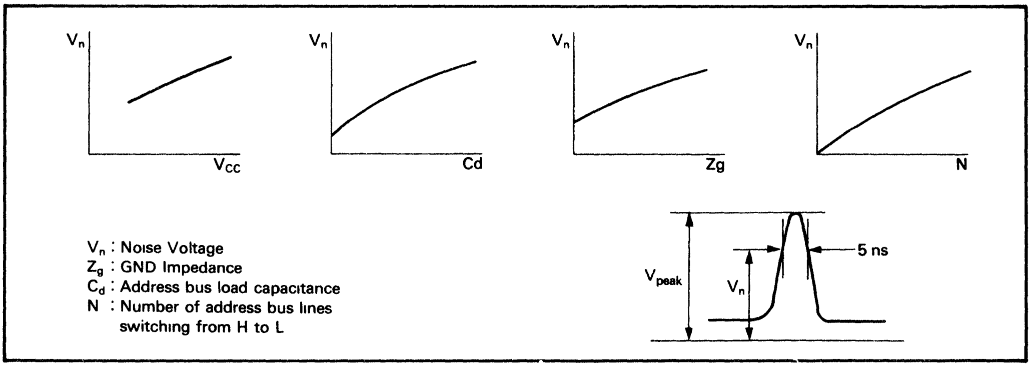


Figure 21. Dependency of the Noise Voltage on Each Parameter

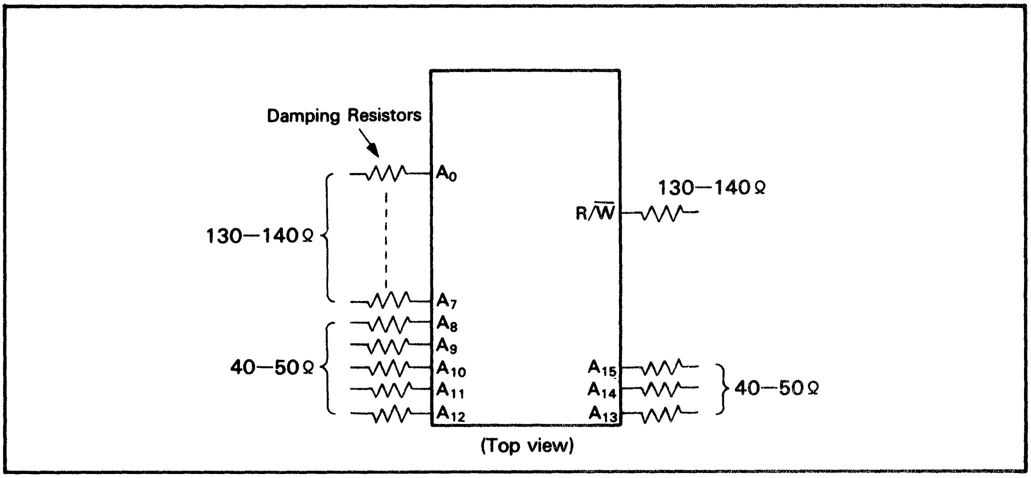


Figure 22. Connecting Damping Resistors to Address Bus

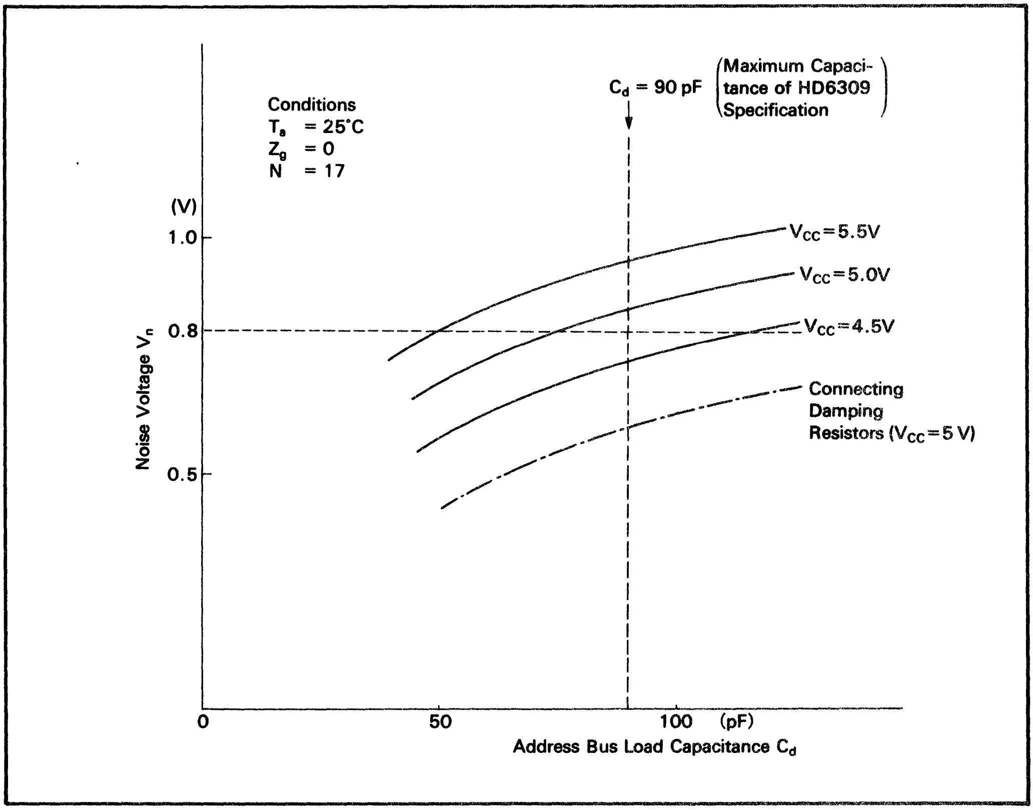


Figure 23. Dependency of the Noise Voltage on the Load Capacitance of the Address Bus

Absolute Maximum Ratings

Item	Symbol	Value	Unit
Supply Voltage	V_{CC}[1]	−0.3 to +7.0	V
Input Voltage	V_{in}[1]	−0.3 to +7.0	V
Maxımum Output Current	$\|I_o\|$[2]	5	mA
Maximum Total Output Current	$\|\Sigma I_o\|$[3]	100	mA
Operatıng Temperature	T_{opr}	−20 to +75	°C
Storage Temperature	T_{stg}	−55 to +150	°C

Notes 1 ,With respect to V_{SS} (system GND)

2 Maxımum output current ıs the maxımum currents whıch can flow out from one output termınal and I/O common termınal (A_0 -A_{15}, R/$\overline{W}$, D_0 -D_7 , BA, BS, Q, E)

3 Maxımum total output current ıs the total sum of output currents whıch can flow out sımultaneously from output termınals and I/O common termınals (A_0 -A_{15}, R/$\overline{W}$, D_0 -D_7 , BA, BS, Q, E)

4 Permanent LSI damage may occur ıf maximum ratıngs are exceeded Normal operatıon should be under recommended operatıng condıtıons If these condıtıons are exceeded, ıt could affect relıabılıty of LSI

Recommended Operating Conditions

Item		Symbol	Min	Typ	Max	Unit
Supply Voltage		V_{CC}[1]	4.5	5.0	5.5	V
Input Voltage	EXTAL	V_{IL}[1]	−0.3		0.6	V
	Other Inputs		−0.3		0.8	V
	$\overline{RES}$	V_{IH}[1]	V_{CC}−0.5		V_{CC}	V
	EXTAL		V_{CC}×0.7		V_{CC}	V
	Other Inputs		2.0		V_{CC}	V
Operatıng Temperature		T_{opr}	−20	25	75	°C

Note· 1 Wıth respect to Vss (system GND)

Electrical Characteristics

DC Characteristics (V_{CC}=5.0 V ± 10%, V_{SS}=0 V, Ta=−20 to +75°C, unless otherwise noted.)

			HD63B09			HD63C09				
Item		**Symbol**	**Min**	**Typ**	**Max**	**Min**	**Typ**	**Max**	**Unit**	**Test Condition**
Input High Voltage	$\overline{RES}$	V_{IH}	$V_{CC}-0.5$		V_{CC}	$V_{CC}-0.5$		V_{CC}	V	
	EXTAL		$V_{CC}\times0.7$		V_{CC}	$V_{CC}\times0.7$		V_{CC}		
	Other Inputs		2.0		V_{CC}	2.0		V_{CC}		
Input Low Voltage	EXTAL	V_{IL}	−0.3		0.6	−0.3		0.6	V	
	Other Inputs		−0.3		0.8	−0.3		0.8		
Input Leakage Current	Except EXTAL, XTAL	I_{in}	−2.5		2.5	−2.5		2.5	μA	V_{in}=0 to V_{CC}, V_{CC}=max
Three State (Off State) Input Current	D_0-D_7	I_{TSI}	−10		10	−10		10	μA	V_{in}=0.4 to V_{CC}, V_{CC}=max
	A_0-A_{15}, $R/\overline{W}$		−10		10	−10		10		
Output High Voltage	D_0-D_7	V_{OH}	4.1			4.1			V	I_{LOAD}=−400μA
			$V_{CC}-0.1$			$V_{CC}-0.1$				I_{LOAD}≦−10μA
	A_0-A_{15}, $R/\overline{W}$, Q, E		4.1			4.1				I_{LOAD}=−400μA
			$V_{CC}-0.1$			$V_{CC}-0.1$				I_{LOAD}≦−10μA
	BA, BS		4.1			4.1				I_{LOAD}=−400μA
			$V_{CC}-0.1$			$V_{CC}-0.1$				I_{LOAD}≦−10μA
Output Low Voltage		V_{OL}			0.5			0.5	V	I_{LOAD}=2mA
Input Capacitance	D_0-D_7	C_{in}			15			15	pF	V_{in}=0V, T_a=25°C, f=1MHz
	Except D_0-D_7				10			10		
Output Capacitance	A_0-A_{15}, $R/\overline{W}$, BA, BS	C_{out}			12			12	pF	
Current Dissipation		I_{CC}			24			36	mA	Operating
					15			18		Sleeping

AC Characteristics (V_{CC}=5.0 V ± 10%, V_{SS}=0 V, Ta=−20 to +75°C, unless otherwise noted.)

Clock Timing

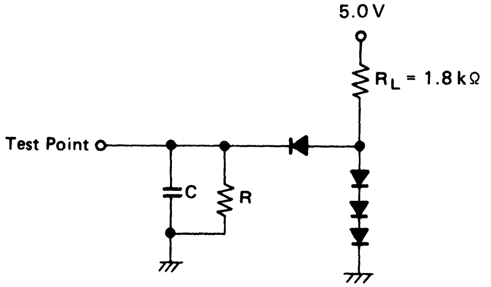
		HD63B09			HD63C09				
Item	**Symbol**	**Min**	**Typ**	**Max**	**Min**	**Typ**	**Max**	**Unit**	**Test Condition**
Frequency of Operation (Crystal External Input)	f_{XTAL}	2		8	2		12	MHz	Figs. 25, 26
Cycle Time	t_{cyc}	500		2000	333		2000	ns	
Total Up Time	t_{UT}	480			310			ns	
Processor Clock High	t_{PWEH}	220		5000	140		5000	ns	
Processor Clock Low	t_{PWEL}	210		1000	140		1000	ns	
E Rise and Fall Time	t_{Er}, t_{Ef}			20			20	ns	
E_{Low} to Q_{High} Time	t_{AVS}	100		140	70		100	ns	
Q Clock High	t_{PWQH}	220		1000	140		1000	ns	
Q Clock Low	t_{PWQL}	220		5000	140		5000	ns	
Q Rise and Fall Time	t_{Qr}, t_{Qf}			20			20	ns	
Q_{Low} to E_{Low} Time	t_{QE}	100			70			ns	

Bus Timing

			HD63B09			HD63C09				
Item		**Symbol**	**Min**	**Typ**	**Max**	**Min**	**Typ**	**Max**	**Unit**	**Test Condition**
Address Delay		t_{AD}			110			110	ns	Figs. 25, 26
Peripheral Read Access Time (t_{UT}-t_{AD}-t_{DSR}=t_{ACC})		t_{ACC}	330			160			ns	
Data Set Up Time (Read)		t_{DSR}	40			40			ns	
Input Data Hold Time		t_{DHR}	10			10			ns	
Address Hold Time	Ta=0 to +75°C	t_{AH}	20			20			ns	
	Ta=−20 to 0°C		10			10				
Data Delay Time (Write)		t_{DDW}			110			70	ns	
Output Hold Time	Ta=0 to +75°C	t_{DHW}	30			30			ns	
	Ta=−20 to 0°C		20			20				

Processor Control Timing

		HD63B09			HD63C09				
Item	Symbol	Min	Typ	Max	Min	Typ	Max	Unit	Test Condition
MRDY Set Up Time	t_{PCSM}	110			70			ns	Figs. 3 – 7
MRDY Set Up Time 2	t_{PCSM2}	240			160			ns	Figs. 11, 12
Interrupts Set Up Time	t_{PCS}	110			70			ns	
$\overline{\text{HALT}}$ Set Up Time	t_{PCSH}	110			70			ns	
$\overline{\text{RES}}$ Set Up Time	t_{PCSR}	110			110			ns	
$\overline{\text{DMA/BREQ}}$ Set Up Time	t_{PCSD}	110			70			ns	
Processor Control Rise and Fall Time	t_{PCr}, t_{PCf}			100			100	ns	
Crystal Oscillator Start Time	t_{RC}	20			20			ms	



- C= 30 pF (BA, BS)
  130 pF (D_0-D_7, E, Q)
  90 pF (A_0-A_{15}, $R/\overline{W}$)
- R= 10 kΩ (D_0-D_7)
  10 kΩ (A_0-A_{15}, E, Q, $R/\overline{W}$)
  10 kΩ (BA, BS)

All diodes are 1S2074Ⓗ or equivalent.
C includes stray capacitance.

Figure 24. Bus Timing Test Load

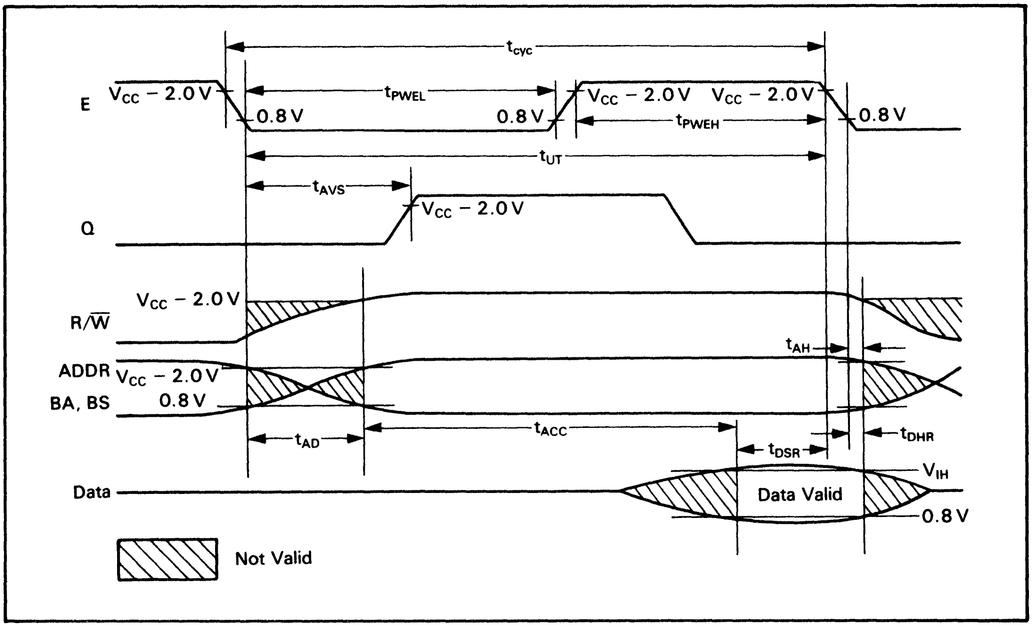


Figure 25. Read Data from Memory or Peripherals

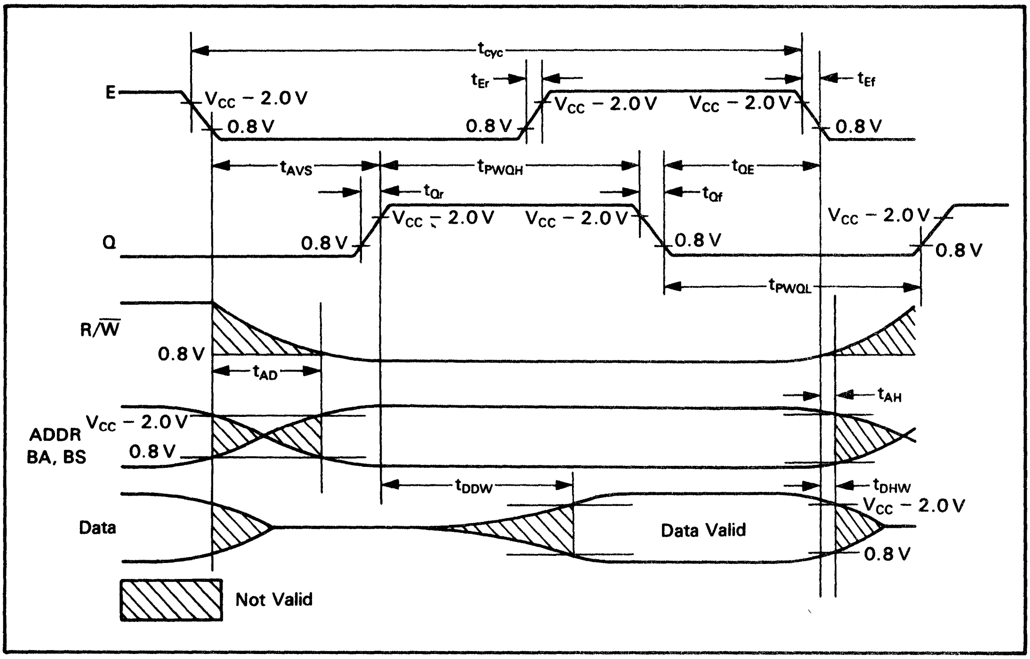


Figure 26. Write Data to Memory or Peripherals

HD63B09E, HD63C09E
CMOS MPU (Micro Processing Unit)

The HD6309E is the highest 8-bit microprocessor of HMCS6800 family, which is just compatible with the conventional HD6809E.

The HD6309E has hardware and software features which make it an ideal processor for higher level language execution or standard controller applications. External clock inputs are provided to allow synchronization with peripherals, systems or other MPUs.

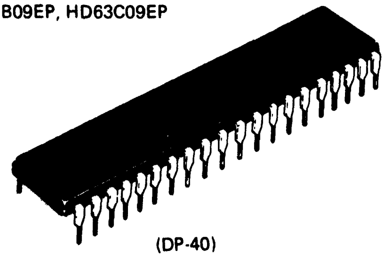
The HD6309E is complete CMOS device and the power dissipation is extremely low. Moreover, the SYNC and CWAI instruction makes low power application possible.

■ FEATURES

- Hardware – Interface with All HMCS6800 Peripherals
- Software – Object Code Compatible with the HD6809E
- Low Power Consumption Mode (Sleep mode)
  SYNC state of SYNC Instruction
  WAIT state of CWAI Instruction
- External Clock Inputs, E and Q, Allow Synchronization
- Wide Operation Range
  f = 0.5 to 3MHz (V_{CC}=5V±10%)

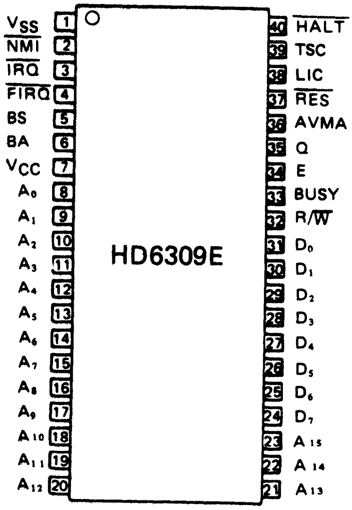
Type No.	Bus Timing
HD63B09E	2.0MHz
HD63C09E	3.0MHz

HD63B09EP, HD63C09EP



(DP-40)

■ PIN ARRANGEMENT



(Top View)

■ ABSOLUTE MAXIMUM RATINGS

Item	Symbol	Value	Unit
Supply Voltage	V_{CC}*	-0.3 ~ +7.0	V
Input Voltage	V_{in}*	-0.3 ~ +7.0	V
Maximum Output Current	$\|I_O\|$**	5	mA
Maximum Total Output Current	$\|\Sigma I_O\|$***	100	mA
Operating Temperature	T_{opr}	-20 ~ +75	°C
Storage Temperature	T_{stg}	-55 ~ +150	°C

* With respect to V_{SS} (SYSTEM GND)

** Maximum output current is the maximum currents which can flow out from one output terminal and I/O common terminal. (A_0 ~ A_{15}, R/$\overline{W}$, D_0 ~ D_7, BA, BS, LIC, AVMA, BUSY)

*** Maximum total output current is the total sum of output currents which can flow out simultaneously from output terminals and I/O common terminals. (A_0 ~ A_{15}, R/$\overline{W}$, D_0 ~ D_7, BA, BS, LIC, AVMA, BUSY)

(NOTE) Permanent LSI damage may occur if maximum ratings are exceeded. Normal operation should be under recommended operating conditions. If these conditions are exceeded, it could affect reliability of LSI.

■ RECOMMENDED OPERATING CONDITIONS

Item		Symbol	min	typ	max	Unit
Supply Voltage		V_{CC}*	4.5	5.0	5.5	V
Input Voltage	Logic, $\overline{RES}$	V_{IL}*	–0.3	–	0.8	V
	E, Q	V_{ILC}*	–0.3	–	0.4	V
	Logic	V_{IH}*	2.0	–	V_{CC}	V
	E, Q		3.0	–	V_{CC}	V
	$\overline{RES}$		V_{CC}–0.5	–	V_{CC}	V
Operating Temperature		T_{opr}	–20	25	75	°C

* With respect to V_{SS} (SYSTEM GND)

■ ELECTRICAL CHARACTERISTICS

● DC CHARACTERISTICS (V_{CC}=5.0V±10%, V_{SS}=0V, Ta=- 20 ~ +75°C, unless otherwise noted.)

Item		Symbol	Test Condition	HD63B09E			HD63C09E			Unit
				min	typ*	max	min	typ*	max	
Input "High" Voltage	Logic	V_{IH}		2.0	–	V_{CC}	2.0	–	V_{CC}	V
	E, Q	V_{IH}		3.0	–	V_{CC}	3.0	–	V_{CC}	V
	$\overline{RES}$	V_{IHR}		V_{CC}–0.5	–	V_{CC}	V_{CC}–0.5	–	V_{CC}	V
Input "Low" Voltage	Logic, $\overline{RES}$	V_{IL}		–0.3	–	0.8	–0.3	–	0.8	V
	E, Q	V_{ILC}		–0.3	–	0.4	–0.3	–	0.4	V
Input Leakage Current	Logic, Q, $\overline{RES}$	I_{in}	V_{in}=0 ~ V_{CC}, V_{CC}=max	–2.5	–	2.5	–2.5	–	2.5	μA
	E			–10	–	10	–10	–	10	μA
Output "High" Voltage	D_0 ~ D_7	V_{OH}	I_{LOAD}=–400μA	4.1	–	–	4.1	–	–	V
			I_{LOAD}≦–10μA	V_{CC}–0.1	–	–	V_{CC}–0.1	–	–	
	A_0 ~ A_{15}, R/$\overline{W}$		I_{LOAD}=–400μA	4.1	–	–	4.1	–	–	V
			I_{LOAD}≦–10μA	V_{CC}–0.1	–	–	V_{CC}–0.1	–	–	
	BA, BS, LIC, AVMA, BUSY		I_{LOAD}=–400μA	4.1	–	–	4.1	–	–	V
			I_{LOAD}≦–10μA	V_{CC}–0.1	–	–	V_{CC}–0.1	–	–	
Output "Low" Voltage		V_{OL}	I_{LOAD}=2mA	–	–	0.5	–	–	0.5	V
Input Capacitance	D_0 ~ D_7, Logic Input Q, $\overline{RES}$	C_{in}	V_{in}=0V, Ta=25°C, f=1MHz	–	10	15	–	10	15	pF
	E			–	30	50	–	30	50	pF
Output Capacitance	A_0 ~ A_{15}, R/$\overline{W}$, BA, BS, LIC, AVMA, BUSY	C_{out}	V_{in}=0V, Ta=25°C, f=1MHz	–	10	15	–	10	15	pF
Frequency of Operation	E, Q	f		0.5	–	2.0	0.5	–	3.0	MHz
Three-State (Off State) Input Current	D_0 ~ D_7	I_{TSI}	V_{in}=0.4~V_{CC}, V_{CC}=max	–10	–	10	–10	–	10	μA
	A_0 ~ A_{15}, R/$\overline{W}$			–10	–	10	–10	–	10	μA
Current Dissipation		I_{CC}	Operating	–	–	20	–	–	30	mA
			Sleeping	–	–	10	–	–	15	

*Ta=25°C, V_{CC}=5V

- **AC CHARACTERISTICS (V_{CC}=5.0V±10%, V_{SS}=0, Ta=−20 ~ +75°C, unless otherwise noted.)**

1. CLOCK TIMING

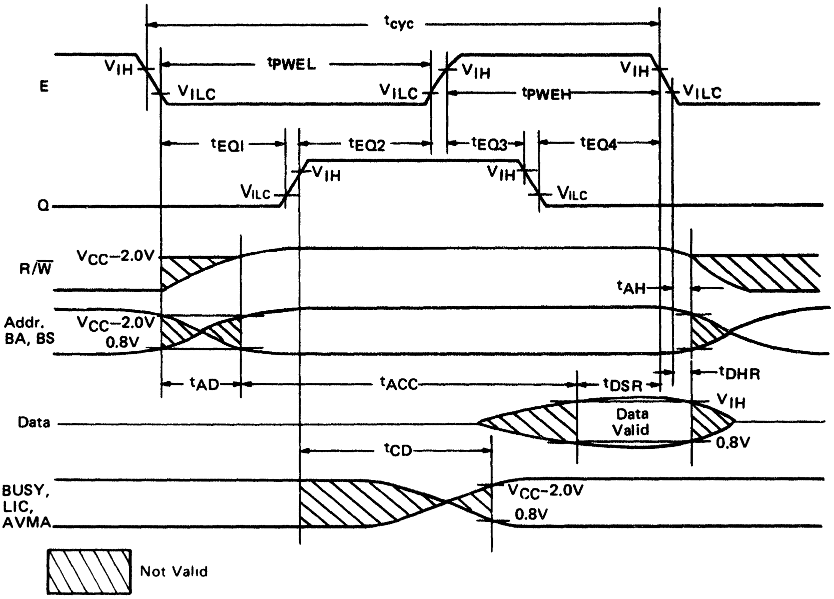
Item		Symbol	Test Condition	HD63B09E			HD63C09E			Unit
				min	typ	max	min	typ	max	
Cycle Time		t_{cyc}	Fig. 1,2	500	–	2000	333	–	2000	ns
E Clock "Low"		t_{PWEL}		210	–	1000	140	–	1000	ns
E Clock "High" (Measured at V_{IH})		t_{PWEH}		220	–	1000	140	–	1000	ns
E Rise and Fall Time		t_{Er}, t_{Ef}		–	–	20	–	–	15	ns
Q Clock "High"		t_{PWQH}		220	–	1000	140	–	1000	ns
Q Rise and Fall Time		t_{Qr}, t_{Qf}		–	–	20	–	–	15	ns
E "Low" to Q Rising	E "Low"→Q"High"	t_{EQ1}		100	–	–	65	–	–	ns
Q "High" to E Rising	Q "High"→E "High"	t_{EQ2}		100	–	–	65	–	–	ns
E "High" to Q Falling	E "High"→Q"Low"	t_{EQ3}		100	–	–	65	–	–	ns
Q "Low" to E Falling	Q "Low"→E "Low"	t_{EQ4}		100	–	–	65	–	–	ns

2. BUS TIMING

Item		Symbol	Test Condition	HD63B09E			HD63C09E			Unit
				min	typ	max	min	typ	max	
Address Delay		t_{AD}	Fig. 1, 2	–	–	110	–	–	110	ns
Address Hold Time (Address, R/$\overline{W}$, BA, BS)	Ta = 0 ~ 75°C	t_{AH}		20	–	–	20	–	–	ns
	Ta = −20~0°C			10	–	–	10	–	–	
Peripheral Read Access Times ($t_{cyc}-t_{Ef}-t_{AD}-t_{DSR}=t_{ACC}$)		t_{ACC}		330	–	–	185	–	–	ns
Data Setup Time (Read)		t_{DSR}		40	–	–	20	–	–	ns
Input Data Hold Time		t_{DHR}		20	–	–	20	–	–	ns
Data Delay Time (Write)		t_{DDW}		–	–	110	–	–	70	ns
Output Data Hold Time	Ta = 0~75°C	t_{DHW}		30	–	–	30	–	–	ns
	Ta = -20~0°C			20	–	–	20	–	–	

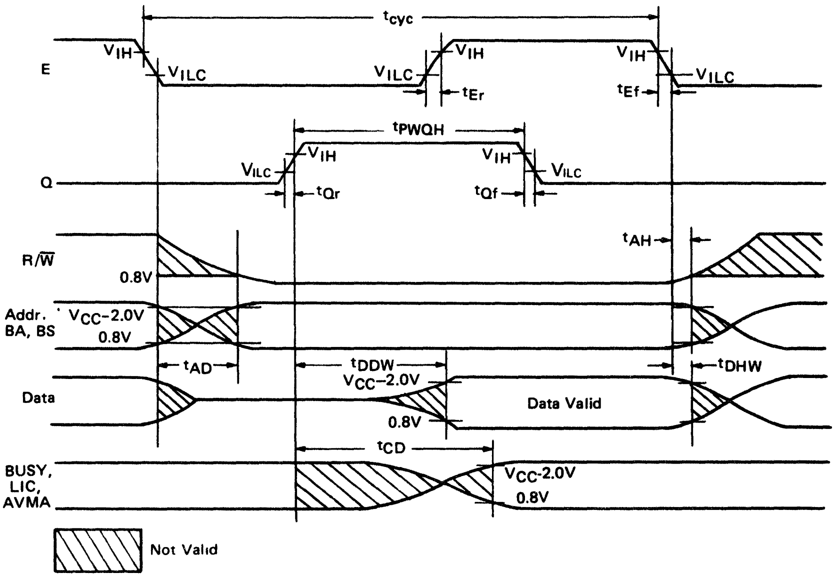
3. PROCESSOR CONTROL TIMING

Item	Symbol	Test Condition	HD63B09E			HD63C09E			Unit
			min	typ	max	min	typ	max	
Control Delay (BUSY, LIC, AVMA)	t_{CD}	Fig. 1, 2, 7 ~ 10, 14 and 17	–	–	200	–	–	130	ns
Interrupts Set Up Time	t_{PCS}		110	–	–	70	–	–	ns
$\overline{HALT}$ Set Up Time	t_{PCS}		110	–	–	70	–	–	ns
$\overline{RES}$ Set Up Time	t_{PCS}		110	–	–	70	–	–	ns
TSC Setup Time	t_{PCS}		110	–	–	70	–	–	ns
TSC Drive to Valid Logic Levels	t_{TSA}		–	–	120	–	–	120	ns
TSC Release MOS Buffers to High Impedance	t_{TSR}		–	–	110	–	–	110	ns
TSC Three-State Delay	t_{TSD}		–	–	80	–	–	80	ns
Processor Control Rise/Fall	t_{PCr}, t_{PCf}		–	–	100	–	–	100	ns
TSC Input Delay	t_{PCT}		30	–	–	30	–	–	ns



(NOTE) Waveform measurements for all inputs and outputs are specified at logic "High" = V_{IHmin} and logic "Low" = V_{ILmax} unless otherwise specified

Figure 1 Read Data from Memory or Peripherals



(NOTE) Waveform measurements for all inputs and outputs are specified at logic "High" = V_{IHmin} and logic "Low" = V_{ILmax} unless otherwise specified.

Figure 2 Write Data to Memory or Peripherals

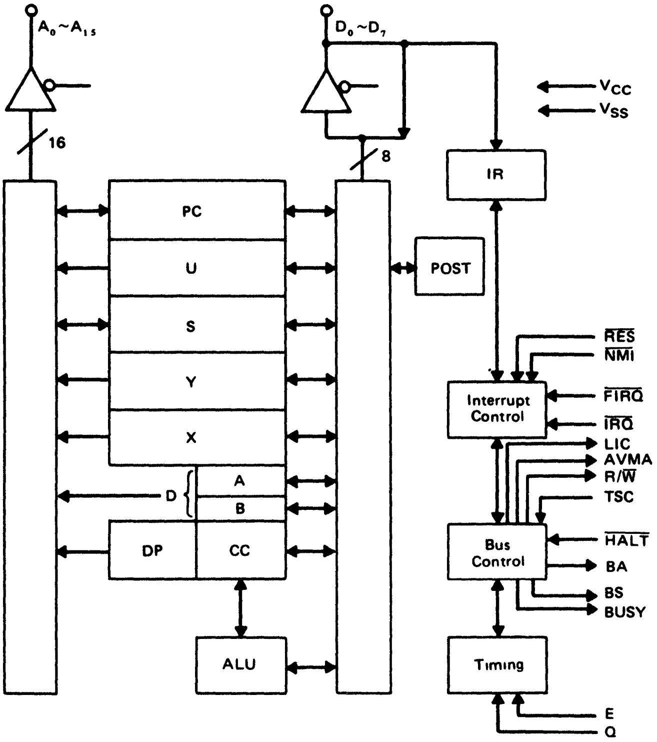
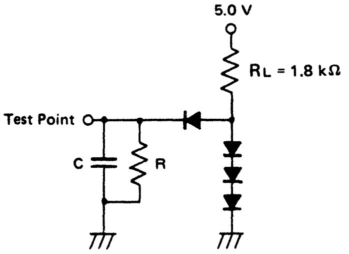


Figure 3 HD6309E Expanded Block Diagram



C = 30 pF for BA, BS, LIC, AVMA, BUSY
130 pF for $D_0 \sim D_7$
90 pF for $A_0 \sim A_{15}$, R/W̄

R = 10 kΩ for $D_0 \sim D_7$
10 kΩ for $A_0 \sim A_{15}$, R/W̄
10 kΩ for BA, BS, LIC, AVMA, BUSY

All diodes are 1S2074(H) or equivalent.
C includes stray capacitance.

Figure 4 Bus Timing Test Load

■ PROGRAMMING MODEL

As shown in Figure 5, the HD6309E adds three registers to the set available in the HD6800. The added registers include a Direct Page Register, the User Stack pointer and a second Index Register.

● Accumulators (A, B, D)

The A and B registers are general purpose accumulators which are used for arithmetic calculations and manipulation of data.

Certain instructions concatenate the A and B registers to form a single 16-bit accumulator. This is referred to as the D Register, and is formed with the A Register as the most significant byte.

● Direct Page Register (DP)

The Direct Page Register of the HD6309E serves to enhance the Direct Addressing Mode. The content of this register appears at the higher address outputs ($A_8 \sim A_{15}$) during direct addressing instruction execution. This allows the direct mode to be used at any place in memory, under program control. To ensure HD6800 compatibility, all bits of this register are cleared during Processor Reset.

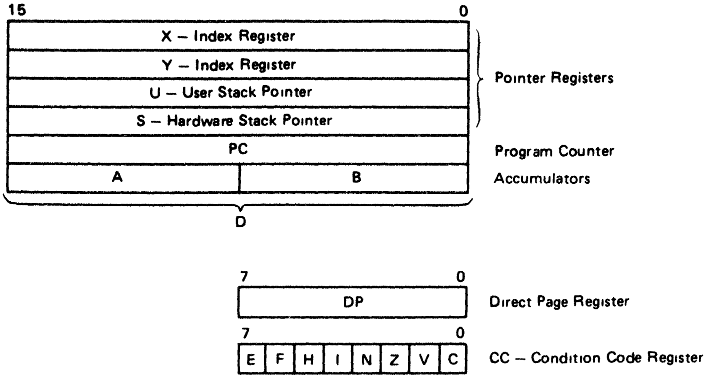


Figure 5 Programming Model of The Microprocessing Unit

- **Index Registers (X, Y)**

The Index Registers are used in indexed mode of addressing. The 16-bit address in this register takes part in the calculation of effective addresses. This address may be used to point to data directly or may be modified by an optional constant or register offset. During some indexed modes, the contents of the index register are incremented or decremented to point to the next item of tabular type data. All four pointer registers (X, Y, U, S) may be used as index registers.

- **Stack Pointer (U, S)**

The Hardware Stack Pointer (S) is used automatically by the processor during subroutine calls and interrupts. The User Stack Pointer (U) is controlled exclusively by the programmer thus allowing arguments to be passed to and from subroutines with ease. The U-register is frequently used as a stack marker. Both Stack Pointers have the same indexed mode addressing capabilities as the X and Y registers, but also support **Push** and **Pull** instructions. This allows the HD6309E to be used efficiently as a·stack processor, greatly enhancing its ability to support higher level languages and modular programming.

(NOTE) The stack pointers of the HD6309E point to the top of the stack, in contrast to the HD6800 stack pointer, which pointed to the next free location on stack.

- **Program Counter (PC)**

The Program Counter is used by the processor to point to the address of the next instruction to be executed by the processor. Relative Addressing is provided allowing the Program Counter to be used like an index register in some situations.

- **Condition Code Register (CC)**

The Condition Code Register defines the state of the processor at any given time. See Figure 6.

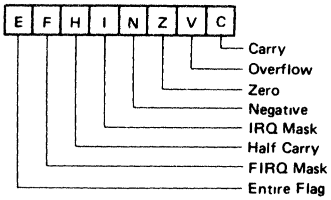


Figure 6 Condition Code Register Format

■ CONDITION CODE REGISTER DESCRIPTION

- **Bit 0 (C)**

Bit 0 is the carry flag, and is usually the carry from the binary ALU. C is also used to represent a 'borrow' from subtract like instructions (CMP, NEG, SUB, SBC) and is the complement of the carry from the binary ALU.

- **Bit 1 (V)**

Bit 1 is the overflow flag, and is set to a one by an operation which causes a signed two's complement arithmetic overflow. This overflow is detected in an operation in which the carry from the MSB in the ALU does not match the carry from the MSB-1.

- **Bit 2 (Z)**

Bit 2 is the zero flag, and is set to a one if the result of the previous operation was identically zero.

- **Bit 3 (N)**

Bit 3 is the negative flag, which contains exactly the value of the MSB of the result of the preceding operation. Thus, a negative two's-complement result will leave N set to a one.

- **Bit 4 (I)**

Bit 4 is the $\overline{\text{IRQ}}$ mask bit. The processor will not recognize interrupts from the $\overline{\text{IRQ}}$ line if this bit is set to a one. $\overline{\text{NMI}}$, $\overline{\text{FIRQ}}$, $\overline{\text{IRQ}}$, $\overline{\text{RES}}$ and SWI all set I to a one; SWI2 and SWI3 do not affect I.

- **Bit 5 (H)**

Bit 5 is the half-carry bit, and is used to indicate a carry from bit 3 in the ALU as a result of an 8-bit addition only (ADC or ADD). This bit is used by the DAA instruction to perform a BCD decimal add adjust operation. The state of this flag is undefined in all subtract-like instructions.

- **Bit 6 (F)**

Bit 6 is the $\overline{\text{FIRQ}}$ mask bit. The processor will not recognize interrupts from the $\overline{\text{FIRQ}}$ line if this bit is a one. $\overline{\text{NMI}}$, $\overline{\text{FIRQ}}$, SWI, and $\overline{\text{RES}}$ all set F to a one. $\overline{\text{IRQ}}$, SWI2 and SWI3 do not affect F.

- **Bit 7 (E)**

Bit 7 is the entire flag, and when set to a one indicates that the complete machine state (all the registers) was stacked, as opposed to the subset state (PC and CC). The E bit of the stacked CC is used on a return from interrupt (RTI) to determine the extent of the unstacking. Therefore, the current E left in the Condition Code Register represents past action.

■ HD6309E MPU SIGNAL DESCRIPTION

- **Power (Vss, Vcc)**

Two pins are used to supply powei to the part: Vss is ground or 0 volts, while Vcc is +5.0 V ±10%.

- **Address Bus ($A_0 \sim A_{15}$)**

Sixteen pins are used to output address information from the MPU onto the Address Bus. When the processor does not require the bus for a data transfer, it will output address $FFFF_{16}$, R/$\overline{\text{W}}$ = "High", and BS = "Low"; this is a "dummy access" or $\overline{\text{VMA}}$ cycle. All address bus drivers are made high-impedance when output Bus Available (BA) is "High" or when TSC is asserted. Each pin will drive one Schottky TTL load or four LS TTL loads, and 90 pF. Refer to Figures 1 and 2.

- **Data Bus ($D_0 \sim D_7$)**

These eight pins provide communication with the system bi-directional data bus. Each pin will drive one Schottky TTL load or four LS TTL loads, and 130 pF.

- **Read/Write (R/$\overline{\text{W}}$)**

This signal indicates the direction of data transfer on the data bus. A "Low" indicates that the MPU is writing data onto the data bus. R/$\overline{\text{W}}$ is made high impedance when BA is "High" or when TSC is asserted. Refer to Figures 1 and 2.

- **$\overline{\text{RES}}$**

A "Low" level on this Schmitt-trigger input for greater than one bus cycle will reset the MPU, as shown in Figure 7. The Reset vectors are fetched from locations $FFFE_{16}$ and $FFFF_{16}$ (Table 1) when Interrupt Acknowledge is true, ($\overline{\text{BA}}$ · BS = 1). During initial power-on, the Reset line should be held "Low" until the clock input signals are fully operational.

Because the HD6309E Reset pin has a Schmitt-trigger input with a threshold voltage higher than that of standard peripherals, a simple R/C network may be used to reset the entire system. This higher threshold voltage ensures that all peripherals are out of the reset state before the Processor.

Table 1 Memory Map for Interrupt Vectors

Memory Map for Vector Locations		Interrupt Vector Description
MS	LS	
FFFE	FFFF	$\overline{\text{RES}}$
FFFC	FFFD	$\overline{\text{NMI}}$
FFFA	FFFB	SWI
FFF8	FFF9	$\overline{\text{IRQ}}$
FFF6	FFF7	$\overline{\text{FIRQ}}$
FFF4	FFF5	SWI2
FFF2	FFF3	SWI3
FFF0	FFF1	Reserved

- **$\overline{\text{HALT}}$**

A "Low" level on this input pin will cause the MPU to stop running at the end of the present instruction and remain halted indefinitely without loss of data. When halted, the BA output is driven "High" indicating the buses are high impedance. BS is also "High" which indicates the processor is in the Halt state. While halted, the MPU will not respond to external real-time requests ($\overline{\text{FIRQ}}$, $\overline{\text{IRQ}}$) although $\overline{\text{NMI}}$ or $\overline{\text{RES}}$ will be latched for later response. During the Halt state Q and E should continue to run normally. A halted state (BA · BS = 1) can be achieved by pulling $\overline{\text{HALT}}$ "Low" while $\overline{\text{RES}}$ is still "Low". See Figure 8.

- **Bus Available, Bus Status (BA, BS)**

The Bus Available output is an indication of an internal control signal which makes the MOS buses of the MPU high impedance. When BA goes "Low", a dead cycle will elapse before the MPU acquires the bus. BA will not be asserted when TSC is active, thus allowing dead cycle consistency.

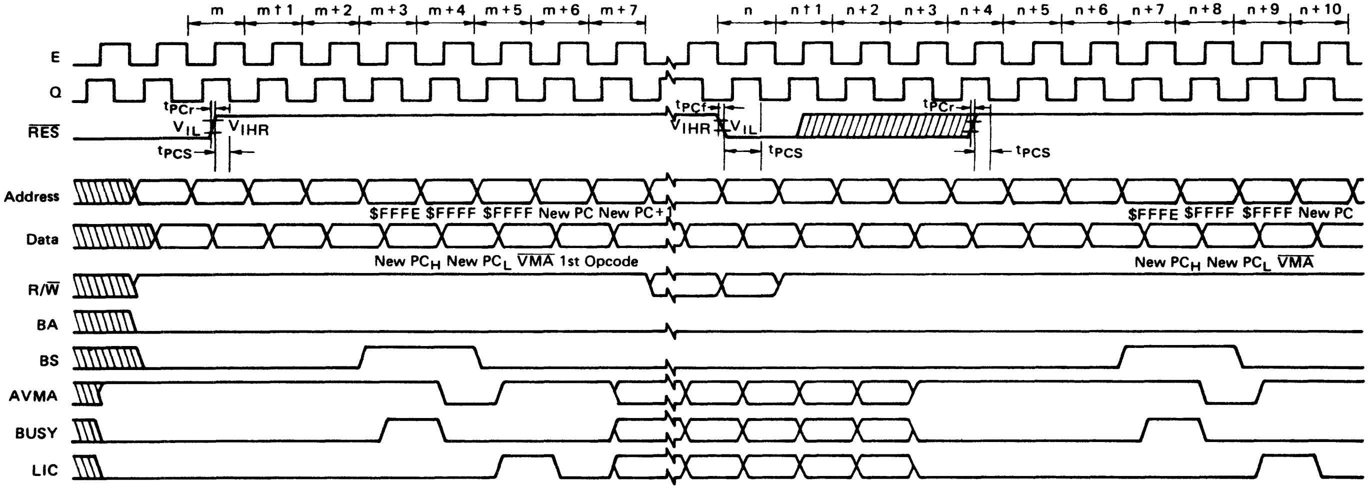
The Bus Status output signal, when decoded with BA, represents the MPU state (valid with leading edge of Q).

MPU State		MPU State Definition
BA	BS	
0	0	Normal (Running)
0	1	Interrupt or RESET Acknowledge
1	0	SYNC Acknowledge
1	1	HALT Acknowledge

Interrupt Acknowledge is indicated during both cycles of a hardware-vector-fetch ($\overline{\text{RES}}$, $\overline{\text{NMI}}$, $\overline{\text{FIRQ}}$, $\overline{\text{IRQ}}$, SWI, SWI2, SWI3). This signal, plus decoding of the lower four address lines, can provide the user with an indication of which interrupt level is being serviced and allow vectoring by device. See Table 1.

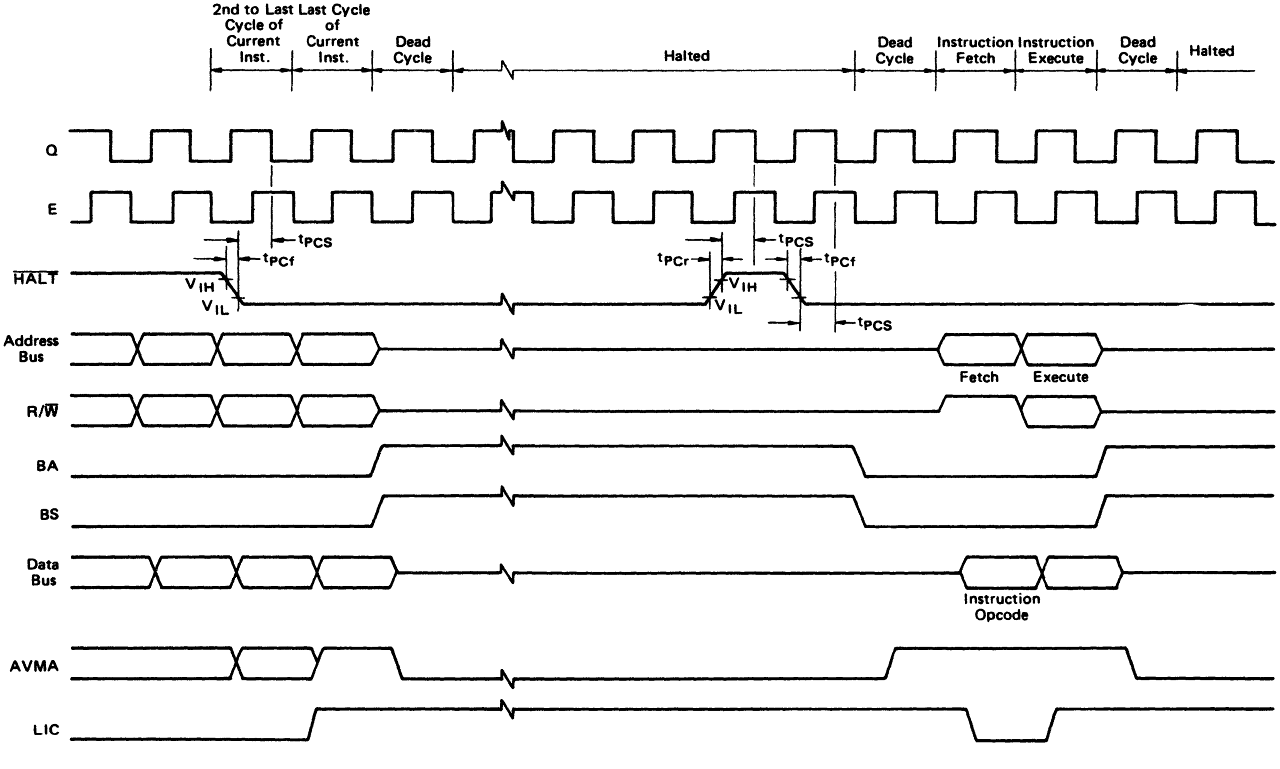
Sync Acknowledge is indicated while the MPU is waiting for external synchronization on an interrupt line.

Halt Acknowledge is indicated when the HD6309E is in a Halt condition.



(NOTE) Waveform measurements for all inputs and outputs are specified at logic "High" = V_{IHmin} and logic "Low" = V_{ILmax} unless otherwise specified.

Figure 7 $\overline{RES}$ Timing



(NOTE) Waveform measurements for all inputs and outputs are specified at logic "High" = V_{IHmin} and logic "Low" = V_{ILmax} unless otherwise specified.

Figure 8 $\overline{\text{HALT}}$ and Single Instruction Execution for System Debug

- **Non Maskable Interrupt ($\overline{\text{NMI}}$)***

A negative transition on this input requests that a non-maskable interrupt sequence be generated. A non-maskable interrupt cannot be inhibited by the program, and also has a higher priority than $\overline{\text{FIRQ}}$, $\overline{\text{IRQ}}$ or software interrupts. During recognition of an $\overline{\text{NMI}}$, the entire machine state is saved on the hardware stack. After reset, an $\overline{\text{NMI}}$ will not be recognized until the first program load of the Hardware Stack Pointer (S). The pulse width of $\overline{\text{NMI}}$ low must be at least one E cycle. If the $\overline{\text{NMI}}$ input does not meet the minimum set up with respect to Q, the interrupt will not be recognized until the next cycle. See Figure 9.

- **Fast-Interrupt Request ($\overline{\text{FIRQ}}$)***

A "Low" level on this input pin will initiate a fast interrupt sequence, provided its mask bit (F) in the CC is clear. This sequence has priority over the standard Interrupt Request ($\overline{\text{IRQ}}$), and is fast in the sense that it stacks only the contents of the condition code register and the program counter. The interrupt service routine should clear the source of the interrupt before doing an RTI. See Figure 10.

- **Interrupt Request ($\overline{\text{IRQ}}$)***

A "Low" level input on this pin will initiate an Interrupt Request sequence provided the mask bit (I) in the CC is clear. Since $\overline{\text{IRQ}}$ stacks the entire machine state it provides a slower response to interrupts than $\overline{\text{FIRQ}}$. $\overline{\text{IRQ}}$ also has a lower priority than $\overline{\text{FIRQ}}$. Again, the interrupt service routine should clear the source of the interrupt before doing an RTI. See Figure 9.

- **Clock Inputs E, Q**

E and Q are the clock signals required by the HD6309E. Q must lead E; that is, a transition on Q must be followed by a similar transition on E after a minimum delay. Addresses will be valid from the MPU, t_{AD} after the falling edge of E, and data will be latched from the bus by the falling edge of E. While the Q input is fully TTL compatible, the E input directly drives internal MOS circuitry and, thus, requires levels above normal TTL levels. This approach minimizes clock skew inherent with an internal buffer. Timing and waveforms for E and Q are shown in Figures 1 and 2 while Figure 11 shows a simple clock generator for the HD6309E.

- **BUSY**

Busy will be "High" for the read and modify cycles of a read-modify-write instruction and during the access of the first byte of a double-byte operation (e.g., LDX, STD, ADDD). Busy is also "High" during the first byte of any indirect or other vector fetch (e.g., jump extended, SWI indirect etc.).

In a multi-processor system, busy indicates the need to defer the rearbitration of the next bus cycle to insure the integrity of the above operations. This difference provides the indivisible memory access required for a "test-and-set" primitive, using any one of several read-modify-write instructions.

Busy does not become active during PSH or PUL operations. A typical read-modify-write instruction (ASL) is shown in Figure 12. Timing information is given in Figure 13. Busy is valid t_{CD} after the rising edge of Q.

- **AVMA**

AVMA is the Advanced VMA signal and indicates that the MPU will use the bus in the following bus cycle. The predictive nature of the AVMA signal allows efficient shared-bus multiprocessor systems. AVMA is "Low" when the MPU is in either a HALT or SYNC state. AVMA is valid t_{CD} after the rising edge of Q.

- **LIC**

LIC (Last Instruction Cycle) is "High" during the last cycle of every instruction, and its transition from "High" to "Low" will indicate that the first byte of an opcode will be latched at the end of the present bus cycle. LIC will be "High" when the MPU is Halted at the end of an instruction, (i.e., not in CWAI or RESET) in SYNC state or while stacking during interrupts. LIC is valid t_{CD} after the rising edge of Q.

- **TSC**

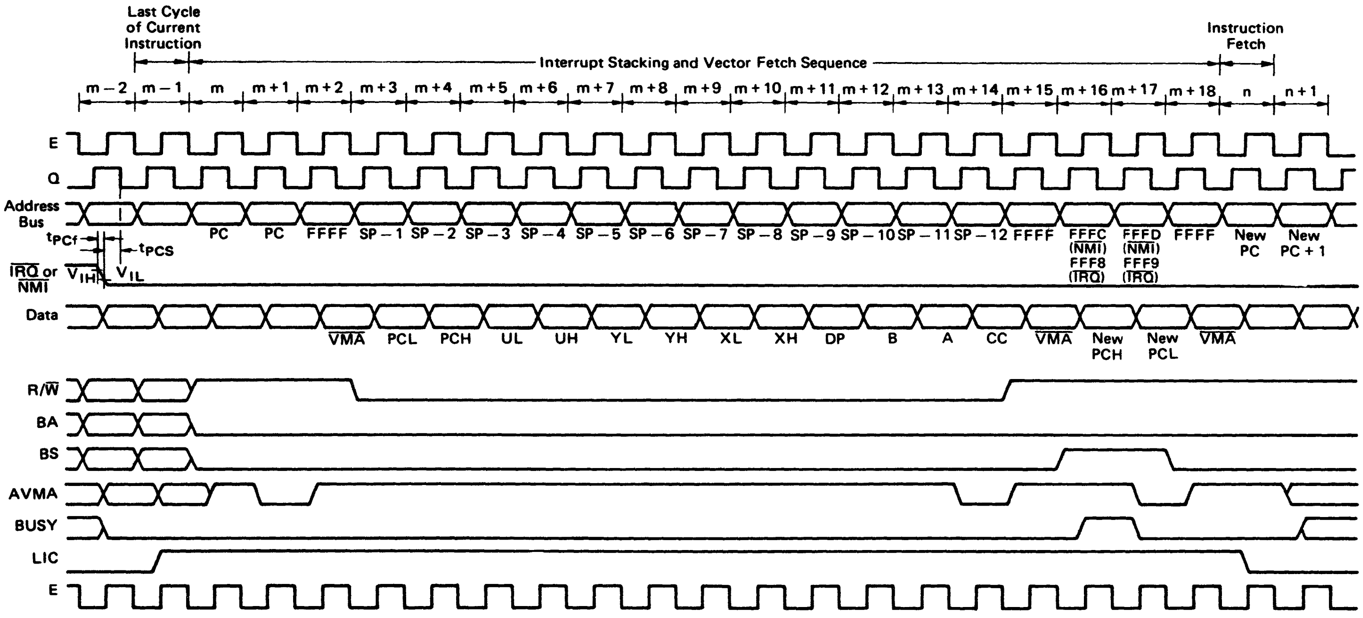
TSC (Three-State Control) will cause MOS address, data, and R/$\overline{\text{W}}$ buffers to assume a high-impedance state. The control signals (BA, BS, BUSY, AVMA and LIC) will not go to the high-impedance state. TSC is intended to allow a single bus to be shared with other bus masters (processors or DMA controllers).

While E is "Low", TSC controls the address buffers and R/$\overline{\text{W}}$ directly. The data bus buffers during a write operation are in a high-impedance state until Q rises at which time, if TSC is true, they will remain in a high-impedance state. If TSC is held beyond the rising edge of E, then it will be internally latched, keeping the bus drivers in a high-impedance state for the remainder of the bus cycle. See Figure 14.

- **MPU Operation**

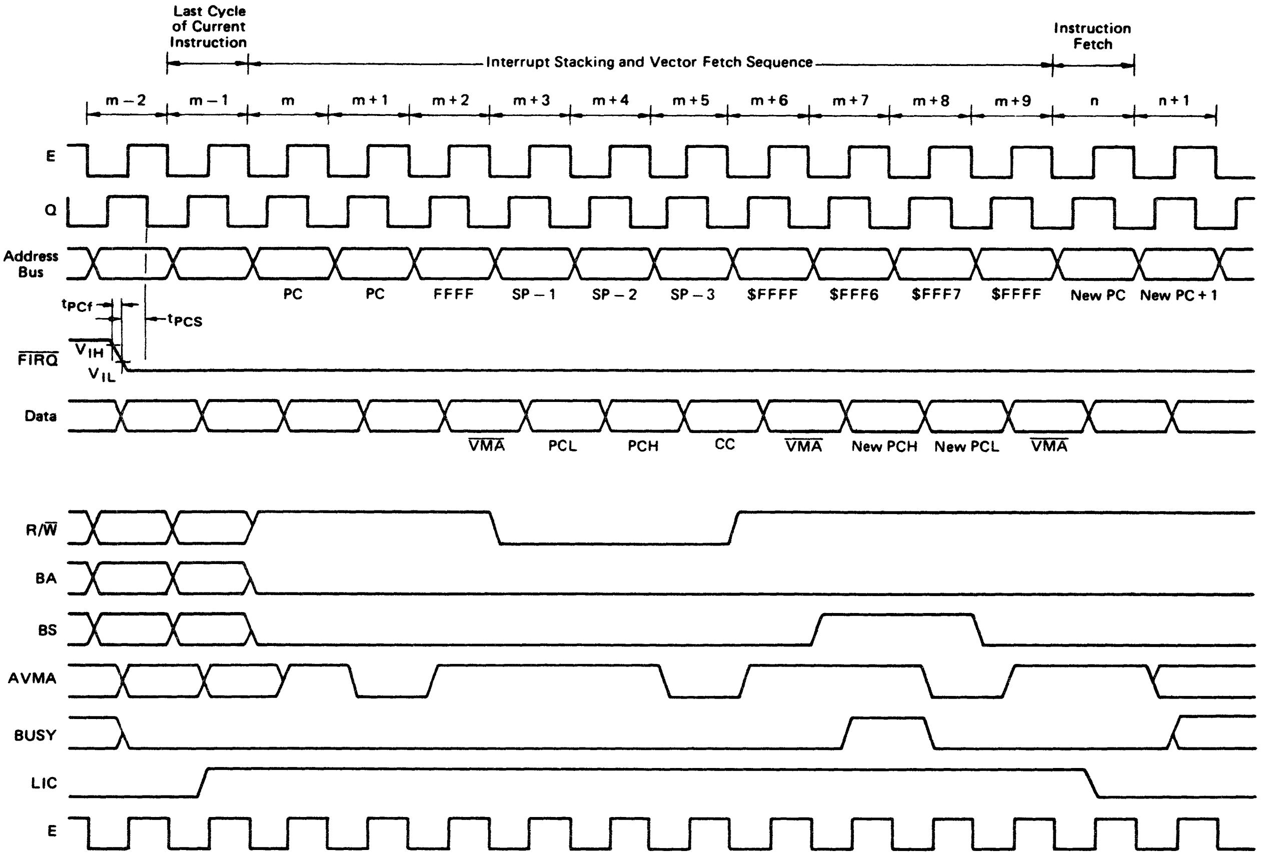
During normal operation, the MPU fetches an instruction from memory and then executes the requested function. This sequence begins after $\overline{\text{RES}}$ and is repeated indefinitely unless altered by a special instruction or hardware occurrence. Software instructions that alter normal MPU operation are: SWI, SWI2, SWI3, CWAI, RTI and SYNC. An interrupt or $\overline{\text{HALT}}$ input can also alter the normal execution of instructions. Figure 15 illustrates the flow chart for the HD6309E.

* $\overline{\text{NMI}}$, $\overline{\text{FIRQ}}$, and $\overline{\text{IRQ}}$ requests are sampled on the falling edge of Q. One cycle is required for synchronization before these interrupts are recognized. The pending interrupt(s) will not be serviced until completion of the current instruction unless a SYNC or CWAI condition is present. If $\overline{\text{IRQ}}$ and $\overline{\text{FIRQ}}$ do not remain "Low" until completion of the current instiuction they may not be recognized. However, $\overline{\text{NMI}}$ is latched and need only remain "Low" for one cycle.



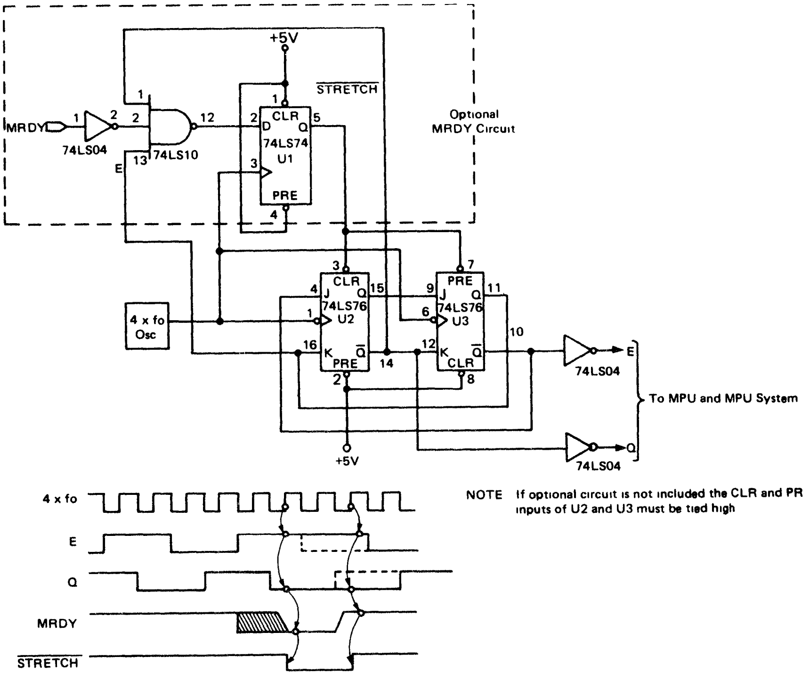
(NOTE) Waveform measurements for all inputs and outputs are specified at logic "High" = V_{IHmin} and logic "Low" = V_{ILmax} unless otherwise specified. E clock shown for reference only.

Figure 9 IRQ and NMI Interrupt Timing



(NOTE) Waveform measurements for all inputs and outputs are specified at logic "High" = V_{IHmin} and logic "Low" = V_{ILmax} unless otherwise specified.
E clock shown for reference only.

Figure 10 $\overline{\text{FIRQ}}$ Interrupt Timing

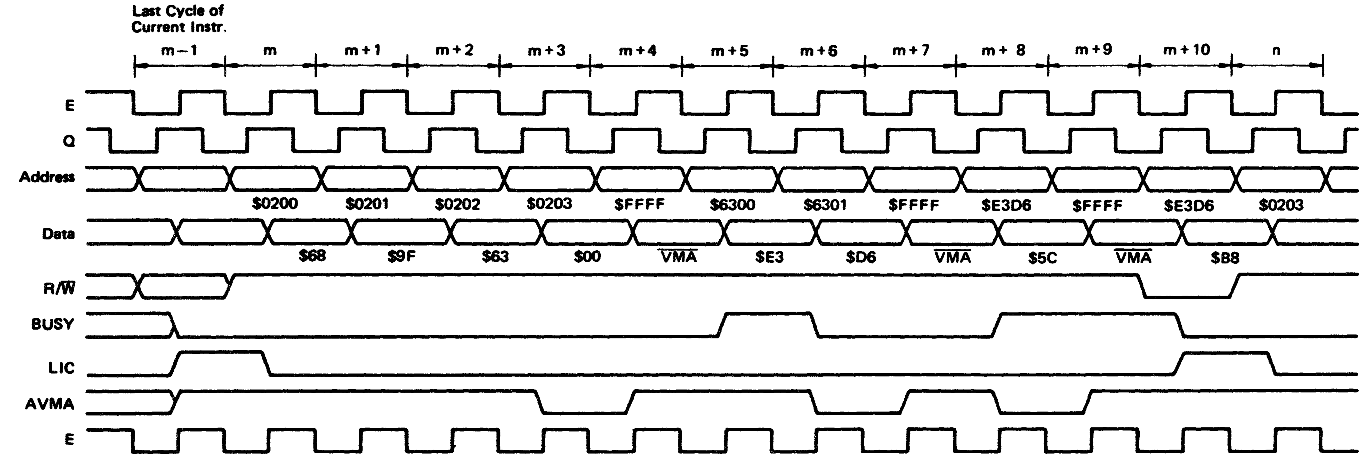


NOTE If optional circuit is not included the CLR and PRE inputs of U2 and U3 must be tied high

Figure 11 HD6309E Clock Generator

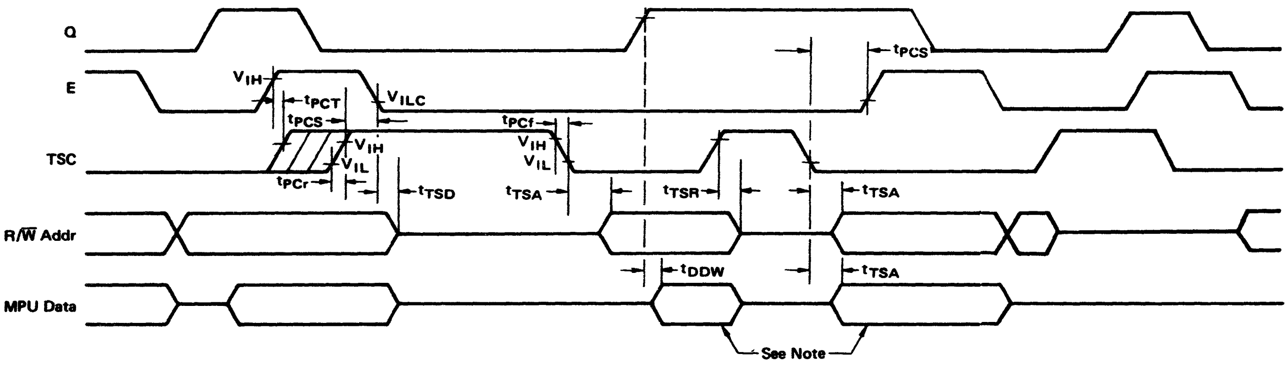
Memory Location	Memory Contents	Contents Description
PC → $0200	$68	ASL Indexed Opcode
$0201	$9F	Extended Indirect Postbyte
$0202	$63	Indirect Address Hi-Byte
$0203	$00	Indirect Address Lo-Byte
$0204		Next Main Instruction
$6300	$E3	Effective Address Hi-Byte
$6301	$D6	Effective Address Lo-Byte
$E3D6	$5C	Target Data

Figure 12 Read Modify Write Instruction Example (ASL Extended Indirect)



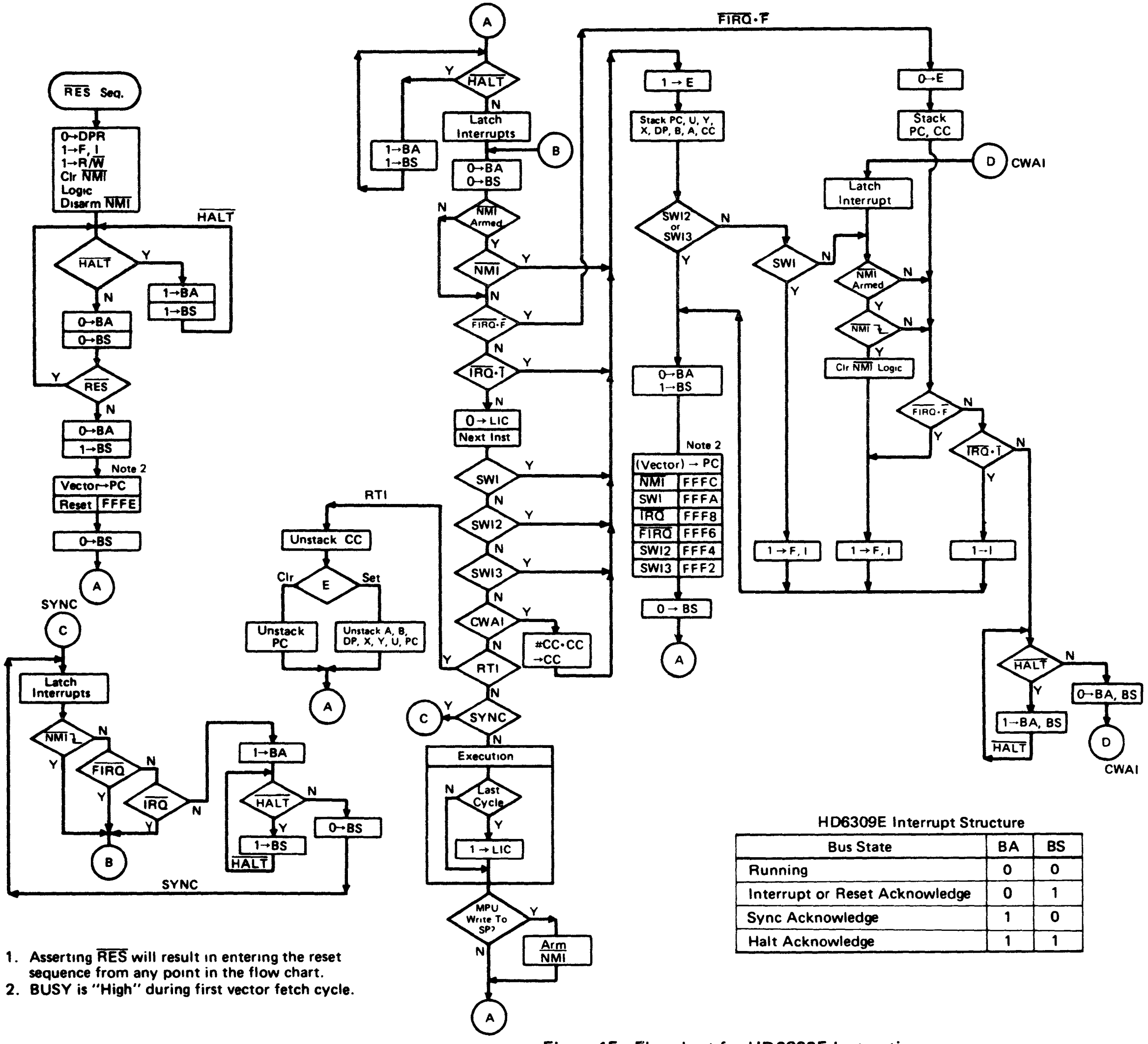
(NOTE) Waveform measurements for all inputs and outputs are specified at logic "High" = V_{IHmin} and logic "Low" = V_{ILmax} unless otherwise specified.

Figure 13 BUSY Timing (ASL Extended Indirect Instruction)



(NOTES) Data will be asserted by the MPU only during the interval while R/W̄ is "Low" and E or Q is "High".
Waveform measurements for all inputs and outputs are specified at logic "High" = V_{IHmin} and logic "Low" = V_{ILmax} unless otherwise specified.

Figure 14 TSC Timing



HD6309E Interrupt Structure

Bus State	BA	BS
Running	0	0
Interrupt or Reset Acknowledge	0	1
Sync Acknowledge	1	0
Halt Acknowledge	1	1

(NOTES) 1. Asserting RES will result in entering the reset sequence from any point in the flow chart.
2. BUSY is "High" during first vector fetch cycle.

Figure 15 Flowchart for HD6309E Instruction

■ ADDRESSING MODES

The basic instructions of any computer are greatly enhanced by the presence of powerful addressing modes. The HD6309E has the most complete set of addressing modes available on any microcomputer today. For example, the HD6309E has 59 basic instructions; however, it recognizes 1464 different variations of instructions and addressing modes. The addressing modes support modern programming techniques. The following addressing modes are available on the HD6309E:

(1) Implied (Includes Accumulator)
(2) Immediate
(3) Extended
(4) Extended Indirect
(5) Direct
(6) Register
(7) Indexed
  Zero-Offset
  Constant Offset
  Accumulator Offset
  Auto Increment/Decrement
(8) Indexed Indirect
(9) Relative
(10) Program Counter Relative

● Implied (Includes Accumulator)

In this addressing mode, the opcode of the instruction contains all the address information necessary. Examples of Implied Addressing are: ABX, DAA, SWI, ASRA, and CLRB.

● Immediate Addressing

In Immediate Addressing, the effective address of the data is the location immediately following the opcode (i.e., the data to be used in the instruction immediately follows the opcode of the instruction). The HD6309E uses both 8 and 16-bit immediate values depending on the size of argument specified by the opcode. Examples of instructions with Immediate Addressing are:

```
LDA  #$20
LDX  #$F000
LDY  #CAT
```

(NOTE) # signifies immediate addressing, $ signifies hexadecimal value.

● Extended Addressing

In Extended Addressing, the contents of the two bytes immediately following the opcode fully specify the 16-bit effective address used by the instruction. Note that the address generated by an extended instruction defines an absolute address and is not position independent. Examples of Extended Addressing include:

```
LDA  CAT
STX  MOUSE
LDD  $2000
```

● Extended Indirect

As a special case of indexed addressing (discussed below), one level of indirection may be added to Extended Addressing. In Extended Indirect, the two bytes following the postbyte of an Indexed instruction contain the address of the data.

```
LDA  [CAT]
LDX  [$FFFE]
STU  [DOG]
```

● Direct Addressing

Direct addressing is similar to extended addressing except that only one byte of address follows the opcode. This byte specifies the lower 8 bits of the address to be used. The upper 8 bits of the addres‚ are supplied by the direct page register. Since only one byte of address is required in direct addressing, this mode requires less memory and executes faster than extended addressing. Of course, only 256 locations (one page) can be accessed without redefining the contents of the DP register. Since the DP register is set to $00 on Reset, direct addressing on the HD6309E is compatible with direct addressing on the HD6800. Indirection is not allowed in direct addressing. Some examples of direct addressing are:

```
LDA    $30
SETDP  $10    (Assembler directive)
LDB    $1030
LDD    <CAT
```

(NOTE) < is an assembler directive which forces direct addressing.

● Register Addressing

Some opcodes are followed by a byte that defines a register or set of registers to be used by the instruction. This is called a postbyte. Some examples of register addressing are:

```
TFR   X, Y        Transfer X into Y
EXG   A, B        Exchanges A with B
PSHS  A, B, X, Y  Push Y, X, B and A onto S
PULU  X, Y, D     Pull D, X, and Y from U
```

● Indexed Addressing

In all indexed addressing, one of the pointer registers (X, Y, U, S, and sometimes PC) is used in a calculation of the effective address of the operand to be used by the instruction. Five basic types of indexing are available and are discussed below. The postbyte of an indexed instruction specifies the basic type and variation of the addressing mode as well as the pointer register to be used. Figure 16 lists the legal formats for the postbyte. Table 2 gives the assembler form and the number of cycles and bytes added to the basic values for indexed addressing for each variation.

Post-Byte Register Bit								Indexed Addressing Mode
7	6	5	4	3	2	1	0	
0	R	R	d	d	d	d	d	EA = ,R + 5 Bit Offset
1	R	R	0	0	0	0	0	,R +
1	R	R	i	0	0	0	1	,R + +
1	R	R	0	0	0	1	0	, -R
1	R	R	i	0	0	1	1	, - - R
1	R	R	i	0	1	0	0	EA = ,R + 0 Offset
1	R	R	i	0	1	0	1	EA = ,R + ACCB Offset
1	R	R	i	0	1	1	0	EA = ,R + ACCA Offset
1	R	R	i	1	0	0	0	EA = , R + 8 Bit Offset
1	R	R	i	1	0	0	1	EA = ,R + 16 Bit Offset
1	R	R	i	1	0	1	1	EA = ,R + D Offset
1	x	x	i	1	1	0	0	EA = ,PC + 8 Bit Offset
1	x	x	i	1	1	0	1	EA = ,PC + 16 Bit Offset
1	R	R	i	1	1	1	1	EA = [,Address]

Addressing Mode Field

Indirect Field
(Sigh bit when b7 = 0)

Register Field . RR
00 = X
01 = Y
10 = U
11 = S

x = Don't Care
d = Offset Bit
i = { 0 = Non Indirect
  1 = Indirect

Figure 16 Index Addressing Postbyte Register Bit Assignments

Table 2 Indexed Addressing Mode

Type	Forms	Non Indirect				Indirect			
		Assembler Form	Postbyte OP Code	$\overset{+}{\sim}$	$\overset{+}{\#}$	Assembler Form	Postbyte OP Code	$\overset{+}{\sim}$	$\overset{+}{\#}$
Constant Offset From R (2's Complement Offsets)	No Offset	,R	1RR00100	0	0	[,R]	1RR10100	3	0
	5 Bit Offset	n, R	0RRnnnnn	1	0	defaults to 8-bit			
	8 Bit Offset	n, R	1RR01000	1	1	[n, R]	1RR11000	4	1
	16 Bit Offset	n, R	1RR01001	4	2	[n, R]	1RR11001	7	2
Accumulator Offset From R (2's Complement Offsets)	A Register Offset	A, R	1RR00110	1	0	[A, R]	1RR10110	4	0
	B Register Offset	B, R	1RR00101	1	0	[B, R]	1RR10101	4	0
	D Register Offset	D, R	1RR01011	4	0	[D, R]	1RR11011	7	0
Auto Increment/Decrement R	Increment By 1	,R +	1RR00000	2	0	not allowed			
	Increment By 2	,R + +	1RR00001	3	0	[,R + +]	1RR10001	6	0
	Decrement By 1	, - R	1RR00010	2	0	not allowed			
	Decrement By 2	, - - R	1RR00011	3	0	[, - - R]	1RR10011	6	0
Constant Offset From PC (2's Complement Offsets)	8 Bit Offset	n, PCR	1xx01100	1	1	[n, PCR]	1xx11100	4	1
	16 Bit Offset	n, PCR	1xx01101	5	2	[n, PCR]	1xx11101	8	2
Extended Indirect	16 Bit Address	—	—	—	—	[n]	10011111	5	2

R = X, Y, U or S   RR:
x = Don't Care   00 = X
01 = Y
10 = U
11 = S

$\overset{+}{\sim}$ and $\overset{+}{\#}$ indicate the number of additional cycles and bytes for the particular variation.

Zero-Offset Indexed

In this mode, the selected pointer register contains the effective address of the data to be used by the instruction. This is the fastest indexing mode.

Examples are:

```
LDD   0, X
LDA   S
```

Constant Offset Indexed

In this mode, a two's-complement offset and the contents of one of the pointer registers are added to form the effective address of the operand. The pointer register's initial content is unchanged by the addition.

Three sizes of offsets are available:
5-bit (−16 to +15)
8-bit (−128 to +127)
16-bit (−32768 to +32767)

The two's complement 5-bit offset is included in the postbyte and, therefore, is most efficient in use of bytes and cycles. The two's complement 8-bit offset is contained in a single byte following the postbyte. The two's complement 16-bit offset is in the two bytes following the postbyte. In most cases the programmer need not be concerned with the size of this offset since the assembler will select the optimal size automatically.

Examples of constant-offset indexing are:

```
LDA   23, X
LDX   −2, S
LDY   300, X
LDU   CAT, Y
```

Accumulator-Offset Indexed

This mode is similar to constant offset indexed except that the two's-complement value in one of the accumulators (A, B or D) and the contents of one of the pointer registers are added to form the effective address of the operand. The contents of both the accumulator and the pointer register are unchanged by the addition. The postbyte specifies which accumulator to use as an offset and no additional bytes are required. The advantage of an accumulator offset is that the value of the offset can be calculated by a program at run-time.

Some examples are:

```
LDA    B, Y
LDX    D, Y
LEAX   B, X
```

Auto Increment/Decrement Indexed

In the auto increment addressing mode, the pointer register contains the address of the operand. Then, after the pointer register is used it is incremented by one or two. This addressing mode is useful in stepping through tables, moving data, or for the creation of software stacks. In auto decrement, the pointer register is decremented prior to use as the address of the data. The use of auto decrement is similar to that of auto increment; but the tables, etc., are scanned from the high to low addresses. The size of the increment/decrement can be either one or two to allow for tables of either 8- or 16-bit data to be accessed and is selectable by the programmer. The predecrement, post-increment nature of these modes allow them to be used to create additional software stacks that behave identically to the U and S stacks.

Some examples of the auto increment/decrement addressing modes are:

```
LDA   ,X+
STD   ,Y++
LDB   ,-Y
LDX   ,--S
```

Care should be taken in performing operations on 16-bit pointer registers (X, Y, U, S) where the same register is used to calculate the effective address.

Consider the following instruction:

STX 0, X + + (X initialized to 0)

The desired result is to store a 0 in locations $0000 and $0001 then increment X to point to $0002. In reality, the following occurs:

0 → temp	calculate the EA; temp is a holding register
X + 2 → X	perform autoincrement
X → (temp)	do store operation

- **Indexed Indirect**

All of the indexing modes with the exception of auto increment/decrement by one, or a ±4-bit offset may have an additional level of indirection specified. In indirect addressing, the effective address is contained at the location specified by the contents of the Index Register plus any offset. In the example below, the A accumulator is loaded indirectly using an effective address calculated from the Index Register and an offset.

```
        Before Execution
        A = XX (don't care)
        X = $F000
$0100   LDA [$10, X]        EA is now $F010
$F010   $F1                 $F150 is now the
$F011   $50                 new EA
$F150   $AA
        After Execution
        A = $AA (Actual Data Loaded)
        X = $F000
```

All modes of indexed indirect are included except those which are meaningless (e.g., auto increment/decrement by 1 indirect). Some examples of indexed indirect are:

```
LDA   [,X]
LDD   [10,S]
LDA   [B,Y]
LDD   [,X++]
```

- **Relative Addressing**

The byte(s) following the branch opcode is (are) treated as a signed offset which may be added to the program counter. If the branch condition is true then the calculated address (PC + signed offset) is loaded into the program counter. Program execution continues at the new location as indicated by the PC; short (1 byte offset) and long (2 bytes offset) relative addressing modes are available. All of memory can be reached in long relative addressing as an effective address is interpreted modulo 2^{16}. Some examples of relative addressing are:

```
        BEQ     CAT     (short)
        BGT     DOG     (short)
CAT     LBEQ    RAT     (long)
DOG     LBGT    RABBIT  (long)
        •
        •
        •
RAT     NOP
RABBIT  NOP
```

- **Program Counter Relative**

The PC can be used as the pointer register with 8 or 16-bit signed offsets. As in relative addressing, the offset is added to the current PC to create the effective address. The effective address is then used as the address of the operand or data. Program Counter Relative Addressing is used for writing position independent programs. Tables related to a particular routine will maintain the same relationship after the routine is moved, if referenced relative to the Program Counter. Examples are:

```
LDA    CAT, PCR
LEAX   TABLE, PCR
```

Since program counter relative is a type of indexing, an additional level of indirection is available.

```
LDA    [CAT, PCR]
LDU    [DOG, PCR]
```

▪ HD6309E INSTRUCTION SET

The instruction set of the HD6309E is similar to that of the HD6800 and is upward compatible at the source code level. The number of opcodes has been reduced from 72 to 59, but because of the expanded architecture and additional addressing modes, the number of available opcodes (with different addressing modes) has risen from 197 to 1464.

Some of the instructions are described in detail below:

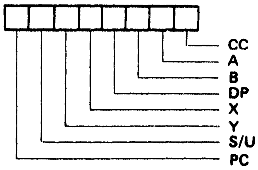
- **PSHU/PSHS**

The push instructions have the capability of pushing onto either the hardware stack (S) or user stack (U) any single register, or set of registers with a single instruction.

- **PULU/PULS**

The pull instructions have the same capability of the push instruction, in reverse order. The byte immediately following the push or pull opcode determines which register or registers are to be pushed or pulled. The actual PUSH/PULL sequence is fixed; each bit defines a unique register to push or pull, as shown in below.

PUSH/PULL POST BYTE



```
          ← Pull Order          Push Order →
PC        U    Y    X    DP   B    A    CC
FFFF ....... ← increasing memory address ....... 0000
PC        S    Y    X    DP   B    A    CC
```

- **TFR/EXG**

Within the HD6309E, any register may be transferred to or exchanged with another of like-size; i.e., 8-bit to 8-bit or 16-bit to 16-bit. Bits 4~7 of postbyte define the source register, while bits 0~3 represent the destination register. These are denoted as follows:

0000 – D	0101 – PC
0001 – X	1000 – A
0010 – Y	1001 – B
0011 – U	1010 – CC
0100 – S	1011 – DP

(NOTE) All other combinations are undefined and INVALID.

TRANSFER/EXCHANGE POST BYTE

SOURCE
DESTINATION

- **LEAX/LEAY/LEAU/LEAS**

The LEA (Load Effective Address) works by calculating the effective address used in an indexed instruction and stores that address value, rather than the data at that address, in a pointer register. This makes all the features of the internal addressing hardware available to the programmer. Some of the implications of this instruction are illustrated in Table 3.

The LEA instruction also allows the user to access data in a position independent manner. For example:

```
          LEAX     MSG1, PCR
          LBSR     PDATA (Print message routine)
           •
           •
MSG1      FCC      'MESSAGE'
```

This sample program prints: 'MESSAGE'. By writing MSG1, PCR, the assembler computes the distance between the present address and MSG1. This result is placed as a constant into the LEAX instruction which will be indexed from the PC value at the time of execution. No matter where the code is located, when it is executed, the computed offset from the PC will put the absolute address of MSG1 into the X pointer register. This code is totally position independent.

The LEA instructions are very powerful and use an internal holding register (temp). Care must be exercised when using the LEA instructions with the autoincrement and autodecrement addressing modes due to the sequence of internal operations. The LEA internal sequence is outlined as follows:

LEAa, b+ (any of the 16-bit pointer registers X, Y, U or S may be substituted for a and b.)
1. b → temp (calculate the EA)
2. b + 1 → b (modify b, postincrement)
3. temp → a (load a)

LEAa, – b
1. b – 1 → temp (calculate EA with predecrement)
2. b – 1 → b (modify b, predecrement)
3. temp → a (load a)

Autoincrement-by-two and autodecrement-by-two instructions work similarly. Note that LEAX, X+ does not change X, however LEAX, –X does decrement X. LEAX 1, X should be used to increment X by one.

Table 3 LEA Examples

Instruction	Operation	Comment
LEAX 10, X	X + 10 → X	Adds 5-bit constant 10 to X
LEAX 500, X	X + 500 → X	Adds 16-bit constant 500 to X
LEAY A, Y	Y + A → Y	Adds 8-bit A accumulator to Y
LEAY D, Y	Y + D → Y	Adds 16-bit D accumulator to Y
LEAU –10, U	U – 10 → U	Subtracts 10 from U
LEAS –10, S	S – 10 → S	Used to reserve area on stack
LEAS 10, S	S + 10 → S	Used to 'clean up' stack
LEAX 5, S	S + 5 → X	Transfers as well as adds

- **MUL**

Multiplies the unsigned binary numbers in the A and B accumulator and places the unsigned result into the 16-bit D accumulator. This unsigned multiply also allows multiple-precision multiplications.

Long and Short Relative Branches

The HD6309E has the capability of program counter relative branching throughout the entire memory map. In this mode, if the branch is to be taken, the 8 or 16-bit signed offset is added to the value of the program counter to be used as the effective address. This allows the program to branch anywhere in the 64k memory map. Position independent code can be easily generated through the use of relative branching. Both short (8-bit) and long (16-bit) branches are available.

- **SYNC**

After encountering a Sync instruction, the MPU enters a Sync state, stops processing instructions and waits for an interrupt. If the pending interrupt is non-maskable ($\overline{\text{NMI}}$) or maskable ($\overline{\text{FIRQ}}$, $\overline{\text{IRQ}}$) with its mask bit (F or I) clear, the processor will clear the Sync state and perform the normal interrupt stacking and service routine. Since $\overline{\text{FIRQ}}$ and $\overline{\text{IRQ}}$ are not edge-triggered, a low level with a minimum duration of three bus cycles is required to assure that the interrupt will be taken. If the pending interrupt is maskable ($\overline{\text{FIRQ}}$, $\overline{\text{IRQ}}$) with its mask bit (F or I) set, the processor will clear the Sync state and continue processing by executing the next inline instruction. Figure 17 depicts Sync timing.

Software Interrupts

A Software Interrupt is an instruction which will cause an interrupt, and its associated vector fetch. These Software Interrupts are useful in operating system calls, software debugging, trace operations, memory mapping, and software development systems. Three levels of SWI are available on this HD6309E, and are prioritized in the following order: SWI, SWI2, SWI3.

16-Bit Operation

The HD6309E has the capability of processing 16-bit data. These instructions include loads, stores, compares, adds, subtracts, transfers, exchanges, pushes and pulls.

- **CYCLE-BY-CYCLE OPERATION**

The address bus cycle-by-cycle performance chart illustrates the memory-access sequence corresponding to each possible instruction and addressing mode in the HD6309E. Each instruction begins with an opcode fetch. While that opcode is being internally decoded, the next program byte is always fetched. (Most instructions will use the next byte, so this technique considerably speeds throughput.) Next, the operation of each opcode will follow the flow chart. $\overline{\text{VMA}}$ is an indication of FFFF_{16} on the address bus, R/$\overline{\text{W}}$ = "High" and BS = "Low". The following examples illustrate the use of the chart; see Figure 18.

Example 1: LBSR (Branch Taken)
Before Execution SP = F000

```
                    •
                    •
                    •
$8000               LBSR        CAT
                    •
                    •
                    •
$A000     CAT       •
```

CYCLE-BY-CYCLE FLOW

Cycle #	Address	Data	R/$\overline{\text{W}}$	Description
1	8000	17	1	Opcode Fetch
2	8001	1F	1	Offset High Byte
3	8002	FD	1	Offset Low Byte
4	FFFF	*	1	$\overline{\text{VMA}}$ Cycle
5	FFFF	*	1	$\overline{\text{VMA}}$ Cycle
6	FFFF	*	1	$\overline{\text{VMA}}$ Cycle
7	FFFF	*	1	$\overline{\text{VMA}}$ Cycle
8	EFFF	03	0	Stack Low Order Byte of Return Address
9	EFFE	80	0	Stack High Order Byte of Return Address

Example 2: DEC (Extended)

```
$8000        DEC          $A000
$A000        FCB          $80
```

CYCLE-BY-CYCLE FLOW

Cycle #	Address	Data	R/$\overline{\text{W}}$	Description
1	8000	7A	1	Opcode Fetch
2	8001	A0	1	Operand Address, High Byte
3	8002	00	1	Operand Address, Low Byte
4	FFFF	*	1	$\overline{\text{VMA}}$ Cycle
5	A000	80	1	Read the Data
6	FFFF	*	1	$\overline{\text{VMA}}$ Cycle
7	A000	7F	0	Store the Decremented Data

* The data bus has the data at that particular address.

■ SLEEP MODE

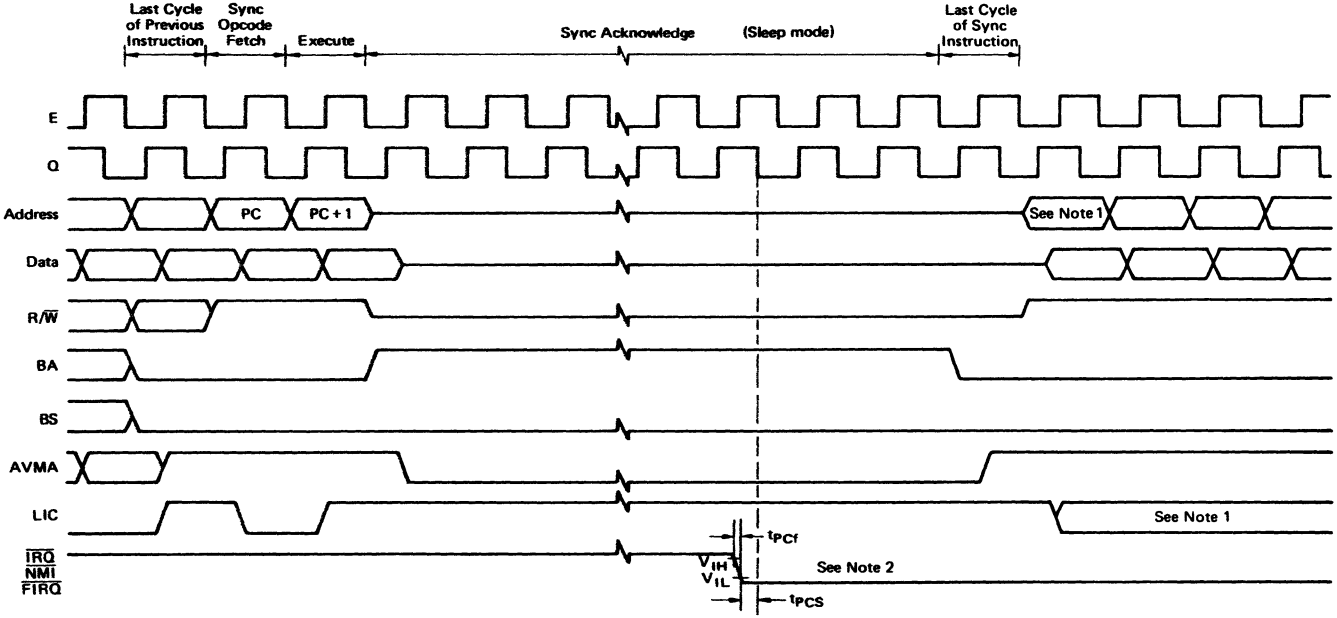
During the interrupt wait period in the SYNC instruction (the SYNC state) and that period in the CWAI instruction (the WAIT state), MPU operation is halted and goes to the sleep mode. However, the state of I/O pins is the same as that of the HD6809E in this mode.

■ HD6309E INSTRUCTION SET TABLES

The instructions of the HD6309E have been broken down into five different categories. They are as follows:

8-Bit operation (Table 4)
16-Bit operation (Table 5)
Index register/stack pointer instructions (Table 6)
Relative branches (long or short) (Table 7)
Miscellaneous instructions (Table 8)

HD6309E instruction set tables and Hexadecimal Values of instructions are shown in Table 9 and Table 10.

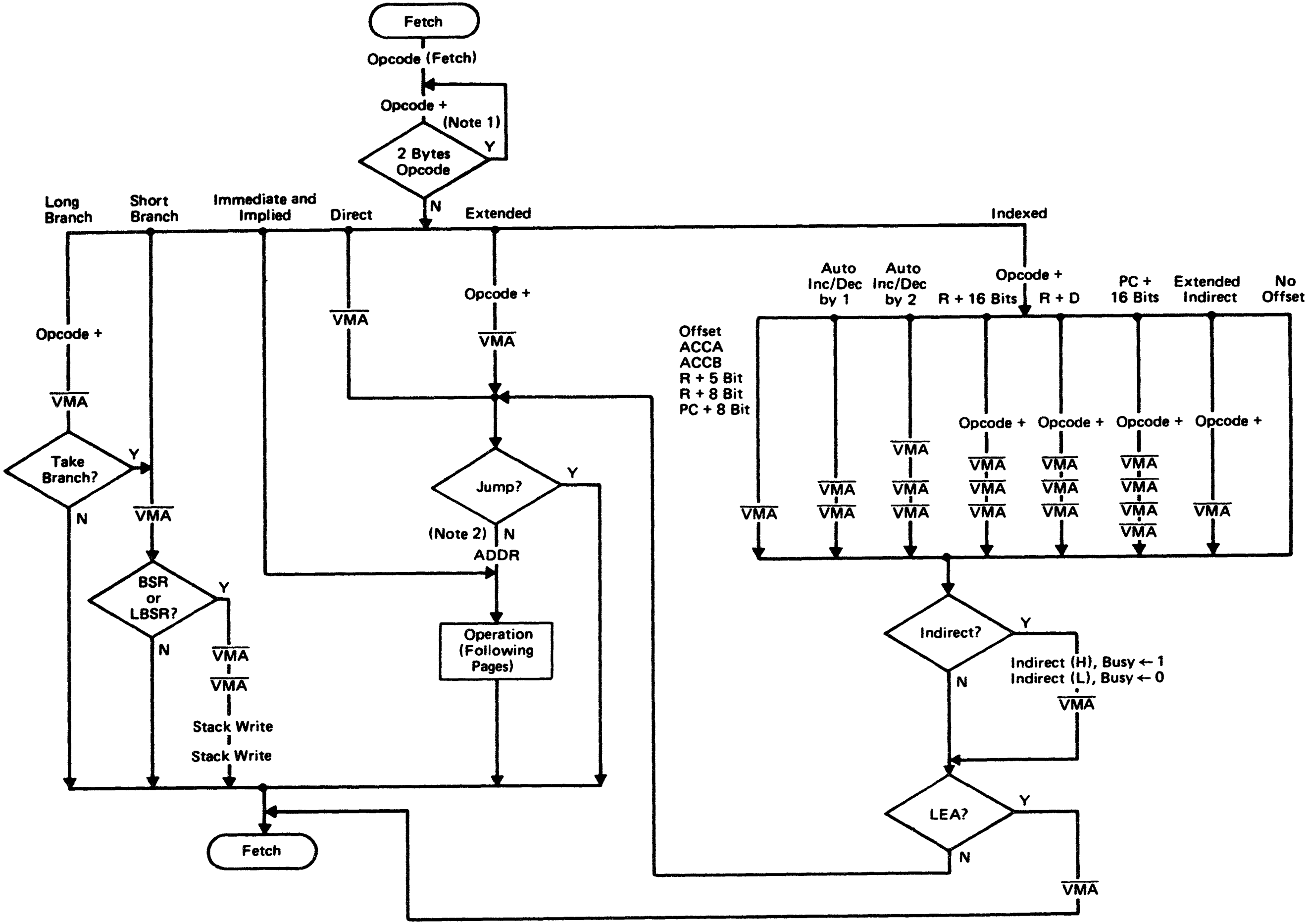


(NOTES) 1. If the associated mask bit is set when the interrupt is requested, LIC will go "Low" and this cycle will be an instruction fetch from address location PC + 1. However, if the interrupt is accepted (NMĪ or an unmasked FIRQ̄ or IRQ̄) LIC will remain "High" and interrupt processing will start with this cycle as (m) on Figure 9 and 10 (Interrupt Timing).

2. If mask bits are clear, IRQ̄ and FIRQ̄ must be held "Low" for three cycles to guarantee that interrupt will be taken, although only one cycle is necessary to bring the processor out of SYNC.

3. Waveform measurements for all inputs and outputs are specified at logic "High" = V_{IHmin} and logic "Low" = V_{ILmax} unless otherwise specified.

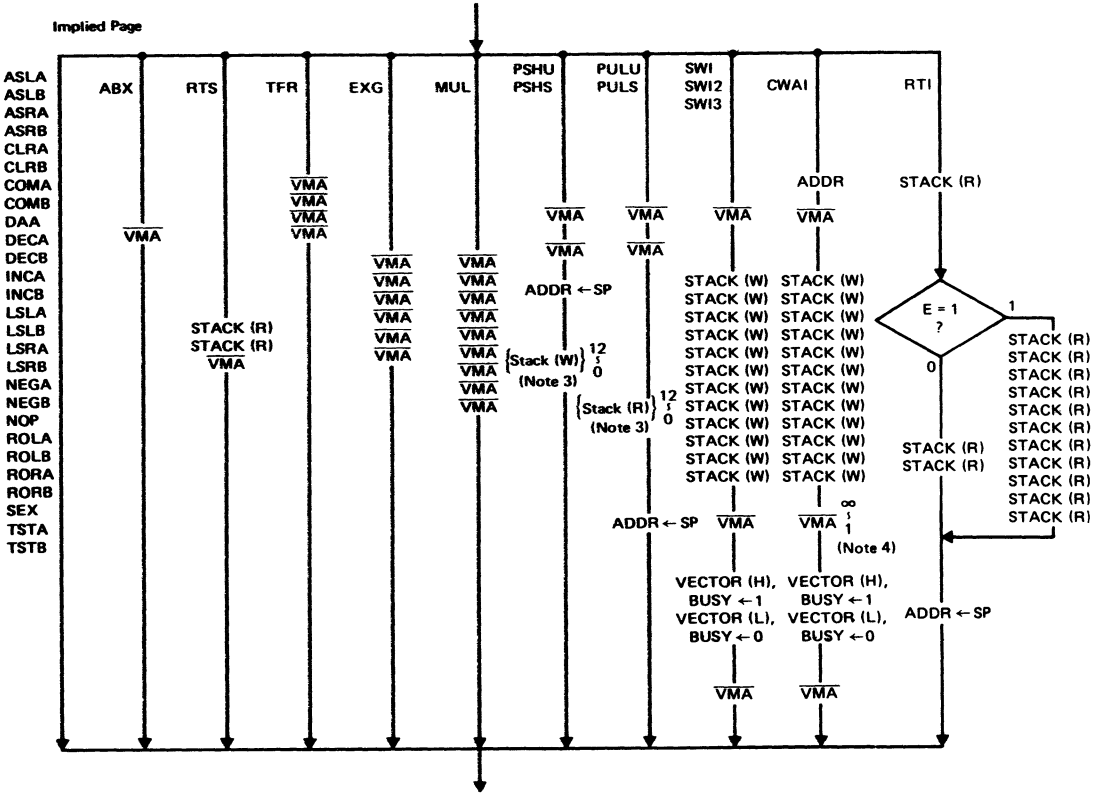
Figure 17 SYNC Timing



(NOTE)

1. Busy = "High" during access of first byte of double byte immediate load.
2. Write operation during store instruction. Busy = "High" during first two cycles of a double-byte access and the first cycle of read-modify-write access.
3. AVMA is asserted on the cycle before a $\overline{\text{VMA}}$ cycle.

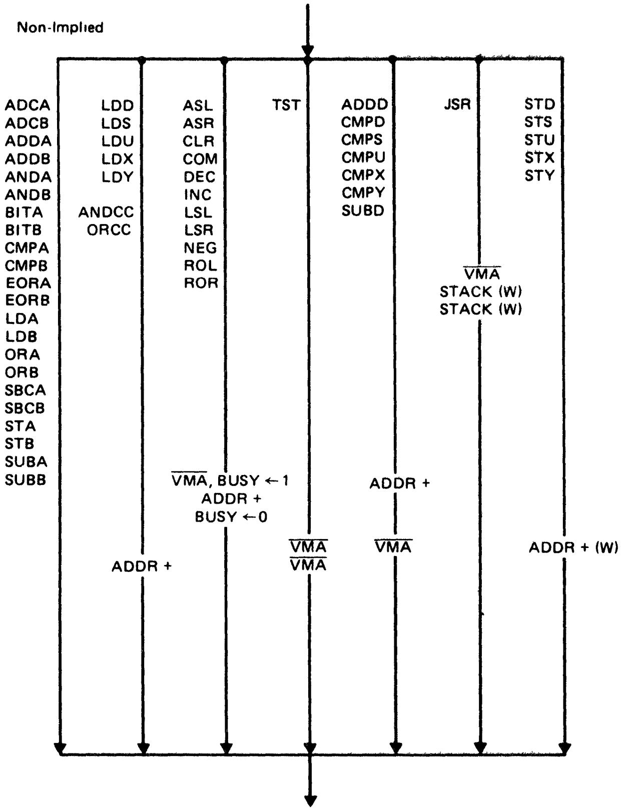
Figure 18 Address Bus Cycle-by-Cycle Performance



(NOTES)

1. Stack (W) refers to the following sequence: SP ← SP − 1, then ADDR ← SP with R/$\overline{\text{W}}$ = "Low"
   Stack (R) refers to the following sequence: ADDR ← SP with R/$\overline{\text{W}}$ = "High", then SP ← SP + 1.
   PSHU, PULU instructions use the user stack pointer (i.e., SP = U) and PSHS, PULS use the hardware stack pointer (i.e., SP = S).
2. Vector refers to the address of an interrupt or reset vector (see Table 1).
3. The number of stack accesses will vary according to the number of bytes saved.
4. $\overline{\text{VMA}}$ cycles will occur until an interrupt occurs.

Figure 18 Address Bus Cycle-by-Cycle Performance (Continued)



(NOTES)
1. Stack (W) refers to the following sequence: SP ← SP — 1, then ADDR ← SP with R/$\overline{W}$ = "Low"
   Stack (R) refers to the following sequence: ADDR ← with R/$\overline{W}$ = "High", then SP ← SP + 1
   PSHU, PULU instructions use the user stack pointer (i.e., SP = U) and PSHS, PULS use the hardware stack pointer (i e , SP = S)
2. Vector refers to the address of an interrupt or reset vector (see Table 1).
3. The number of stack accesses will vary according to the number of bytes saved.
4. $\overline{\text{VMA}}$ cycles will occur until an interrupt occurs

Figure 18 Address Bus Cycle-by-Cycle Performance (Continued)

Table 4 8-Bit Accumulator and Memory Instructions

Mnemonic(s)	Operation
ADCA, ADCB	Add memory to accumulator with carry
ADDA, ADDB	Add memory to accumulator
ANDA, ANDB	And memory with accumulator
ASL, ASLA, ASLB	Arithmetic shift of accumulator or memory left
ASR, ASRA, ASRB	Arithmetic shift of accumulator or memory right
BITA, BITB	Bit test memory with accumulator
CLR, CLRA, CLRB	Clear accumulator or memory location
CMPA, CMPB	Compare memory from accumulator
COM, COMA, COMB	Complement accumultor or memory location
DAA	Decimal adjust A accumulator
DEC, DECA, DECB	Decrement accumulator or memory location
EORA, EORB	Exclusive or memory with accumulator
EXG R1, R2	Exchange R1 with R2 (R1, R2 = A, B, CC, DP)
INC, INCA, INCB	Increment accumulator or memory location
LDA, LDB	Load accumulator from memory
LSL, LSLA, LSLB	Logical shift left accumulator or memory location
LSR, LSRA, LSRB	Logical shift right accumulator or memory location

(Continued)

Table 4 8-Bit Accumulator and Memory Instructions (Continued)

Mnemonic(s)	Operation
MUL	Unsigned multiply (A × B → D)
NEG, NEGA, NEGB	Negate accumulator or memory
ORA, ORB	Or memory with accumulator
ROL, ROLA, ROLB	Rotate accumulator or memory left
ROR, RORA, RORB	Rotate accumulator or memory right
SBCA, SBCB	Subtract memory from accumulator with borrow
STA, STB	Store accumulator to memory
SUBA, SUBB	Subtract memory from accumulator
TST, TSTA, TSTB	Test accumulator or memory location
TFR R1, R2	Transfer R1 to R2 (R1, R2 = A, B, CC, DP)

(NOTE) A, B, CC or DP may be pushed to (pulled from) either stack with PSHS, PSHU (PULS, PULU) instructions.

Table 5 16-Bit Accumulator and Memory Instructions

Mnemonic(s)	Operation
ADDD	Add memory to D accumulator
CMPD	Compare memory from D accumulator
EXG D, R	Exchange D with X, Y, S, U or PC
LDD	Load D accumulator from memory
SEX	Sign Extend B accumulator into A accumulator
STD	Store D accumulator to memory
SUBD	Subtract memory from D accumulator
TFR D, R	Transfer D to X, Y, S, U or PC
TFR R, D	Transfer X, Y, S, U or PC to D

(NOTE) D may be pushed (pulled) to either stack with PSHS, PSHU (PULS, PULU) instructions.

Table 6 Index Register Stack Pointer Instructions

Mnemonic(s)	Operation
CMPS, CMPU	Compare memory from stack pointer
CMPX, CMPY	Compare memory from index register
EXG R1, R2	Exchange D, X, Y, S, U or PC with D, X, Y, S, U or PC
LEAS, LEAU	Load effective address into stack pointer
LEAX, LEAY	Load effective address into index register
LDS, LDU	Load stack pointer from memory
LDX, LDY	Load index register from memory
PSHS	Push A, B, CC, DP, D, X, Y, U, or PC onto hardware stack
PSHU	Push A, B, CC, DP, D, X, Y, S, or PC onto user stack
PULS	Pull A, B, CC, DP, D, X, Y, U or PC from hardware stack
PULU	Pull A, B, CC, DP, D, X, Y, S or PC from user stack
STS, STU	Store stack pointer to memory
STX, STY	Store index register to memory
TFR R1, R2	Transfer D, X, Y, S, U or PC to D, X, Y, S, U or PC
ABX	Add B accumulator to X (unsigned)

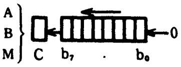
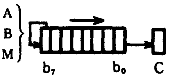
Table 7 Branch Instructions

Mnemonic(s)	Operation
SIMPLE BRANCHES	
BEQ, LBEQ	Branch if equal
BNE, LBNE	Branch if not equal
BMI, LBMI	Branch if minus
BPL, LBPL	Branch if plus
BCS, LBCS	Branch if carry set
BCC, LBCC	Branch if carry clear
BVS, LBVS	Branch if overflow set
BVC, LBVC	Branch if overflow clear
SIGNED BRANCHES	
BGT, LBGT	Branch if greater (signed)
BGE, LBGE	Branch if greater than or equal (signed)
BEQ, LBEQ	Branch if equal
BLE, LBLE	Branch if less than or equal (signed)
BLT, LBLT	Branch if less than (signed)
UNSIGNED BRANCHES	
BHI, LBHI	Branch if higher (unsigned)
BHS, LBHS	Branch if higher or same (unsigned)
BEQ, LBEQ	Branch if equal
BLS, LBLS	Branch if lower or same (unsigned)
BLO, LBLO	Branch if lower (unsigned)
OTHER BRANCHES	
BSR, LBSR	Branch to subroutine
BRA, LBRA	Branch always
BRN, LBRN	Branch never

Table 8 Miscellaneous Instructions

Mnemonic(s)	Operation
ANDCC	AND condition code register
CWAI	AND condition code register, then wait for interrupt
NOP	No operation
ORCC	OR condition code register
JMP	Jump
JSR	Jump to subroutine
RTI	Return from interrupt
RTS	Return from subroutine
SWI, SWI2, SWI3	Software interrupt (absolute indirect)
SYNC	Synchronize with interrupt line

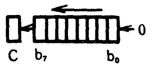
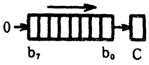
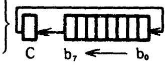
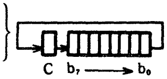
Table 9. HD6309E Instruction Set Table

INSTRUCTIONS/ FORMS		IMP ACCM REG			DIRECT			EXTND			IMMED			INDEX①			RELATIVE			DESCRIPTION	7	6	5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~⑤	#		E	F	H	I	N	Z	V	C
ABX		3A	3	1																B + X→X	●	●	●	●	●	●	●	●
																				(UNSIGNED)								
ADC	ADCA				99	4	2	B9	5	3	89	2	2	A9	4 +	2 +				A + M + C→A	●	●	↕	●	↕	↕	↕	↕
	ADCB				D9	4	2	F9	5	3	C9	2	2	E9	4 +	2 +				B + M + C→B	●	●	↕	●	↕	↕	↕	↕
ADD	ADDA				9B	4	2	BB	5	3	8B	2	2	AB	4 +	2 +				A + M→A	●	●	↕	●	↕	↕	↕	↕
	ADDB				DB	4	2	FB	5	3	CB	2	2	EB	4 +	2 +				B + M→B	●	●	↕	●	↕	↕	↕	↕
	ADDD				D3	6	2	F3	7	3	C3	4	3	E3	6 +	2 +				D + M M + 1→D	●	●	●	●	↕	↕	↕	↕
AND	ANDA				94	4	2	B4	5	3	84	2	2	A4	4 +	2 +				A∧M→A	●	●	●	●	↕	↕	R	●
	ANDB				D4	4	2	F4	5	3	C4	2	2	E4	4 +	2 +				B∧M→B	●	●	●	●	↕	↕	R	●
	ANDCC										1C	3	2							C C∧IMM→C C	(			⑦				)
ASL	ASLA	48	2	1																	●	●	⑧	●	↕	↕	↕	↕
	ASLB	58	2	1																	●	●	⑧	●	↕	↕	↕	↕
	ASL				08	6	2	78	7	3				68	6 +	2 +					●	●	⑧	●	↕	↕	↕	↕
ASR	ASRA	47	2	1																	●	●	⑧	●	↕	↕	●	↕
	ASRB	57	2	1																	●	●	⑧	●	↕	↕	●	↕
	ASR				07	6	2	77	7	3				67	6 +	2 +					●	●	⑧	●	↕	↕	●	↕
BCC	BCC																24	3	2	Branch C = 0	●	●	●	●	●	●	●	●
	LBCC																10	5(6)	4	Long Branch	●	●	●	●	●	●	●	●
																	24			C = 0								
BCS	BCS																25	3	2	Branch C = 1	●	●	●	●	●	●	●	●
	LBCS																10	5(6)	4	Long Branch	●	●	●	●	●	●	●	●
																	25			C = 1								
BEQ	BEQ																27	3	2	Branch Z = 1	●	●	●	●	●	●	●	●
	LBEQ																10	5(6)	4	Long Branch	●	●	●	●	●	●	●	●
																	27			Z = 1								
BGE	BGE																2C	3	2	Branch N⊕V = 0	●	●	●	●	●	●	●	●
	LBGE																10	5(6)	4	Long Branch	●	●	●	●	●	●	●	●
																	2C			N⊕V = 0								
BGT	BGT																2E	3	2	Branch Z∨(N⊕V) = 0	●	●	●	●	●	●	●	●
	LBGT																10	5(6)	4	Long Branch	●	●	●	●	●	●	●	●
																	2E			Z∨(N⊕V) = 0								
BHI	BHI																22	3	2	Branch C∨ Z = 0	●	●	●	●	●	●	●	●
	LBHI																10	5(6)	4	Long Branch	●	●	●	●	●	●	●	●
																	22			C∨ Z = 0								
BHS	BHS																24	3	2	Branch C = 0	●	●	●	●	●	●	●	●
	LBHS																10	5(6)	4	Long Branch	●	●	●	●	●	●	●	●
																	24			C = 0								
BIT	BITA				95	4	2	B5	5	3	85	2	2	A5	4 +	2 +				Bit Test A (M∧A)	●	●	●	●	↕	↕	R	●
	BITB				D5	4	2	F5	5	3	C5	2	2	E5	4 +	2 +				Bit Test B (M∧B)	●	●	●	●	↕	↕	R	●
BLE	BLE																2F	3	2	Branch Z∨(N⊕V) = 1	●	●	●	●	●	●	●	●
	LBLE																10	5(6)	4	Long Branch	●	●	●	●	●	●	●	●
																	2F			Z∨(N⊕V) = 1								
BLO	BLO																25	3	2	Branch C = 1	●	●	●	●	●	●	●	●
	LBLO																10	5(6)	4	Long Branch	●	●	●	●	●	●	●	●
																	25			C = 1								
BLS	BLS																23	3	2	Branch C∨ Z = 1	●	●	●	●	●	●	●	●
	LBLS																10	5(6)	4	Long Branch	●	●	●	●	●	●	●	●
																	23			C∨ Z = 1								
BLT	BLT																2D	3	2	Branch N⊕V = 1	●	●	●	●	●	●	●	●
	LBLT																10	5(6)	4	Long Branch	●	●	●	●	●	●	●	●
																	2D			N⊕V = 1								
BMI	BMI																2B	3	2	Branch N = 1	●	●	●	●	●	●	●	●
	LBMI																10	5(6)	4	Long Branch	●	●	●	●	●	●	●	●
																	2B			N = 1								

(Continued)

INSTRUCTIONS/ FORMS		IMP ACCM REG.			DIRECT			EXTND			IMMED			INDEX①			RELATIVE			DESCRIPTION	7	6	5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~⑤	#		E	F	H	I	N	Z	V	C
BNE	BNE																26	3	2	Branch Z = 0	●	●	●	●	●	●	●	●
	LBNE																10	5(6)	4	Long Branch	●	●	●	●	●	●	●	●
																	26			Z = 0								
BPL	BPL																2A	3	2	Branch N = 0	●	●	●	●	●	●	●	●
	LBPL																10	5(6)	4	Long Branch	●	●	●	●	●	●	●	●
																	2A			N = 0								
BRA	BRA																20	3	2	Branch Always	●	●	●	●	●	●	●	●
	LBRA																16	5	3	Long Branch Always	●	●	●	●	●	●	●	●
BRN	BRN																21	3	2	Branch Never	●	●	●	●	●	●	●	●
	LBRN																10	5	4	Long Branch Never	●	●	●	●	●	●	●	●
																	21											
BSR	BSR																8D	7	2	Branch to	●	●	●	●	●	●	●	●
																				Subroutine								
	LBSR																17	9	3	Long Branch to	●	●	●	●	●	●	●	●
																				Subroutine								
BVC	BVC																28	3	2	Branch V = 0	●	●	●	●	●	●	●	●
	LBVC																10	5(6)	4	Long Branch	●	●	●	●	●	●	●	●
																	28			V = 0								
BVS	BVS																29	3	2	Branch V = 1	●	●	●	●	●	●	●	●
	LBVS																10	5(6)	4	Long Branch	●	●	●	●	●	●	●	●
																	29			V = 1								
CLR	CLRA	4F	2	1																0→A	●	●	●	●	R	S	R	R
	CLRB	5F	2	1																0→B	●	●	●	●	R	S	R	R
	CLR				0F	6	2	7F	7	3				6F	6+	2+				0→M	●	●	●	●	R	S	R	R
CMP	CMPA				91	4	2	B1	5	3	81	2	2	A1	4+	2+				Compare M from A	●	●	⑧	●	↕	↕	↕	↕
	CMPB				D1	4	2	F1	5	3	C1	2	2	E1	4+	2+				Compare M from B	●	●	⑧	●	↕	↕	↕	↕
	CMPD				10	7	3	10	8	4	10	5	4	10	7+	3+				Compare M M+1	●	●	●	●	↕	↕	↕	↕
					93			B3			83			A3						from D								
	CMPS				11	7	3	11	8	4	11	5	4	11	7+	3+				Compare M M+1	●	●	●	●	↕	↕	↕	↕
					9C			BC			8C			AC						from S								
	CMPU				11	7	3	11	8	4	11	5	4	11	7+	3+				Compare M M+1	●	●	●	●	↕	↕	↕	↕
					93			B3			83			A3						from U								
	CMPX				9C	6	2	BC	7	3	8C	4	3	AC	6+	2+				Compare M M+1	●	●	●	●	↕	↕	↕	↕
																				from X								
	CMPY				10	7	3	10	8	4	10	5	4	10	7+	3+				Compare M M+1	●	●	●	●	↕	↕	↕	↕
					9C			BC			8C			AC						from Y								
COM	COMA	43	2	1																Ā→A	●	●	●	●	↕	↕	R	S
	COMB	53	2	1																B̄→B	●	●	●	●	↕	↕	R	S
	COM				03	6	2	73	7	3				63	6+	2+				M̄→M	●	●	●	●	↕	↕	R	S
CWAI											3C	≥20	2							CC ∧ IMM→CC	S	(			⑦			)
																				Wait for Interrupt								
DAA		19	2	1																Decimal Adjust A	●	●	●	●	↕	↕	⑧	↕
DEC	DECA	4A	2	1																A − 1→A	●	●	●	●	↕	↕	↕	●
	DECB	5A	2	1																B − 1→B	●	●	●	●	↕	↕	↕	●
	DEC				0A	6	2	7A	7	3				6A	6+	2+				M − 1→M	●	●	●	●	↕	↕	↕	●
EOR	EORA				98	4	2	B8	5	3	88	2	2	A8	4+	2+				A⊕M→A	●	●	●	●	↕	↕	R	●
	EORB				D8	4	2	F8	5	3	C8	2	2	E8	4+	2+				B⊕M→B	●	●	●	●	↕	↕	R	●
EXG	R1, R2	1E	8	2																R1↔R2②	(			⑩				)
INC	INCA	4C	2	1																A + 1→A	●	●	●	●	↕	↕	↕	●
	INCB	5C	2	1																B + 1→B	●	●	●	●	↕	↕	↕	●
	INC				0C	6	2	7C	7	3				6C	6+	2+				M + 1→M	●	●	●	●	↕	↕	↕	●
JMP					0E	3	2	7E	4	3				6E	3+	2+				EA③→PC	●	●	●	●	●	●	●	●
JSR					9D	7	2	BD	8	3				AD	7+	2+				Jump to Subroutine	●	●	●	●	●	●	●	●

(Continued)

INSTRUCTIONS/ FORMS		IMP ACCM REG			DIRECT			EXTND			IMMED			INDEX①			RELATIVE			DESCRIPTION	7	6	5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~⑤	#		E	F	H	I	N	Z	V	C
LD	LDA				96	4	2	B6	5	3	86	2	2	A6	4+	2+				M→A	●	●	●	●	↕	↕	R	●
	LDB				D6	4	2	F6	5	3	C6	2	2	E6	4+	2+				M→B	●	●	●	●	↕	↕	R	●
	LDD				DC	5	2	FC	6	3	CC	3	3	EC	5+	2+				MM+1→D	●	●	●	●	↕	↕	R	●
	LDS				10 DE	6	3	10 FE	7	4	10 CE	4	4	10 EE	6+	3+				MM+1→S	●	●	●	●	↕	↕	R	●
	LDU				DE	5	2	FE	6	3	CE	3	3	EE	5+	2+				M.M+1→U	●	●	●	●	↕	↕	R	●
	LDX				9E	5	2	BE	6	3	8E	3	3	AE	5+	2+				MM+1→X	●	●	●	●	↕	↕	R	●
	LDY				10 9E	6	3	10 BE	7	4	10 8E	4	4	10 AE	6+	3+				MM+1→Y	●	●	●	●	↕	↕	R	●
LEA	LEAS													32	4+	2+				EA③→S	●	●	●	●	●	●	●	●
	LEAU													33	4+	2+				EA③→U	●	●	●	●	●	●	●	●
	LEAX													30	4+	2+				EA③→X	●	●	●	●	●	↕	●	●
	LEAY													31	4+	2+				EA③→Y	●	●	●	●	●	↕	●	●
LSL	LSLA	48	2	1																A	●	●	●	●	↕	↕	↕	↕
	LSLB	58	2	1																B 	●	●	●	●	↕	↕	↕	↕
	LSL				08	6	2	78	7	3				68	6+	2+				M	●	●	●	●	↕	↕	↕	↕
LSR	LSRA	44	2	1																A	●	●	●	●	R	↕	●	↕
	LSRB	54	2	1																B 	●	●	●	●	R	↕	●	↕
	LSR				04	6	2	74	7	3				64	6+	2+				M	●	●	●	●	R	↕	●	↕
MUL		3D	11	1																A × B→D (Unsigned)	●	●	●	●	●	↕	●	⑨
NEG	NEGA	40	2	1																$\overline{A}$ + 1→A	●	●	⑧	●	↕	↕	↕	↕
	NEGB	50	2	1																$\overline{B}$ + 1→B	●	●	⑧	●	↕	↕	↕	↕
	NEG				00	6	2	70	7	3				60	6+	2+				$\overline{M}$ + 1→M	●	●	⑧	●	↕	↕	↕	↕
NOP		12	2	1																No Operation	●	●	●	●	●	●	●	●
OR	ORA				9A	4	2	BA	5	3	8A	2	2	AA	4+	2+				A∨M→A	●	●	●	●	↕	↕	R	●
	ORB	.			DA	4	2	FA	5	3	CA	2	2	EA	4+	2+				B∨M→B	●	●	●	●	↕	↕	R	●
	ORCC										1A	3	2							CC∨ IMM→CC	(			⑦				)
PSH	PSHS	34	5+④	2																Push Registers on S Stack	●	●	●	●	●	●	●	●
	PSHU	36	5+④	2																Push Registers on U Stack	●	●	●	●	●	●	●	●
PUL	PULS	35	5+④	2																Pull Registers from S Stack	(			⑩				)
	PULU	37	5+④	2																Pull Registers from U Stack	(			⑩				)
ROL	ROLA	49	2	1																A	●	●	●	●	↕	↕	↕	↕
	ROLB	59	2	1																B 	●	●	●	●	↕	↕	↕	↕
	ROL				09	6	2	79	7	3				69	6+	2+				M	●	●	●	●	↕	↕	↕	↕
ROR	RORA	46	2	1																A	●	●	●	●	↕	↕	●	↕
	RORB	56	2	1																B 	●	●	●	●	↕	↕	●	↕
	ROR				06	6	2	76	7	3				66	6+	2+				M	●	●	●	●	↕	↕	●	↕
RTI		3B	6/15	1																Return from Interrupt	(			⑦				)
RTS		39	5	1																Return from Subroutine	●	●	●	●	●	●	●	●
SBC	SBCA				92	4	2	B2	5	3	82	2	2	A2	4+	2+				A−M−C→A	●	●	⑧	●	↕	↕	↕	↕
	SBCB				D2	4	2	F2	5	3	C2	2	2	E2	4+	2+				B−M−C→B	●	●	⑧	●	↕	↕	↕	↕
SEX		1D	2	1																Sign Extend B into A {Bのビット7 = 1 FF→A {Bのビット7 = 0 0→A	●	●	●	●	↕	↕	●	●

(Continued)

INSTRUCTIONS/ FORMS		IMP ACCM REG			DIRECT			EXTND			IMMED			INDEX①			RELATIVE			DESCRIPTION	7	6	5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~⑤	#		E	F	H	I	N	Z	V	C
ST	STA				97	4	2	B7	5	3				A7	4+	2+				A→M	●	●	●	●	↕	↕	R	●
	STB				D7	4	2	F7	5	3				E7	4+	2+				B→M	●	●	●	●	↕	↕	R	●
	STD				DD	5	2	FD	6	3				ED	5+	2+				D→M M+1	●	●	●	●	↕	↕	R	●
	STS				10 DF	6	3	10 FF	7	4				10 EF	6+	3+				S→M M+1	●	●	●	●	↕	↕	R	●
	STU				DF	5	2	FF	6	3				EF	5+	2+				U→M M+1	●	●	●	●	↕	↕	R	●
	STX				9F	5	2	BF	6	3				AF	5+	2+				X→M M+1	●	●	●	●	↕	↕	R	●
	STY				10 9F	6	3	10 BF	7	4				10 AF	6+	3+				Y→M M+1	●	●	●	●	↕	↕	R	●
SUB	SUBA				90	4	2	B0	5	3	80	2	2	A0	4+	2+				A−M→A	●	●	⑧	●	↕	↕	↕	↕
	SUBB				D0	4	2	F0	5	3	C0	2	2	E0	4+	2+				B−M→B	●	●	⑧	●	↕	↕	↕	↕
	SUBD				93	6	2	B3	7	3	83	4	3	A3	6+	2+				D−M M+1→D	●	●	●	●	↕	↕	↕	↕
SWI	SWI⑥	3F	19	1																Software interrupt 1	S	S	●	S	●	●	●	●
	SWI2⑥	10 3F	20	2																Software interrupt 2	S	●	●	●	●	●	●	●
	SWI3⑥	11 3F	20	2																Software interrupt 3	S	●	●	●	●	●	●	●
SYNC		13	≧4	1																Synchronize to interrupt	●	●	●	●	●	●	●	●
TFR	R1, R2	1F	6	2																R1 → R2②	(			⑩				)
TST	TSTA	4D	2	1																Test A	●	●	●	●	↕	↕	R	●
	TSTB	5D	2	1																Test B	●	●	●	●	↕	↕	R	●
	TST				0D	6	2	7D	7	3				6D	6+	2+				Test M	●	●	●	●	↕	↕	R	●

(NOTES)

(1) This column gives a base cycle and byte count. To obtain total count, and the values obtained from the INDEXED ADDRESSING MODES table.
(2) R1 and R2 may be any pair of 8 bit or any pair of 16 bit registers.
The 8 bit registers are: A, B, CC, DP
The 16 bit registers are: X, Y, U, S, D, PC
③ EA is the effective address.
④ The PSH and PUL instructions require 5 cycle plus 1 cycle for each byte pushed or pulled.
⑤ 5(6) means: 5 cycles if branch not taken, 6 cycles if taken.
⑥ SWI sets 1 and F bits. SWI2 and SWI3 do not affect I and F.
⑦ Conditions Codes set as a direct result of the instruction.
⑧ Value of half-carry flag is undefined.
⑨ Special Case — Carry set if b7 is SET.
⑩ Condition Codes set as a direct result of the instruction if CC is specified, and not affected otherwise.

LEGEND:

OP	Operation Code (Hexadecimal)	Z	Zero (byte)
~	Number of MPU Cycles	V	Overflow, 2's complement
#	Number of Program Bytes	C	Carry from bit 7
+	Arithmetic Plus	↕	Test and set if true, cleared otherwise
−	Arithmetic Minus	●	Not Affected
×	Multiply	CC	Condition Code Register
$\overline{M}$	Complement of M	:	Concatenation
→	Transfer Into	∨	Logical or
H	Half-carry (from bit 3)	∧	Logical and
N	Negative (sign bit)	⊕	Logical Exclusive or

Table 10 Hexadecimal Values of Machine Codes

OP	Mnem	Mode	~	#	OP	Mnem	Mode	~	#	OP	Mnem	Mode	~	#
00	NEG	Direct	6	2	30	LEAX	Indexed	4+	2+	60	NEG	Indexed	6+	2+
01	*	↑			31	LEAY	↑	4+	2+	61	*	↑		
02	*				32	LEAS	↓	4+	2+	62	*			
03	COM		6	2	33	LEAU	Indexed	4+	2+	63	COM		6+	2+
04	LSR		6	2	34	PSHS	Implied	5+	2	64	LSR		6+	2+
05	*				35	PULS	↑	5+	2	65	*			
06	ROR		6	2	36	PSHU		5+	2	66	ROR		6+	2+
07	ASR		6	2	37	PULU		5+	2	67	ASR		6+	2+
08	ASL, LSL		6	2	38	*				68	ASL, LSL		6+	2+
09	ROL		6	2	39	RTS		5	1	69	ROL		6+	2+
0A	DEC		6	2	3A	ABX	↓	3	1	6A	DEC		6+	2+
0B	*				3B	RTI	Implied	6, 15	1	6B	*			
0C	INC		6	2	3C	CWAI	Immed	≥20	2	6C	INC		6+	2+
0D	TST		6	2	3D	MUL	Implied	11	1	6D	TST		6+	2+
0E	JMP	↓	3	2	3E	*				6E	JMP	↓	3+	2+
0F	CLR	Direct	6	2	3F	SWI	Implied	19	1	6F	CLR	Indexed	6+	2+
10	} See	–	–	–	40	NEGA	Implied	2	1	70	NEG	Extended	7	3
11	} Next Page	–	–	–	41	*	↑			71	*	↑		
12	NOP	Implied	2	1	42	*				72	*			
13	SYNC	Implied	≥4	1	43	COMA		2	1	73	COM		7	3
14	*				44	LSRA		2	1	74	LSR		7	3
15	*				45	*				75	*			
16	LBRA	Relative	5	3	46	RORA		2	1	76	ROR		7	3
17	LBSR	Relative	9	3	47	ASRA		2	1	77	ASR		7	3
18	*				48	ASLA, LSLA		2	1	78	ASL, LSL		7	3
19	DAA	Implied	2	1	49	ROLA		2	1	79	ROL		7	3
1A	ORCC	Immed	3	2	4A	DECA		2	1	7A	DEC		7	3
1B	*	–			4B	*				7B	*			
1C	ANDCC	Immed	3	2	4C	INCA		2	1	7C	INC		7	3
1D	SEX	Implied	2	1	4D	TSTA		2	1	7D	TST		7	3
1E	EXG	↕	8	2	4E	*	↓			7E	JMP	↓	4	3
1F	TFR	Implied	6	2	4F	CLRA	Implied	2	1	7F	CLR	Extended	7	3
20	BRA	Relative	3	2	50	NEGB	Implied	2	1	80	SUBA	Immed	2	2
21	BRN	↑	3	2	51	*	↑			81	CMPA	↑	2	2
22	BHI		3	2	52	*				82	SBCA		2	2
23	BLS		3	2	53	COMB		2	1	83	SUBD		4	3
24	BHS, BCC		3	2	54	LSRB		2	1	84	ANDA		2	2
25	BLO, BCS		3	2	55	*				85	BITA		2	2
26	BNE		3	2	56	RORB		2	1	86	LDA		2	2
27	BEQ		3	2	57	ASRB		2	1	87	*			
28	BVC		3	2	58	ASLB, LSLB		2	1	88	EORA		2	2
29	BVS		3	2	59	ROLB		2	1	89	ADCA		2	2
2A	BPL		3	2	5A	DECB		2	1	8A	ORA		2	2
2B	BMI		3	2	5B	*				8B	ADDA	↓	2	2
2C	BGE		3	2	5C	INCB		2	1	8C	CMPX	Immed	4	3
2D	BLT		3	2	5D	TSTB		2	1	8D	BSR	Relative	7	2
2E	BGT	↓	3	2	5E	*	↓			8E	LDX	Immed	3	3
2F	BLE	Relative	3	2	5F	CLRB	Implied	2	1	8F	*			

LEGEND:
~ Number of MPU cycles (less possible push pull or indexed-mode cycles)
# Number of program bytes
* Denotes unused opcode

(to be continued)

OP	Mnem	Mode	~	#	OP	Mnem	Mode	~	#	OP	Mnem	Mode	~	#
90	SUBA	Direct	4	2	C6	LDB	Immed	2	2	FC	LDD	Extended	6	3
91	CMPA	↑	4	2	C7	*	↑			FD	STD	↑	6	3
92	SBCA		4	2	C8	EORB		2	2	FE	LDU	↓	6	3
93	SUBD		6	2	C9	ADCB		2	2	FF	STU	Extended	6	3
94	ANDA		4	2	CA	ORB		2	2					
95	BITA		4	2	CB	ADDB		2	2					
96	LDA		4	2	CC	LDD		3	3		2 Bytes Opcode			
97	STA		4	2	CD	*	↓							
98	EORA		4	2	CE	LDU	Immed	3	3	1021	LBRN	Relative	5	4
99	ADCA		4	2	CF	*				1022	LBHI	↑	5(6)	4
9A	ORA		4	2						1023	LBLS		5(6)	4
9B	ADDA		4	2	D0	SUBB	Direct	4	2	1024	LBHS, LBCC		5(6)	4
9C	CMPX		6	2	D1	CMPB	↑	4	2	1025	LBCS, LBLO		5(6)	4
9D	JSR		7	2	D2	SBCB		4	2	1026	LBNE		5(6)	4
9E	LDX	↓	5	2	D3	ADDD		6	2	1027	LBEQ		5(6)	4
9F	STX	Direct	5	2	D4	ANDB		4	2	1028	LBVC		5(6)	4
					D5	BITB		4	2	1029	LBVS		5(6)	4
A0	SUBA	Indexed	4+	2+	D6	LDB		4	2	102A	LBPL		5(6)	4
A1	CMPA	↑	4+	2+	D7	STB		4	2	102B	LBMI		5(6)	4
A2	SBCA		4+	2+	D8	EORB		4	2	102C	LBGE		5(6)	4
A3	SUBD		6+	2+	D9	ADCB		4	2	102D	LBLT		5(6)	4
A4	ANDA		4+	2+	DA	ORB		4	2	102E	LBGT	↓	5(6)	4
A5	BITA		4+	2+	DB	ADDB		4	2	102F	LBLE	Relative	5(6)	4
A6	LDA		4+	2+	DC	LDD		5	2	103F	SWI2	Implied	20	2
A7	STA		4+	2+	DD	STD		5	2	1083	CMPD	Immed	5	4
A8	EORA		4+	2+	DE	LDU	↓	5	2	108C	CMPY	↕	5	4
A9	ADCA		4+	2+	DF	STU	Direct	5	2	108E	LDY	Immed	4	4
AA	ORA		4+	2+						1093	CMPD	Direct	7	3
AB	ADDA		4+	2+	E0	SUBB	Indexed	4+	2+	109C	CMPY	↑	7	3
AC	CMPX		6+	2+	E1	CMPB	↑	4+	2+	109E	LDY	↓	6	3
AD	JSR		7+	2+	E2	SBCB		4+	2+	109F	STY	Direct	6	3
AE	LDX	↓	5+	2+	E3	ADDD		6+	2+	10A3	CMPD	Indexed	7+	3+
AF	STX	Indexed	5+	2+	E4	ANDB		4+	2+	10AC	CMPY	↑	7+	3+
					E5	BITB		4+	2+	10AE	LDY	↓	6+	3+
B0	SUBA	Extended	5	3	E6	LDB		4+	2+	10AF	STY	Indexed	6+	3+
B1	CMPA	↑	5	3	E7	STB		4+	2+	10B3	CMPD	Extended	8	4
B2	SBCA		5	3	E8	EORB		4+	2+	10BC	CMPY	↑	8	4
B3	SUBD		7	3	E9	ADCB		4+	2+	10BE	LDY	↓	7	4
B4	ANDA		5	3	EA	ORB		4+	2+	10BF	STY	Extended	7	4
B5	BITA		5	3	EB	ADDB		4+	2+	10CE	LDS	Immed	4	4
B6	LDA		5	3	EC	LDD		5+	2+	10DE	LDS	Direct	6	3
B7	STA		5	3	ED	STD		5+	2+	10DF	STS	Direct	6	3
B8	EORA		5	3	EE	LDU	↓	5+	2+	10EE	LDS	Indexed	6+	3+
B9	ADCA		5	3	EF	STU	Indexed	5+	2+	10EF	STS	Indexed	6+	3+
BA	ORA		5	3						10FE	LDS	Extended	7	4
BB	ADDA		5	3	F0	SUBB	Extended	5	3	10FF	STS	Extended	7	4
BC	CMPX		7	3	F1	CMPB	↑	5	3	113F	SWI3	Implied	20	2
BD	JSR		8	3	F2	SBCB		5	3	1183	CMPU	Immed	5	4
BE	LDX	↓	6	3	F3	ADDD		7	3	118C	CMPS	Immed	5	4
BF	STX	Extended	6	3	F4	ANDB		5	3	1193	CMPU	Direct	7	3
					F5	BITB		5	3	119C	CMPS	Direct	7	3
C0	SUBB	Immed	2	2	F6	LDB		5	3	11A3	CMPU	Indexed	7+	3+
C1	CMPB	↑	2	2	F7	STB		5	3	11AC	CMPS	Indexed	7+	3+
C2	SBCB		2	2	F8	EORB		5	3	11B3	CMPU	Extended	8	4
C3	ADDD		4	3	F9	ADCB		5	3	11BC	CMPS	Extended	8	4
C4	ANDB	↓	2	2	FA	ORB	↓	5	3					
C5	BITB	Immed	2	2	FB	ADDB	Extended	5	3					

(NOTE): All unused opcodes are both undefined and illegal

■ NOTE FOR USE

● Execution Sequence of CLR Instruction

Example: CLR (Extended)

$8000	CLR	$A000
$A000	FCB	$80

Cycle #	Address	Data	R/$\overline{W}$	Description
1	8000	7F	1	Opcode Fetch
2	8001	A0	1	Operand Address, High Byte
3	8002	00	1	Operand Address, Low Byte
4	FFFF	*	1	$\overline{\text{VMA}}$ Cycle
5	A000	80	1	Read the Data
6	FFFF	*	1	$\overline{\text{VMA}}$ Cycle
7	A000	00	0	Store Fixed "00" into Specified Location

* The data bus has the data at that particular address.

Cycle-by-cycle flow of CLR instruction (Direct, Extended, Indexed Addressing Mode) is shown below. In this sequence the content of the memory location specified by the operand is read before writing "00" into it. Note that status Flags, such as IRQ Flag, will be cleared by this extra data read operation when accessing the control/status register (sharing the same address between read and write) of peripheral devices.

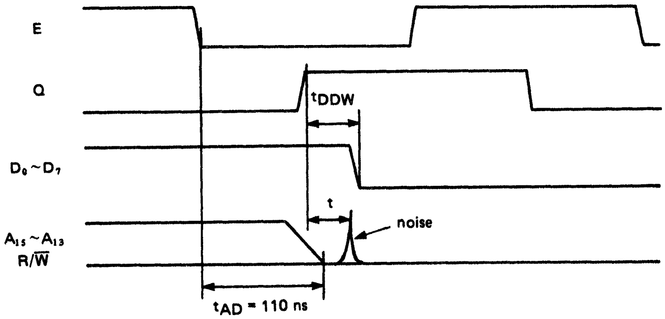
● The Noise of HD6309E at Bus Outputs Changing

We shall notify you of the noise of the HD6309E.

The noise over 0.8V may appear on the output signals when data bus or address bus outputs change from "High" to "Low". Problems and countermeasure are shown as follows.

(1) The Noise at Data Bus Outputs Changing ("High→"Low")

Problem: The noise over 0.8V may appear on A_{15}~A_{13}, R/W outputs change (worst case; $FF→$00) as shown in Figure 19.



Noise peak (worst case); about 1.5V
Test condition
Ta = −20°C
V_{CC} = 5.5V
Number of data bus lines switching from "High" to "Low" = 8
($FF→$00) data bus load capacitance = 130pF

Period of the noise occurrence (reference data)

t = 6~34ns (Ta = −20°C)
t = 8~43ns (Ta = 25°C)
t = 12~54ns (Ta = 75°C)

Figure 19 Noise at data bus output changing

Countermeasure: If the noise level can not be reduced by controlling data bus load capacitance or reducing V_{CC} in your application system, connect damping resistors (about 100~150Ω) to data bus to reduce the noise level as shown in Figure 20. Table 11 shows the relationship between damping resistors and electrical characteristics. Connecting damping resistors to data bus is effective to reduce the noise level as shown in Figure 21.

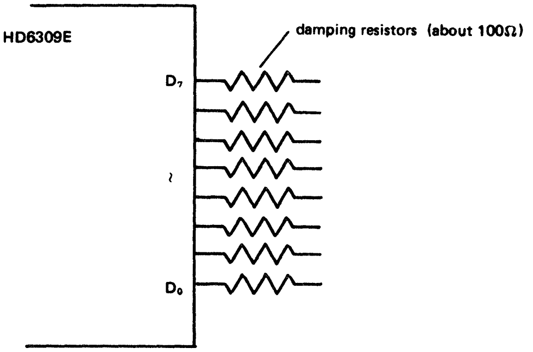
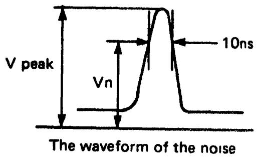


Figure 20 Connecting damping resistors to data bus

Table 11 The relationship between damping resistors and electrical characteristics

			R = 0Ω	R = 100 ~ 150Ω
HD63B09E (2MHz)	t_{DHW}	Ta = −20~0°C	20 ns	10 ns
		Ta = 0~75°C	30 ns	15 ns
HD63C09E (3MHz)	t_{DDW}		70 ns	80 ns
	t_{DHW}	Ta = −20~0°C	20 ns	10 ns
		Ta = 0~75°C	30 ns	15 ns



The waveform of the noise

Test condition
V_{CC} = 5.5V
Ta = −20°C
data bus load capacitance
= 130pF

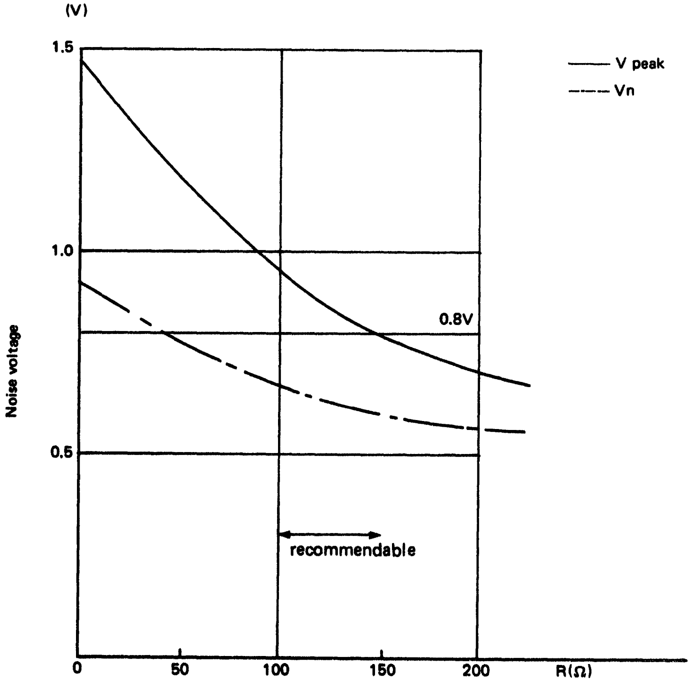
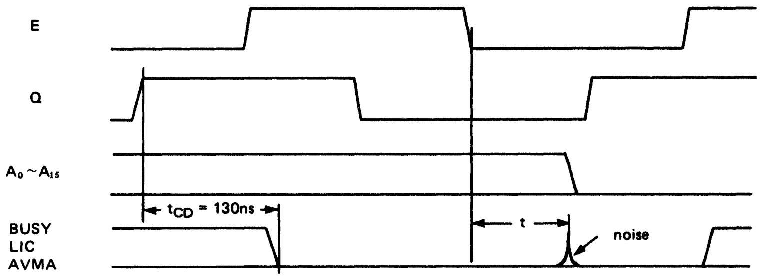


Figure 21 An example of the dependency of the noise voltage on damping resistors

2. The Noise at Address Bus Outputs Changing ("High" → "Low")

Problem: The noise over 0.8V may appear on BUSY, LIC, AVMA outputs when address bus outputs change (worst case; $FFFF→$0000) as shown in Figure 22.



Noise peak (worst case); about 1.5V

Test condition

Ta = −20°C

V_{CC} = 5.5V

Number of address bus lines switching from "High" to "Low" = 16 ($FFFF→$0000) address bus load capacitance = 90pF

Period of the noise occurrence (reference data)

t = 25~65ns (Ta = −20°C)
t = 30~74ns (Ta = 25°C)
t = 34~83ns (Ta = 75°C)

Figure 22 Noise at address bus output changing

Countermeasure: To prevent the noise on BUSY, LIC, AVMA outputs from appearing, this signals must be latched at the negative edge of E or Q clock as shown in Figure 23. An example of countermeasure circuit is shown in Figure 24.

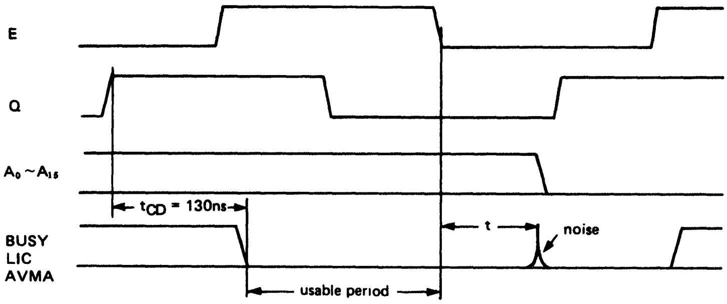


Figure 23 An example of countermeasure of the noise

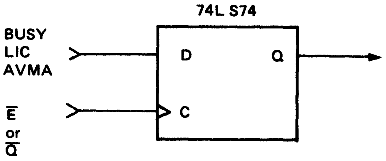


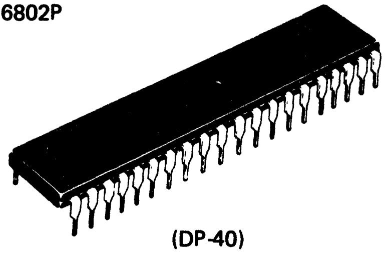
Figure 24 An example of countermeasure circuit

HD6802
MPU (Microprocessor with Clock and RAM)

The HD6802 is a monolithic 8-bit microprocessor that contains all the registers and accumulators of the present HD6800 plus an internal clock oscillator and driver on the same chip. In addition, the HD6802 has 128 bytes of RAM on the chip located at hex addresses 0000 to 007F. The first 32 bytes of RAM, at hex addresses 0000 to 001F, may be retained in a low power mode by utilizing V_{CC} standby, thus facilitating memory retention during a power-down situation.

The HD6802 is completely software compatible with the HD6800 as well as the entire HMCS6800 family of parts. Hence, the HD6802 is expandable to 65k words.

HD6802P

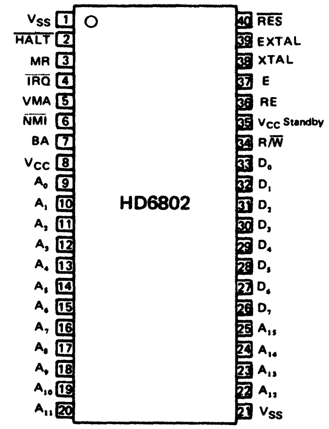


(DP-40)

■ FEATURES

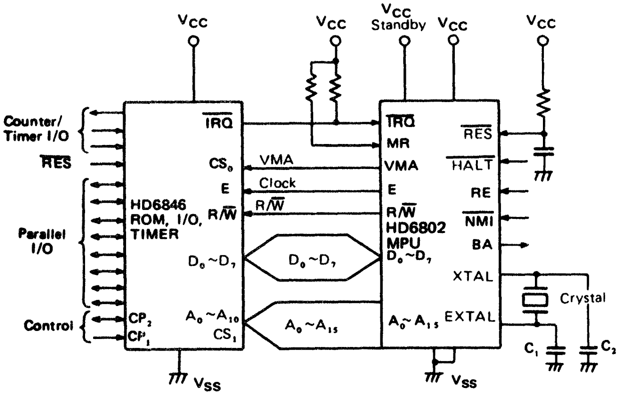
- On-Chip Clock Circuit
- 128 × 8 Bit On-Chip RAM
- 32 Bytes of RAM are Retainable
- Software-Compatible with the HD6800
- Expandable to 65k words
- Standard TTL-Compatible Inputs and Outputs
- 8 Bit Word Size
- 16 Bit Memory Addressing
- Interrupt Capability
- Compatible with MC6802

■ PIN ARRANGEMENT



(Top View)

■ BLOCK DIAGRAM



■ ABSOLUTE MAXIMUM RATINGS

Item	Symbol	Value	Unit
Supply Voltage	V_{CC} * V_{CC} Standby*	-0.3 ~ +7.0	V
Input Voltage	V_{in} *	-0.3 ~ +7.0	V
Operating Temperature	T_{opr}	-20 ~ +75	°C
Storage Temperature	T_{stg}	-55 ~ +150	°C

* With respect to V_{SS} (SYSTEM GND)

(NOTE) Permanent LSI damage may occur if maximum ratings are exceeded. Normal operation should be under recommended operating conditions. If these conditions are exceeded, it could affect reliability of LSI.

■ RECOMMENDED OPERATING CONDITIONS

Item	Symbol		min	typ	max	Unit
Supply Voltage	V_{CC} * V_{CC} Standby*		4.75	5.0	5.25	V
Input Voltage	V_{IL} *		-0.3	–	0.8	V
	V_{IH} *	Except $\overline{RES}$	2.0	–	V_{CC}	V
		$\overline{RES}$	4.25	–	V_{CC}	V
Operation Temperature	T_{opr}		-20	25	75	°C

* With respect to V_{SS} (SYSTEM GND)

■ ELECTRICAL CHARACTERISTICS

● DC CHARACTERISTICS (V_{CC}=5.0V±5%, V_{CC} Standby=5.0V±5%, V_{SS}=0V, Ta=-20~+75°C, unless otherwise noted.)

Item		Symbol	Test Condition	min	typ**	max	Unit
Input "High" Voltage	Except $\overline{RES}$	V_{IH}		2.0	–	V_{CC}	V
	$\overline{RES}$			4.25	–	V_{CC}	
Input "Low" Voltage	Except $\overline{RES}$	*** V_{IL}		-0.3	–	0.8	V
	$\overline{RES}$			-0.3	–	0.8	
Output "High" Voltage	D_0~D_7, E	V_{OH}	I_{OH} = -205μA	2.4	–	–	V
	A_0~A_{15}, R/$\overline{W}$, VMA		I_{OH} = -145μA	2.4	–	–	
	BA		I_{OH} = -100μA	2.4	–	–	
Output "Low" Voltage		V_{OL}	I_{OL} = 1.6mA	–	–	0.4	V
Three State (Off State) Input Current	D_0~D_7	I_{TSI}	V_{in} = 0.4~2.4V	-10	–	10	μA
Input Leakage Current	Except D_0~D_7 ****	I_{in}	V_{in} = 0~5.25V	-2.5	–	2.5	μA
Power Dissipation		P_D *		–	0.6	1.2	W
Input Capacitance	D_0~D_7	C_{in}	V_{in}=0V, T_a=25°C, f=1.0MHz	–	10	12.5	pF
	Except D_0~D_7			–	6.5	10	
Output Capacitance	A_0~A_{15}, R/$\overline{W}$, BA, VMA, E	C_{out}	V_{in}=0V, T_a=25°C, f=1.0MHz	–	–	12	pF

* In power-down mode, maximum power dissipation is less than 42mW.

** T_a=25°C, V_{CC} =5V

*** As $\overline{RES}$ input has histeresis character, applied voltage up to 2.4V is regarded as "Low" level when it goes up from 0V.

**** Does not include EXTAL and XTAL, which are crystal inputs.

- **AC CHARACTERISTICS (V_{CC}=5.0V±5%, V_{CC} Standby=5.0V±5%, V_{SS}=0V, Ta=-20~+75°C, unless otherwise noted.)**

1. CLOCK TIMING CHARACTERISTICS

Item		Symbol	Test Condition	min	typ	max	Unit
Frequency of Operation	Input Clock ÷ 4	f		0.1	–	1.0	MHz
	Crystal Frequency	f_{XTAL}		1.0	–	4.0	
Cycle Time		t_{cyc}	Fig. 2, Fig. 3	1.0	–	10	μs
Clock Pulse Width	"High" Level	$PW_{\phi H}$	at 2.4V (Fig. 2, Fig. 3)	450	–	4500	ns
	"Low" Level	$PW_{\phi L}$	at 0.4V (Fig. 2, Fig. 3)				
Clock Fall Time		t_{ϕ}	0.4V ~ 2.4V(Fig.2,Fig.3)	–	–	25	ns

2. READ/WRITE TIMING

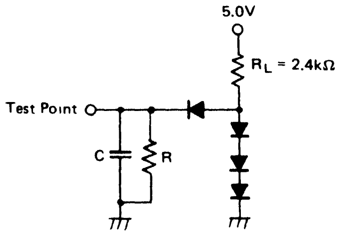
Item	Symbol	Test Condition	min	typ*	max	Unit
Address Delay	t_{AD}	Fig. 2, Fig. 3, Fig. 6	–	–	270	ns
Peripheral Read Access Time	t_{acc}	Fig. 2	530	–	–	ns
Data Setup Time (Read)	t_{DSR}	Fig. 2	100	–	–	ns
Input Data Hold Time	t_H	Fig. 2	10	–	–	ns
Output Data Hold Time	t_H	Fig. 3	20	–	–	ns
Address Hold Time (Address, R/$\overline{W}$, VMA)	t_{AH}	Fig. 2, Fig. 3	10	–	–	ns
Data Delay Time (Write)	t_{DDW}	Fig. 3	–	–	225	ns
Bus Available Delay	t_{BA}	Fig. 4, Fig. 5, Fig. 7, Fig. 8	–	–	250	ns
Processor Controls Processor Control Setup Time	t_{PCS}	Fig. 4~Fig. 7, Fig. 12	200	–	–	ns
Processor Control Rise and Fall Time (Measured at 0.8V and 2.0V)	t_{PCr} t_{PCf}	Fig. 4~Fig. 7, Fig. 12, Fig. 13, Fig. 16	–	–	100	ns

*Ta = 25°C, V_{CC} = 5V

3. POWER DOWN SEQUENCE TIMING, POWER UP RESET TIMING AND MEMORY READY TIMING

Item	Symbol	Test Condition	min	typ	max	Unit
RAM Enable Reset Time (1)	t_{RE1}	Fig. 13	150	–	–	ns
RAM Enable Reset Time (2)	t_{RE2}	Fig. 13	E-3 cycles	–	–	
Reset Release Time	t_{LRES}	Fig. 12	20*	–	–	ms
RAM Enable Reset Time (3)	t_{RE3}	Fig. 12	0	–	–	ns
Memory Ready Setup Time	t_{SMR}	Fig. 16	300	–	–	ns
Memory Ready Hold Time	t_{HMR}	Fig. 16	0	–	200	ns

*t_{RES} = 20 msec min. for S type, 50 msec min. for R type.



C = 130pF for D_0~D_7, E
= 90pF for A_0~A_{15}, R/$\overline{W}$, and VMA
= 30pF for BA
R = 11kΩ for D_0~D_7, E
= 16kΩ for A_0~A_{15}, R/$\overline{W}$, and VMA
= 24kΩ for BA
C includes stray Capacitance.
All diodes are 1S2074 Ⓗ or equivalent

Figure 1 Bus Timing Test Load

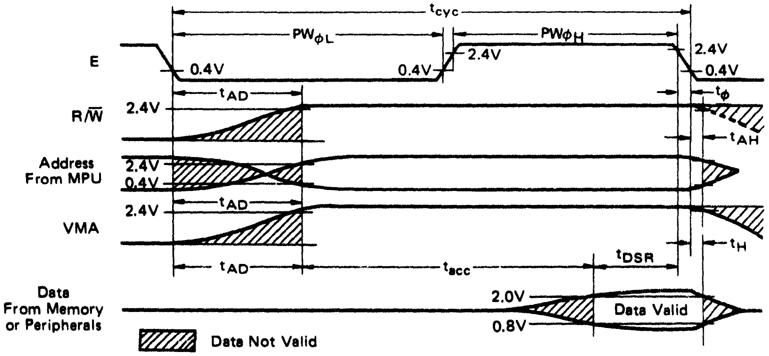


Figure 2 Read Data from Memory or Peripherals

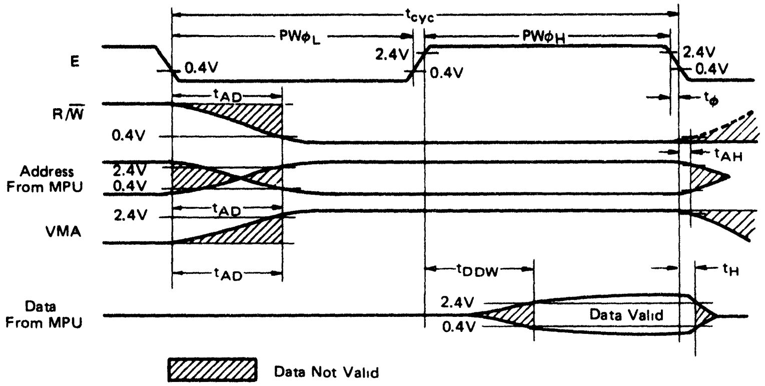


Figure 3 Write Data in Memory or Peripherals

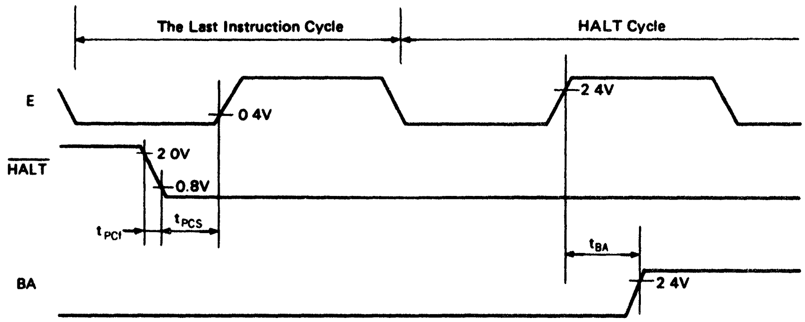


Figure 4 Timing of $\overline{\text{HALT}}$ and BA

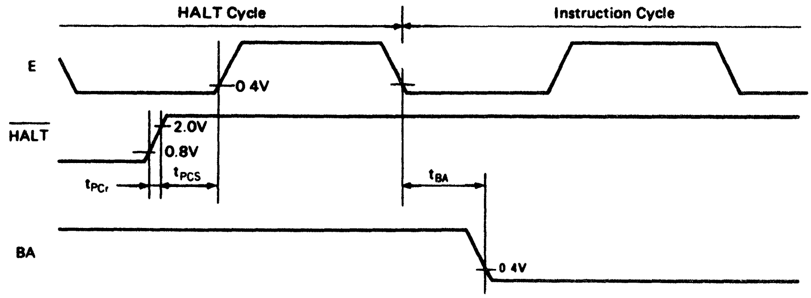


Figure 5 Timing of $\overline{\text{HALT}}$ and BA

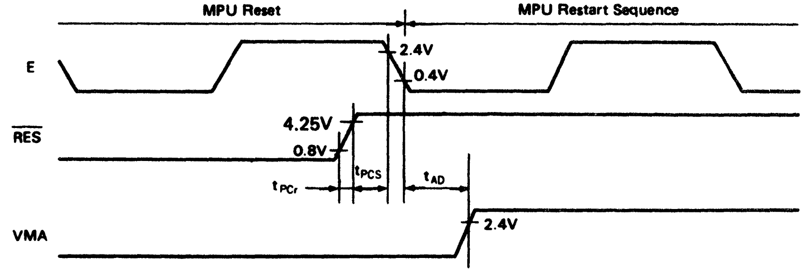


Figure 6 $\overline{\text{RES}}$ and MPU Restart Sequence

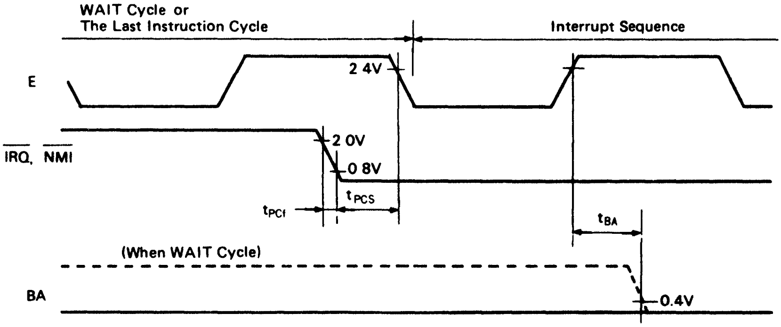


Figure 7 $\overline{\text{IRQ}}$ and $\overline{\text{NMI}}$ Interrupt Timing

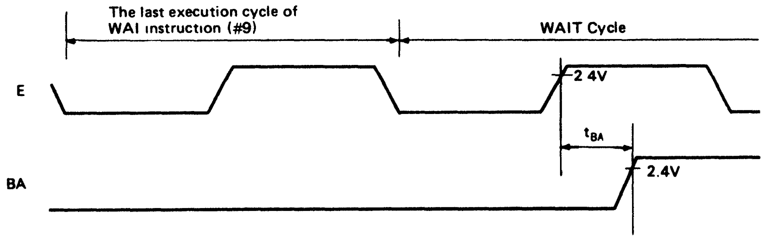


Figure 8 WAI Instruction and BA Timing

■ MPU REGISTERS

A general block diagram of the HD6802 is shown in Fig. 9. As shown, the number and configuration of the registers are the same as for the HD6800. The 128 x 8 bit RAM has been added to the basic MPU. The first 32 bytes may be operated in a low power mode via a V_{CC} standby. These 32 bytes can be retained during power-up and power-down conditions via the RE signal.

The MPU has three 16-bit registers and three 8-bit registers available for use by the programmer (Fig. 10).

- **Program Counter (PC)**

The program counter is a two byte (16-bit) register that points to the current program address.

- **Stack Pointer (SP)**

The stack pointer is a two byte (16-bit) register that contains the address of the next available location in an external push-down/pop-up stack. This stack is normally a random access Read/Write memory that may have any location (address) that is convenient. In those applications that require storage of information in the stack when power is lost, the stack must be non-volatile.

- **Index Register (IX)**

The index register is a two byte register that is used to store data or a sixteen bit memory address for the Indexed mode of memory addressing.

- **Accumulators (ACCA, ACCB)**

The MPU contains two 8-bit accumulators that are used to hold operands and results from an arithmetic logic unit(ALU).

- **Condition Code Register (CCR)**

The condition code register indicates the results of an Arithmetic Logic Unit operation: Negative(N), Zero(Z), Overflow(V), Carry from bit7(C), and half carry from bit3(H). These bits of the Condition Code Register are used as testable conditions for the conditional branch instructions. Bit 4 is the interrupt mask bit(I). The used bits of the Condition Code Register (B6 and B7) are ones.

Fig. 11 shows the order of saving the microprocessor status within the stack.

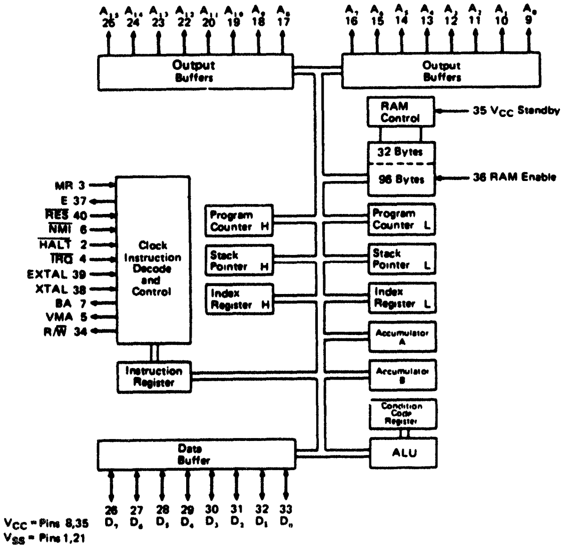


Figure 9 Expanded Block Diagram

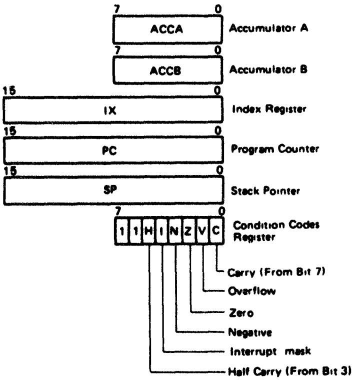


Figure 10 Programming Model of The Microprocessing Unit

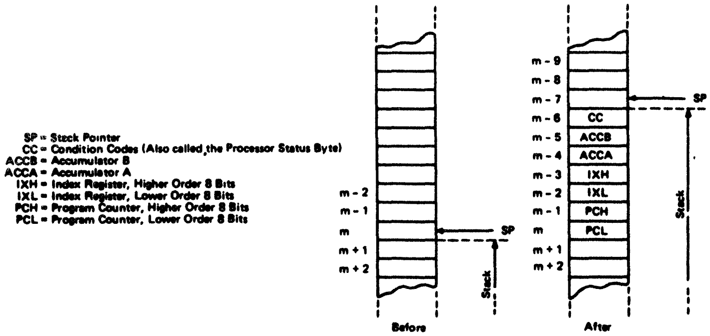

SP = Stack Pointer
CC = Condition Codes (Also called the Processor Status Byte)
ACCB = Accumulator B
ACCA = Accumulator A
IXH = Index Register, Higher Order 8 Bits
IXL = Index Register, Lower Order 8 Bits
PCH = Program Counter, Higher Order 8 Bits
PCL = Program Counter, Lower Order 8 Bits

Figure 11 Saving The Status of The Microprocessor in The Stack

■ HD6802 MPU SIGNAL DESCRIPTION

Proper operation of the MPU requires that certain control and timing signals be provided to accomplish specific functions and that other signal lines be monitored to determine the state of the processor. These control and timing signals for the HD6802 are similar to those of the HD6800 except that TSC, DBE, ϕ_1, ϕ_2 input, and two unused pins have been eliminated, and the following signal and timing lines have been added.

RAM Enable (RE)
Crystal Connections EXTAL and XTAL
Memory Ready(MR)
V_{CC} Standby
Enable ϕ_2 Output(E)

The following is a summary of the HD6802 MPU signals:

● Address Bus ($A_0 \sim A_{15}$)

Sixteen pins are used for the address bus. The outputs are capable of driving one standard TTL load and 90pF.

● Data Bus ($D_0 \sim D_7$)

Eight pins are used for the data bus. It is bidirectional, transferring data to and from the memory and peripheral devices. It also has three-state output buffers capable of driving one standard TTL load and 130pF.

Data Bus will be in the output mode when the internal RAM is accessed. This prohibits external data entering the MPU. It should be noted that the internal RAM is fully decoded from $0000 to $007F. External RAM at $0000 to $007F must be disabled when internal RAM is accessed.

● $\overline{\text{HALT}}$

When this input is in the "Low" state, all activity in the machine will be halted: This input is level sensitive.

In the halt mode, the machine will stop at the end of an instruction. Bus Available will be at a "High" state. Valid Memory Address will be at a "Low" state. The address bus will display the address of the next instruction.

To insure single instruction operation, transition of the $\overline{\text{HALT}}$ line must not occur during the last 250ns of E and the $\overline{\text{HALT}}$ line must go "High" for one Clock cycle.

$\overline{\text{HALT}}$ should be tied "High" if not used. This is good engineering design practice in general and necessary to insure proper operation of the part.

● Read/Write (R/$\overline{\text{W}}$)

This TTL compatible output signals the peripherals and memory devices whether the MPU is in a Read ("High") or Write ("Low") state. The normal standby state of this signal is Read ("High"). When the processor is halted, it will be in the logical one state ("High").

This output is capable of driving one standard TTL load and 90pF.

● Valid Memory Address (VMA)

This output indicates to peripheral devices that there is a valid address on the address bus. In normal operation, this signal should be utilized for enabling peripheral interfaces such as the PIA and ACIA. This signal is not three-state. One standard TTL load and 90pF may be directly driven by this active high signal.

● Bus Available (BA)

The Bus Available signal will normally be in the "Low" state. When activated, it will go to the "High" state indicating that the microprocessor has stopped and that the address bus is available (but not in a three-state condition). This will occur if the $\overline{\text{HALT}}$ line is in the "Low" state or the processor is in the wait state as a result of the execution of a WAI instruction. At such time, all three-state output drivers will go to their off state and other outputs to their normally inactive level.

The processor is removed from the wait state by the occurrence of a maskable (mask bit I=0) or nonmaskable interrupt. This output is capable of driving one standard TTL load and 30pF.

● Interrupt Request ($\overline{\text{IRQ}}$)

This level sensitive input requests that an interrupt sequence be generated within the machine. The processor will wait, until it completes the current instruction that is being executed before it recognizes the request. At that time, if the interrupt mask bit in the Condition Code Register is not set, the machine will begin an interrupt sequence. The index Register, Program Counter, Accumulators, and Condition Code Register are stored away on the stack. Next the MPU will respond to the interrupt request by setting the interrupt mask bit high so that no further interrupts may occur. At the end of the cycle, a 16-bit address will be loaded that points to a vectoring address which is located in memory locations FFF8 and FFF9. An address loaded at these locations causes the MPU to branch to an interrupt routine in memory.

The $\overline{\text{HALT}}$ line must be in the "High" state for interrupts to be serviced. Interrupts will be latched internally while $\overline{\text{HALT}}$ is "Low".

A 3kΩ external register to V_{CC} should be used for wire-OR and optimum control of interrupts.

● Reset ($\overline{\text{RES}}$)

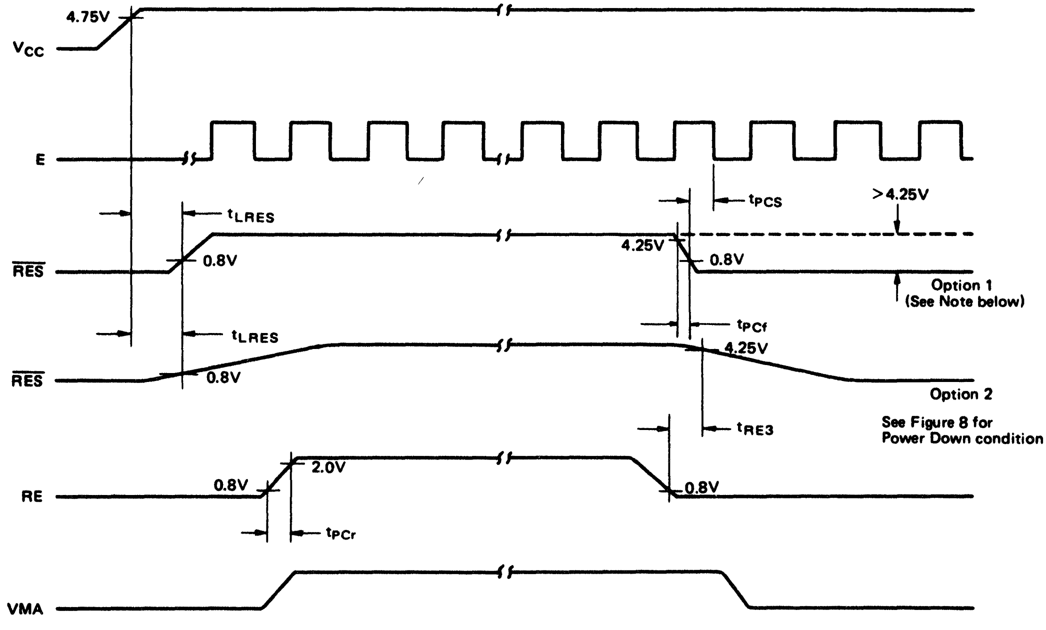
This input is used to reset and start the MPU from a power-down condition, resulting from a power failure or an initial start-up of the processor. When this line is "Low", the MPU is inactive and the information in the registers will be lost. If a "High" level is detected on the input, this will signal the MPU to begin the restart sequence. This will start execution of a routine to initialize the processor from its reset condition. All the higher order address lines will be forced "High". For the restart, the last two(FFFE, FFFF) locations in memory will be used to load the program that is addressed by the program counter. During the restart routine, the interrupt mask bit is set and must be reset before the MPU can be interrupted by $\overline{\text{IRQ}}$. Power-up and reset timing and power-down sequences are shown in Fig. 12 and Fig. 13 respectively.

● Non-Maskable Interrupt ($\overline{\text{NMI}}$)

A low-going edge on this input requests that a non-mask-interrupt sequence be generated within the processor. As with the $\overline{\text{IRQ}}$ signal, the processor will complete the current instruction that is being executed before it recognizes the $\overline{\text{NMI}}$ signal. The interrupt mask bit in the Condition Code Register has no effect on $\overline{\text{NMI}}$.

The Index Register, Program Counter, Accumulators, and Condition Code Register are stored away on the stack. At the end of the cycle, a 16-bit address will be loaded that points to a vectoring address which is located in memory locations FFFC and FFFD. An address loaded at these locations causes the MPU to branch to a non-maskable interrupt routine in memory. A 3kΩ external resistor to V_{CC} should be used for wire-OR and optimum control of interrupts.

Inputs $\overline{\text{IRQ}}$ and $\overline{\text{NMI}}$ are hardware interrupt lines that are sampled when E is "High" and will start the interrupt routine on a "Low" E following the completion of an instruction. $\overline{\text{IRQ}}$ and $\overline{\text{NMI}}$ should be tied "High" if not used. This is good engineering design practice in general and necessary to insure proper operation of the part. Fig. 14 is a flowchart describing the major decision paths and interrupt vectors of the microprocessor. Table 1 gives the memory map for interrupt vectors.



(NOTE) If option 1 is chosen, $\overline{\text{RES}}$ and RE pins can be tied together.

Figure 12 Power-up and Reset Timing

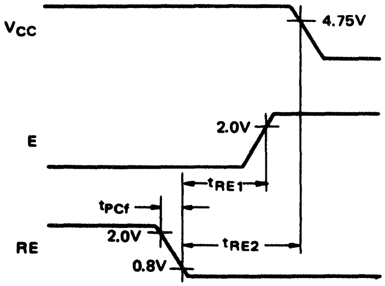


Figure 13 Power-down Sequence

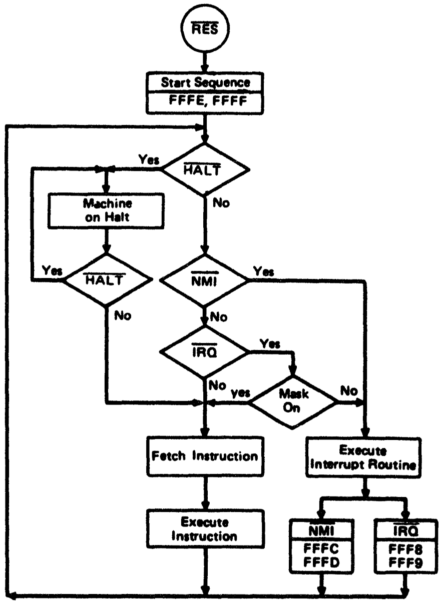


Figure 14 MPU Flow Chart

Table 1 Memory Map for Interrupt Vectors

Vector MS	Vector LS	Description	
FFFE	FFFF	Restart	($\overline{RES}$)
FFFC	FFFD	Non-Maskable Interrupt	($\overline{NMI}$)
FFFA	FFFB	Software Interrupt	(SWI)
FFF8	FFF9	Interrupt Request	($\overline{IRQ}$)

- **RAM Enable (RE)**

A TTL-compatible RAM enable input controls the on-chip RAM of the HD6802. When placed in the "High" state, the on-chip memory is enabled to respond to the MPU controls. In the "Low" state, RAM is disabled. This pin may also be utilized to disable reading and writing the on-chip RAM during a power-down situation. RAM enable must be "Low" three cycles before V_{CC} goes below 4.75V during power-down.

RE should be tied to the correct "High" or "Low" state if not used. This is good engineering design practice in general and necessary to insure proper operation of the part.

- **EXTAL and XTAL**

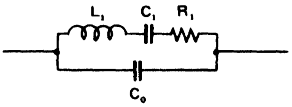
The HD6802 has an internal oscillator that may be crystal controlled. These connections are for a parallel resonant fundamental crystal (AT cut). A divide-by-four circuit has been added to the HD6802 so that a 4MHz crystal may be used in lieu of a 1MHz crystal for a more cost-effective system. Pin39 of the HD6802 may be driven externally by a TTL input signal if a separate clock is required. Pin38 is to be left open in this mode.

An RC network is not directly usable as a frequency source on pins 38 and 39. An RC network type TTL or CMOS oscillator will work well as long as the TTL or CMOS output drives the HD6802.

If an external clock is used, it may not be halted for more than 4.5μs. The HD6802 is a dynamic part except for the internal RAM, and requires the external clock to retain information.

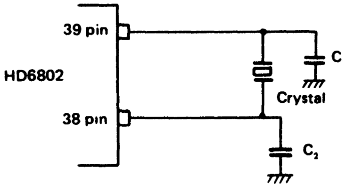
Conditions for Crystal (4 MHz)

- AT Cut Parallel resonant
- C_0 = 7 pF max.
- R_1 = 80Ω max.



Crystal Equivalent Circuit

Recommended Oscillator (4MHz)



$C_1 = C_2 = 22pF \pm 20\%$

Figure 15 Crystal Oscillator

When using the crystal, see the note for Board Design of the Oscillation Circuit in HD6802.

- **Memory Ready (MR)**

MR is a TTL compatible input control signal which allows stretching of E. When MR is "High", E will be in normal operation. When MR is "Low", E may be stretched integral multiples of half periods, thus allowing interface to slow memories. Memory Ready timing is shown in Fig. 16.

MR should be tied "High" if not used. This is good engineering design practice in general and necessary to insure proper operation of the part. A maximum stretch is 4.5μs.

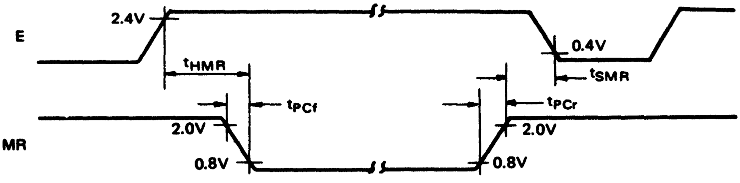


Figure 16 Memory Ready Control Function

- **Enable (E)**

This pin supplies the clock for the MPU and the rest of the system. This is a single phase, TTL compatible clock. This clock may be conditioned by a Memory Ready Signal. This is equivalent to ϕ_2 on the HD6800.

- **V_{CC} Standby**

This pin supplies the dc voltage to the first 32 bytes of RAM as well as the RAM Enable (RE) control logic. Thus retention of data in this portion of the RAM on a power up, power-down, or standby condition is guaranteed at the range of 4.0 V to 5.25 V. Maximum current drain at 5.25V is 8mA.

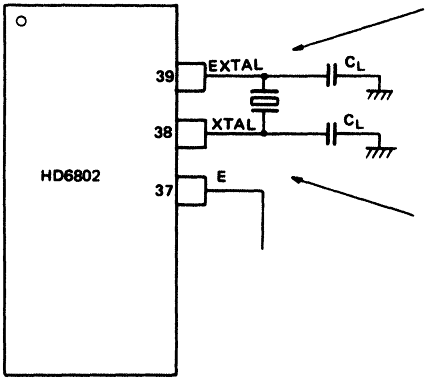
■ MPU INSTRUCTION SET

The HD6802 has a set of 72 different instructions. Included are binary and decimal arithmetic, logical, shift, rotate, load, store, conditional or unconditional branch, interrupt and stack manipulation instructions.

This instruction set is the same as that for the 6800MPU(HD6800 etc.) and is not explained again in this data sheet.

■ NOTE FOR BOARD DESIGN OF THE OSCILLATION CIRCUIT IN HD6802

In designing the board, the following notes should be taken when the crystal oscillator is used.

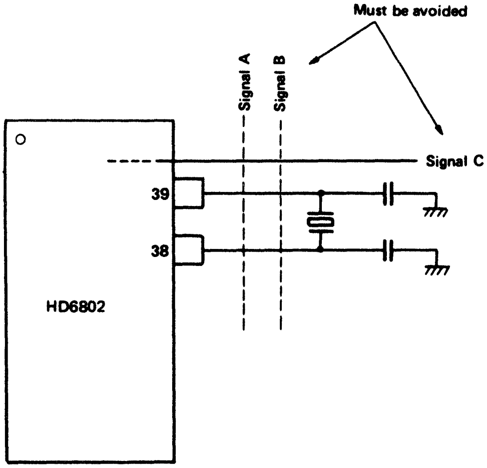


Crystal oscillator and load capacity C_L must be placed near the LSI as much as possible.

(Normal oscillation may be disturbed when external noise is induced to pin 38 and 39.)

Pin 38 signal line should be wired apart from pin 37 signal line as much as possible. Don't wire them in parallel, or normal oscillation may be disturbed when E signal is feedbacked to XTAL.

The following design must be avoided.



A signal line or a power source line must not cross or go near the oscillation circuit line as shown in the left figure to prevent the induction from these lines and perform the correct oscillation. The resistance among XTAL, EXTAL and other pins should be over 10MΩ.

Figure 17 Note for Board Design of the Oscillation Circuit

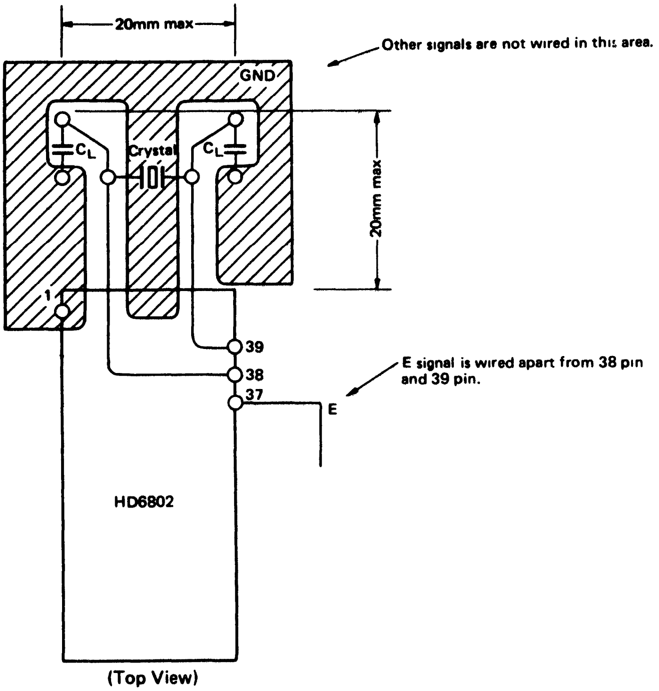

20mm max
Other signals are not wired in this area.
GND
C_L
Crystal
C_L
20mm max
1
39
38
37
E
E signal is wired apart from 38 pin and 39 pin.
HD6802
(Top View)

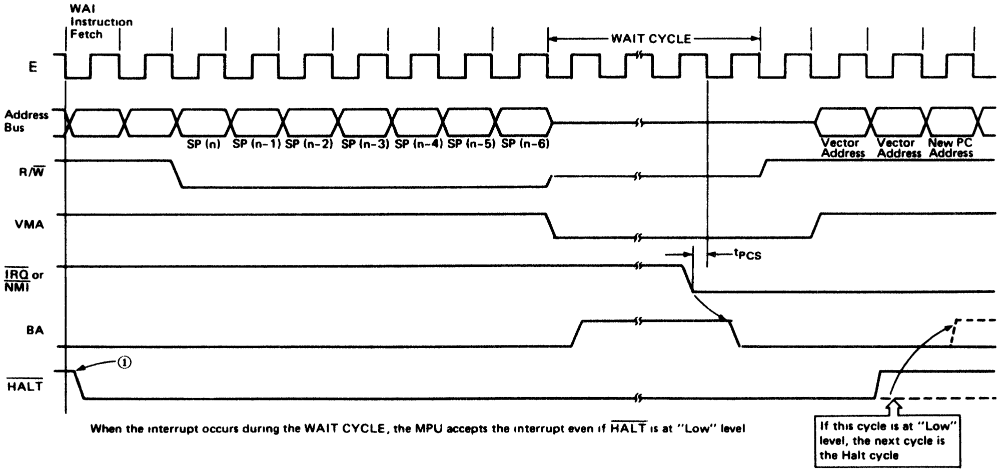
Figure 18 Example of Board Design Using the Crystal Oscillator

■ NOTE FOR THE RELATION BETWEEN WAI INSTRUCTION AND HALT OPERATION OF HD6802

When $\overline{\text{HALT}}$ input signal is asserted to "Low" level, the MPU will be halted after the execution of the current instruction except WAI instruction.

The "Halt" signal is not accepted after the fetch cycle of the WAI instruction (See ① in Fig. 19). In the case of the "WAI" instruction, the MPU enters the "WAIT" cycle after stacking the internal registers and outputs the "High" level on the BA line.

When an interrupt request signal is input to the MPU, the MPU accepts the interrupt regardless the "Halt" signal and releases the "WAIT" state and outputs the interrupt's vector address. If the "Halt" signal is "Low" level, the MPU halts after the fetch of new PC contents. The sequense is shown below.



When the interrupt occurs during the WAIT CYCLE, the MPU accepts the interrupt even if $\overline{\text{HALT}}$ is at "Low" level

If this cycle is at "Low" level, the next cycle is the Halt cycle

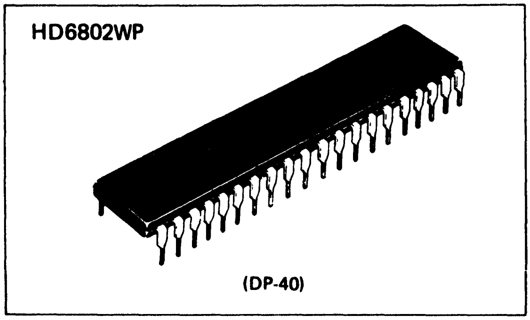
Figure 19 HD6802 WAIT CYCLE & $\overline{\text{HALT}}$ Request

HD6802W
MPU (Microprocessor with Clock and RAM)

HD6802W is the enhanced version of HD6802 which contains MPU, clock and 256 bytes RAM. Internal RAM has been extended from 128 to 256 bytes to increase the capacity of system read/write memory for handling temporary data and manipulating the stack.

The internal RAM is located at hex addresses 0000 to 00FF. The first 32 bytes of RAM, at hex addresses 0000 to 001F, may be retained in a low power mode by utilizing V_{CC} standby, thus facilitating memory retention during a power-down situation.

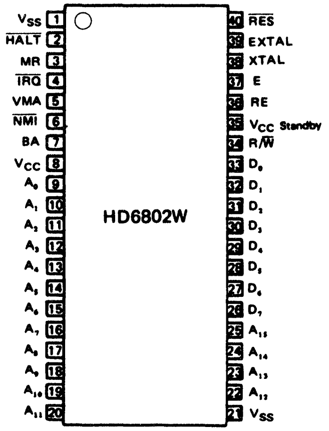
The HD6802W is completely software compatible with the HD6800 as well as the entire HMCS6800 family of parts. Hence, the HD6802W is expandable to 65k words.



■ FEATURES

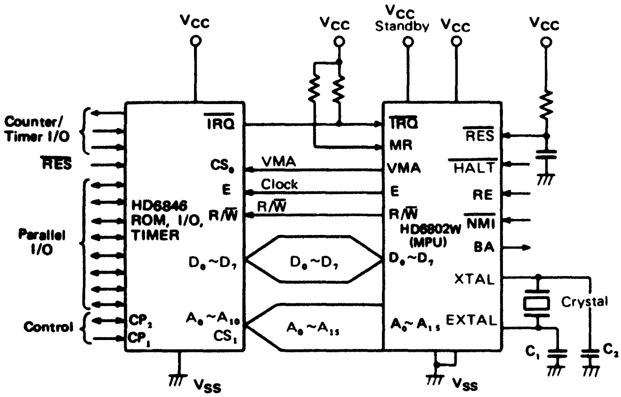
- On-Chip Clock Circuit
- 256 × 8 Bit On-Chip RAM
- 32 Bytes of RAM are Retainable
- Software-Compatible with the HD6800, HD6802
- Expandable to 65k words
- Standard TTL-Compatible Inputs and Outputs
- 8 Bit Word Size
- 16 Bit Memory Addressing
- Interrupt Capability

■ PIN ARRANGEMENT



(Top View)

■ BLOCK DIAGRAM



A expanded block diagram of the HD6802W is shown in Fig. 1. As shown, the number and configuration of the registers are the same as the HD6802 except that the internal RAM has been extended to 256 bytes.

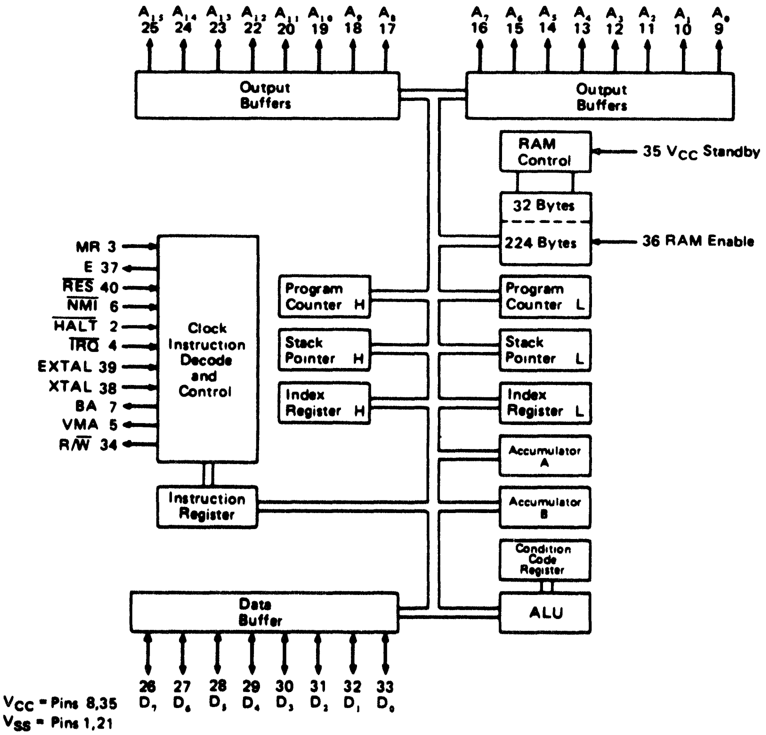


Figure 1 Expanded Block Diagram

Address Map of RAM is shown is Fig. 2.

The HD6802W has 256 bytes of RAM on the chip located at hex addresses 0000 to 00FF. The first 32 bytes of RAM, at hex addresses 0000 to 001F, may be retained in a low power mode by utilizing V_{CC} standby and setting RAM Enable Signal "Low" level, thus facilitating memory retention during a power-down situation.

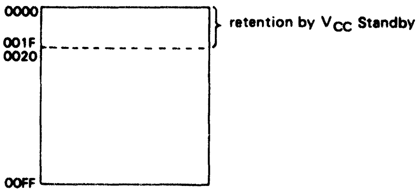


Figure 2 Address Map of HD6802W

■ ABSOLUTE MAXIMUM RATINGS

Item	Symbol	Value	Unit
Supply Voltage	V_{CC} * V_{CC} Standby*	-0.3 ~ +7.0	V
Input Voltage	V_{in}*	-0.3 ~ +7.0	V
Operating Temperature	T_{opr}	-20 ~ +75	°C
Storage Temperature	T_{stg}	-55 ~ +150	°C

* With respect to V_{SS} (SYSTEM GND)

(NOTE) Permanent LSI damage may occur if maximum ratings are exceeded. Normal operation should be under recommended operating conditions. If these conditions are exceeded, it could affect reliability of LSI.

■ RECOMMENDED OPERATING CONDITIONS

Item	Symbol		min	typ	max	Unit
Supply Voltage	V_{CC}*		4.75	5.0	5.25	V
	V_{CC} Standby*		4.0			
Input Voltage	V_{IL}*		-0.3	–	0.8	V
	V_{IH}*	Except $\overline{RES}$	2.0	–	V_{CC}	V
		$\overline{RES}$	V_{CC} -0.75	–	V_{CC}	
Operation Temperature	T_{opr}		-20	25	75	°C

* With respect to V_{SS} (SYSTEM GND)

■ ELECTRICAL CHARACTERISTICS

● DC CHARACTERISTICS (V_{CC}=5.0V±5%, V_{CC} Standby=5.0V±5%, V_{SS}=0V, Ta=-20~+75°C, unless otherwise noted.)

Item		Symbol	Test Condition	min	typ*	max	Unit
Input "High" Voltage	Except $\overline{RES}$	V_{IH}		2.0	–	V_{CC}	V
	$\overline{RES}$			Vcc-0.75	–	V_{CC}	
Input "Low" Voltage	Except $\overline{RES}$	V_{IL}**		-0.3	–	0.8	V
	$\overline{RES}$			-0.3	–	0.8	
Output "High" Voltage	D_0~D_7, E	V_{OH}	I_{OH} = -205μA	2.4	–	–	V
	A_0~A_{15}, R/$\overline{W}$, VMA		I_{OH} = -145μA	2.4	–	–	
	BA		I_{OH} = -100μA	2.4	–	–	
Output "Low" Voltage		V_{OL}	I_{OL} = 1.6mA	–	–	0.4	V
Three State (Off State) Input Current	D_0~D_7	I_{TSI}	V_{in} = 0.4~2.4V	-10	–	10	μA
Input Leakage Current	Except D_0~D_7	Iin ***	V_{in} = 0~5.25V	-2.5	–	2.5	μA
Power Dissipation		P_D****		–	0.7	1.2	W
Input Capacitance	D_0~D_7	C_{in}	V_{in}=0V, T_a=25°C, f=1.0MHz	–	10	12.5	pF
	Except D_0~D_7			–	6.5	10	
Output Capacitance	A_0~A_{15}, R/$\overline{W}$, BA, VMA	C_{out}	V_{in}=0V, T_a=25°C, f=1.0MHz	–	–	12	pF

* T_a=25°C, V_{CC}=5V

** As $\overline{RES}$ input has histeresis character, applied voltage up to 2.4V is regarded as "Low" level when it goes up from 0V.

*** Does not include EXTAL and XTAL, which are crystal inputs.

**** In power-down mode, maximum power dissipation is less than 42mW.

- **AC CHARACTERISTICS (V_{CC}=5.0V±5%, V_{CC} Standby=5.0V±5%, V_{SS}=0V, Ta=-20~+75°C, unless otherwise noted.)**

1. CLOCK TIMING CHARACTERISTICS

Item		Symbol	Test Condition	min	typ	max	Unit
Frequency of Operation	Input Clock ÷ 4	f		0.1	–	1.0	MHz
	Crystal Frequency	f_{XTAL}		1.0	–	4.0	
Cycle Time		t_{cyc}	Fig. 4, Fig. 5	1.0	–	10	μs
Clock Pulse Width	"High" Level	$PW_{\phi H}$	at 2.4V (Fig. 4, Fig. 5)	450	–	4500	ns
	"Low" Level	$PW_{\phi L}$	at 0.4V (Fig. 4, Fig. 5)				
Clock Fall Time		t_{ϕ}	0.4V ~ 2.4V(Fig.4,Fig.5)	–	–	25	ns

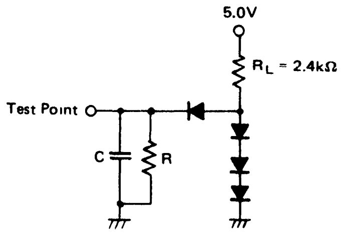
2. READ/WRITE TIMING

Item	Symbol	Test Condition	min	typ*	max	Unit
Address Delay	t_{AD}	Fig. 4, Fig. 5, Fig. 8	–	–	270	ns
Peripheral Read Access Time	t_{acc}	Fig. 4	530	–	–	ns
Data Setup Time (Read)	t_{DSR}	Fig. 4	100	–	–	ns
Input Data Hold Time	t_{H}	Fig. 4	10	–	–	ns
Output Data Hold Time	t_{H}	Fig. 5	20	–	–	ns
Address Hold Time (Address, R/$\overline{W}$, VMA)	t_{AH}	Fig. 4, Fig. 5	10	–	–	ns
Data Delay Time (Write)	t_{DDW}	Fig. 5	–	–	225	ns
Bus Available Delay	t_{BA}	Fig. 6, Fig. 7, Fig. 9, Fig. 10	–	–	250	ns
Processor Controls **Processor Control Setup Time**	t_{PCS}	Fig. 6 ~ Fig. 9, Fig. 11	200	–	–	ns
Processor Control Rise and Fall Time (Measured at 0.8V and 2.0V)	t_{PCr}, t_{PCf}	Fig. 6 ~ Fig. 9, Fig. 11, Fig. 12, Fig. 14	–	–	100	ns

* Ta = 25°C, V_{CC} = 5V

3. POWER DOWN SEQUENCE TIMING, POWER UP RESET TIMING AND MEMORY READY TIMING

Item	Symbol	Test Condition	min	typ	max	Unit
RAM Enable Reset Time (1)	t_{RE1}	Fig. 12	150	–	–	ns
RAM Enable Reset Time (2)	t_{RE2}	Fig. 12	E-3 cycles	–	–	
Reset Release Time	t_{LRES}	Fig. 11	20	–	–	ms
RAM Enable Reset Time (3)	t_{RE3}	Fig. 11	0	–	–	ns
Memory Ready Setup Time	t_{SMR}	Fig. 14	300	–	–	ns
Memory Ready Hold Time	t_{HMR}	Fig. 14	0	–	200	ns



C = 130pF for D_0~D_7, E
= 90pF for A_0~A_{15}, R/$\overline{W}$, and VMA
= 30pF for BA
R = 11kΩ for D_0~D_7, E
= 16kΩ for A_0~A_{15}, R/$\overline{W}$, and VMA
= 24kΩ for BA
C includes stray Capacitance.
All diodes are 1S2074Ⓗ or equivalent.

Figure 3 Bus Timing Test Load

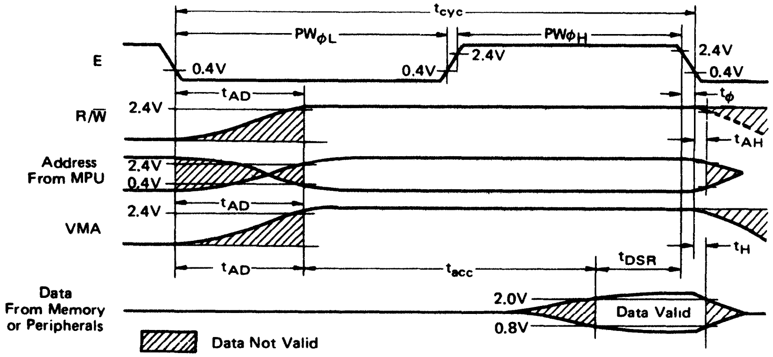


Figure 4 Read Data from Memory or Peripherals

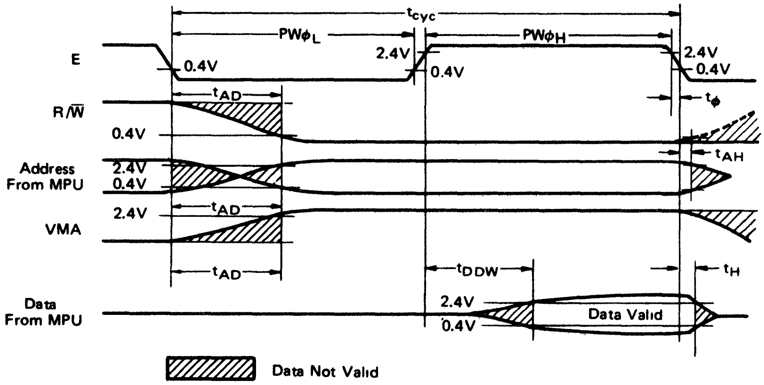


Figure 5 Write Data in Memory or Peripherals

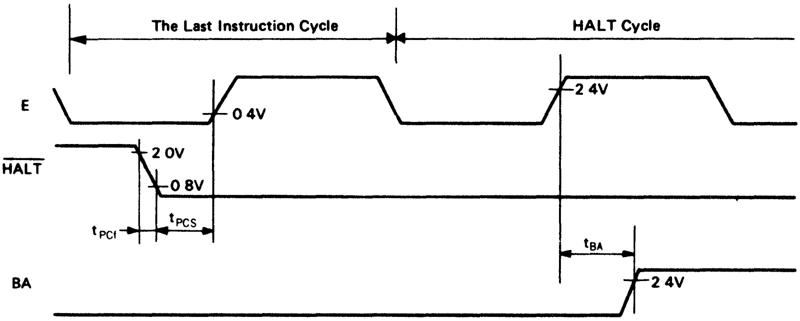


Figure 6 Timing of $\overline{\text{HALT}}$ and BA

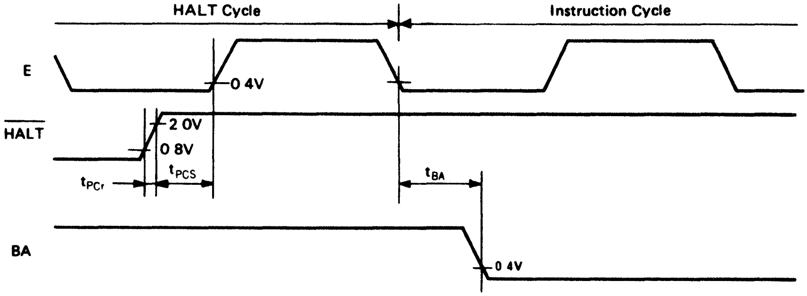


Figure 7 Timing of $\overline{\text{HALT}}$ and BA

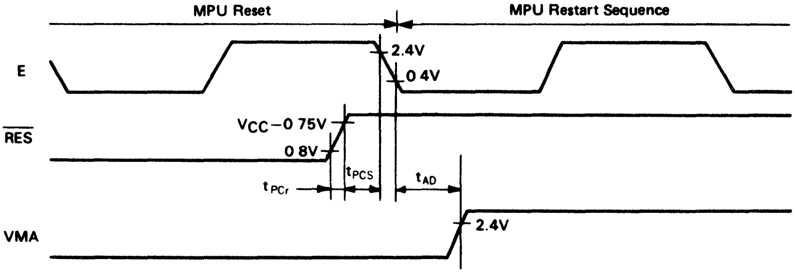


Figure 8 $\overline{\text{RES}}$ and MPU Restart Sequence

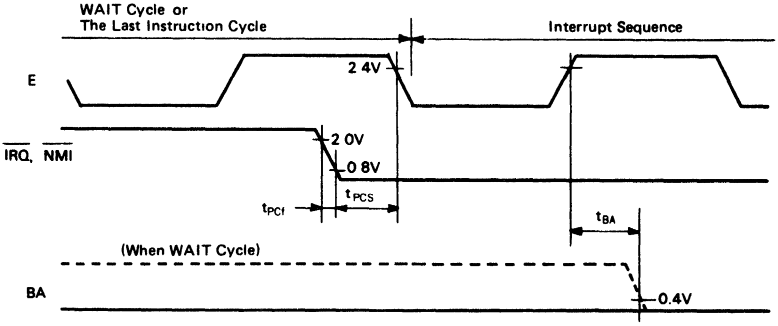


Figure 9 IRQ and NMI Interrupt Timing

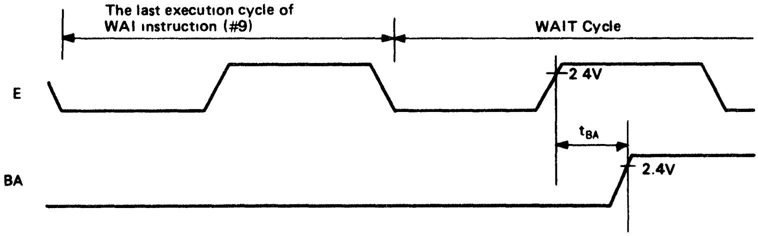


Figure 10 WAI Instruction and BA Timing

■ HD6802W MPU SIGNAL DESCRIPTION

- **Address Bus ($A_0 \sim A_{15}$)**

Sixteen pins are used for the address bus. The outputs are capable of driving one standard TTL load and 90pF.

- **Data Bus ($D_0 \sim D_7$)**

Eight pins are used for the data bus. It is bidirectional, transferring data to and from the memory and peripheral devices. It also has three-state output buffers capable of driving one standard TTL load and 130pF.

Data Bus will be in the output mode when the internal RAM is accessed. This prohibits external data entering the MPU. It should be noted that the internal RAM is fully decoded from \$0000 to \$00FF. External RAM at \$0000 to \$00FF must be disabled when internal RAM is accessed.

- **$\overline{\text{HALT}}$**

When this input is in the "Low" state, all activity in the machine will be halted: This input is level sensitive.

In the halt mode, the machine will stop at the end of an instruction. Bus Available will be at a "High" state. Valid Memory Address will be at a "Low" state. The address bus will display the address of the next instruction.

To insure single instruction operation, transition of the $\overline{\text{HALT}}$ line must not occur during the last t_{PCS} of E and the $\overline{\text{HALT}}$ line must go "High" for one Clock cycle.

$\overline{\text{HALT}}$ should be tied "High" if not used. This is good engineering design practice in general and necessary to insure proper operation of the part.

- **Read/Write (R/$\overline{\text{W}}$)**

This TTL compatible output signals the peripherals and memory devices whether the MPU is in a Read ("High") or Write ("Low") state. The normal standby state of this signal is Read ("High"). When the processor is halted, it will be in the logical one state ("High").

This output is capable of driving one standard TTL load and 90pF.

- **Valid Memory Address (VMA)**

This output indicates to peripheral devices that there is a valid address on the address bus. In normal operation, this signal should be utilized for enabling peripheral interfaces such as the PIA and ACIA. This signal is not three-state. One standard TTL load and 90pF may be directly driven by this active high signal.

- **Bus Available (BA)**

The Bus Available signal will normally be in the "Low" state. When activated, it will go to the "High" state indicating that the microprocessor has stopped and that the address bus is available (but not in a three-state condition). This will occur if the $\overline{\text{HALT}}$ line is in the "Low" state or the processor is in the wait state as a result of the execution of a WAI instruction. At such time, all three-state output drivers will go to their off state and other outputs to their normally inactive level.

The processor is removed from the wait state by the occurrence of a maskable (mask bit I=0) or nonmaskable interrupt. This output is capable of driving one standard TTL load and 30pF.

- **Interrupt Request ($\overline{\text{IRQ}}$)**

This level sensitive input requests that an interrupt sequence be generated within the machine. The processor will wait, until it completes the current instruction that is being executed before it recognizes the request. At that time, if the interrupt mask bit in the Condition Code Register is not set, the machine will begin an interrupt sequence. The index Register, Program Counter, Accumulators, and Condition Code Register are stored away on the stack. Next the MPU will respond to the interrupt request by setting the interrupt mask bit high so that no further interrupts may occur. At the end of the cycle, a 16-bit address will be loaded that points to a vectoring address which is located in memory locations FFF8 and FFF9. An address loaded at these locations causes the MPU to branch to an interrupt routine in memory.

The $\overline{\text{HALT}}$ line must be in the "High" state for interrupts to be serviced. Interrupts will be latched internally while $\overline{\text{HALT}}$ is "Low".

A 3kΩ external register to V_{CC} should be used for wire-OR and optimum control of interrupts.

- **Reset ($\overline{\text{RES}}$)**

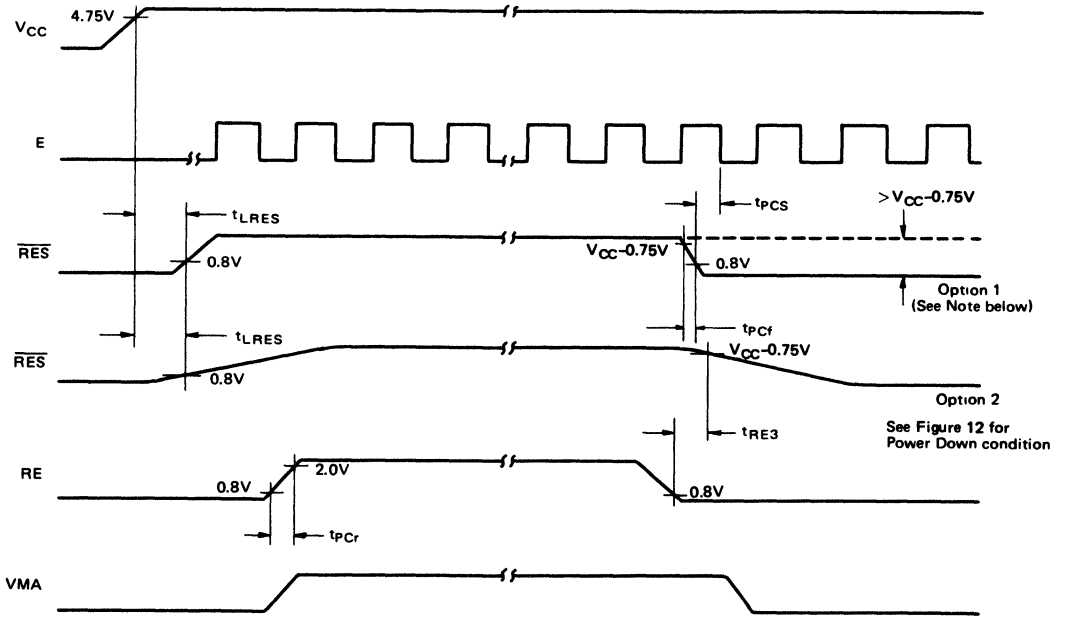
This input is used to reset and start the MPU from a power-down condition, resulting from a power failure or an initial start-up of the processor. When this line is "Low", the MPU is inactive and the information in the registers will be lost. If a "High" level is detected on the input, this will signal the MPU to begin the restart sequence. This will start execution of a routine to initialize the processor from its reset condition. All the higher order address lines will be forced "High". For the restart, the last two(FFFE, FFFF) locations in memory will be used to load the program that is addressed by the program counter. During the restart routine, the interrupt mask bit is set and must be reset before the MPU can be interrupted by $\overline{\text{IRQ}}$. Power-up and reset timing and power-down sequences are shown in Fig. 11 and Fig. 12 respectively.

- **Non-Maskable Interrupt ($\overline{\text{NMI}}$)**

A low-going edge on this input requests that a non-mask-interrupt sequence be generated within the processor. As with the $\overline{\text{IRQ}}$ signal, the processor will complete the current instruction that is being executed before it recognizes the $\overline{\text{NMI}}$ signal. The interrupt mask bit in the Condition Code Register has no effect on $\overline{\text{NMI}}$.

The Index Register, Program Counter, Accumulators, and Condition Code Register are stored away on the stack. At the end of the cycle, a 16-bit address will be loaded that points to a vectoring address which is located in memory locations FFFC and FFFD. An address loaded at these locations causes the MPU to branch to a non-maskable interrupt routine in memory. A 3kΩ external resistor to V_{CC} should be used for wire-OR and optimum control of interrupts.

Inputs $\overline{\text{IRQ}}$ and $\overline{\text{NMI}}$ are hardware interrupt lines that are sampled when E is "High" and will start the interrupt routine on a "Low" E following the completion of an instruction. $\overline{\text{IRQ}}$ and $\overline{\text{NMI}}$ should be tied "High" if not used. This is good engineering design practice in general and necessary to insure proper operation of the part. Fig. 13 is a flowchart describing the major decision paths and interrupt vectors of the microprocessor. Table 1 gives the memory map for interrupt vectors.



(NOTE) If option 1 is chosen, $\overline{\text{RES}}$ and RE pins can be tied together.

Figure 11 Power-up and Reset Timing

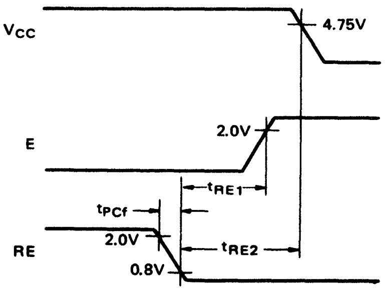


Figure 12 Power-down Sequence

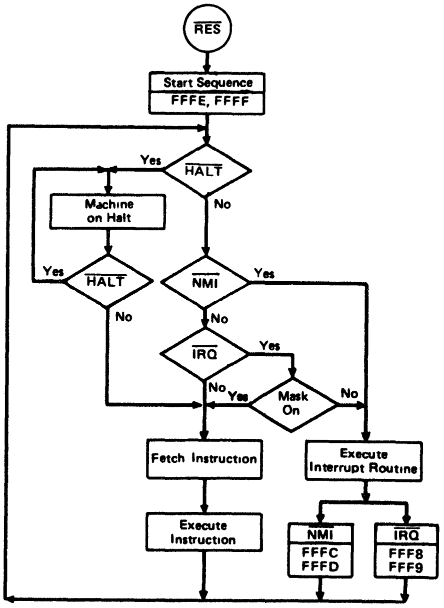


Figure 13 MPU Flow Chart

Table 1 Memory Map for Interrupt Vectors

Vector MS	LS	Description	
FFFE	FFFF	Restart	($\overline{\text{RES}}$)
FFFC	FFFD	Non-Maskable Interrupt	($\overline{\text{NMI}}$)
FFFA	FFFB	Software Interrupt	(SWI)
FFF8	FFF9	Interrupt Request	($\overline{\text{IRQ}}$)

- **RAM Enable (RE)**

A TTL-compatible RAM enable input controls the on-chip RAM of the HD6802W. When placed in the "High" state, the on-chip memory is enabled to respond to the MPU controls. In the "Low" state, RAM is disabled. This pin may also be utilized to disable reading and writing the on-chip RAM during a power-down situation. RAM enable must be "Low" three cycles before V_{CC} goes below 4.75V during power-down.

RE should be tied to the correct "High" or "Low" state if not used. This is good engineering design practice in general and necessary to insure proper operation of the part.

- **EXTAL and XTAL**

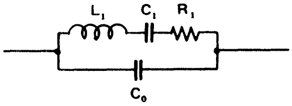
The HD6802W has an internal oscillator that may be crystal controlled. These connections are for a parallel resonant fundamental crystal (AT cut). A divide-by-four circuit has been added to the HD6802W so that a 4MHz crystal may be used in lieu of a 1MHz crystal for a more cost-effective system. Pin39 of the HD6802W may be driven externally by a TTL input signal if a separate clock is required. Pin38 is to be left open in this mode.

An RC network is not directly usable as a frequency source on pins 38 and 39. An RC network type TTL or CMOS oscillator will work well as long as the TTL or CMOS output drives the HD6802W.

If an external clock is used, it may not be halted for more than 4.5μs. The HD6802W is a dynamic part except for the internal RAM, and requires the external clock to retain information.

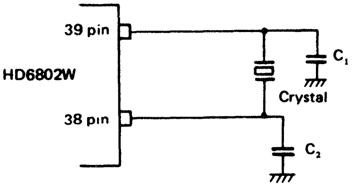
Conditions for Crystal (4 MHz)

- AT Cut Parallel resonant
- C_0 = 7 pF max.
- R_1 = 80Ω max.



Crystal Equivalent Circuit

Recommended Oscillator (4MHz)



C_1 = C_2 = 22pF ± 20%

When using the crystal, see the note for Board Design of the Oscillation Circuit in HD6802W.

- **Memory Ready (MR)**

MR is a TTL compatible input control signal which allows stretching of E. When MR is "High", E will be in normal operation. When MR is "Low", E may be stretched integral multiples of half periods, thus allowing interface to slow memories. Memory Ready timing is shown in Fig. 14.

MR should be tied "High" if not used. This is good engineering design practice in general and necessary to insure proper operation of the part. A maximum stretch is 4.5μs.

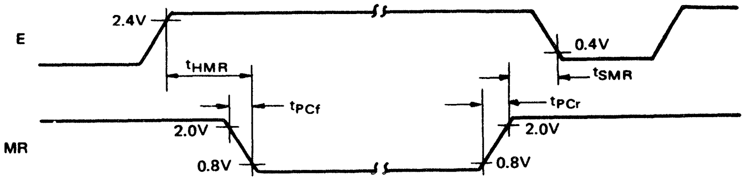


Figure 14 Memory Ready Control Function

- **Enable (E)**

This pin supplies the clock for the MPU and the rest of the system. This is a single phase, TTL compatible clock. This clock may be conditioned by a Memory Ready Signal. This is equivalent to ϕ_2 on the HD6800.

- **V_{CC} Standby**

This pin sùpplies the dc voltage to the first 32 bytes of RAM as well as the RAM Enable (RE) control logic. Thus retention of data in this portion of the RAM on a power-up, power-down, or standby condition is guaranteed at the range of 4.0 V to 5.25 V.

Maximum current drain at 5.25V is 8mA.

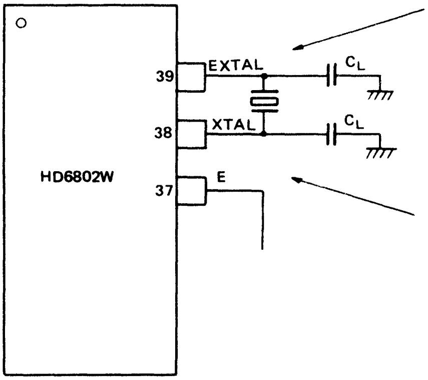
■ MPU INSTRUCTION SET

The HD6802W has a set of 72 different instructions. Included are binary and decimal arithmetic, logical, shift, rotate, load, store, conditional or unconditional branch, interrupt and stack manipulation instructions.

This instruction set is the same as that for the 6800MPU (HD6800 etc.) and is not explained again in this data sheet.

■ NOTE FOR BOARD DESIGN OF THE OSCILLATION CIRCUIT IN HD6802W

In designing the board, the following notes should be taken when the crystal oscillator is used.

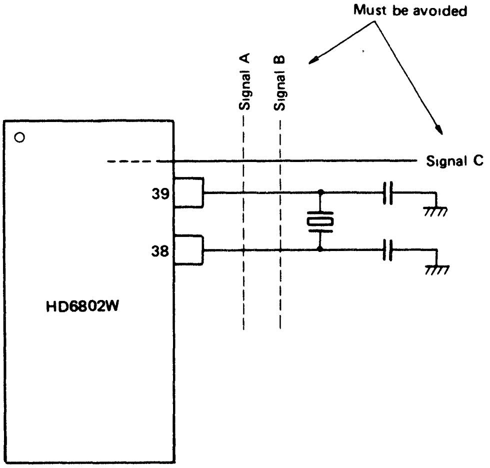


Crystal oscillator and load capacity C_L must be placed near the LSI as much as possible.

(Normal oscillation may be disturbed when external noise is induced to pin 38 and 39.)

Pin 38 signal line should be wired apart from pin 37 signal line as much as possible. Don't wire them in parallel, or normal oscillation may be disturbed when E signal is feedbacked to XTAL.

The following design must be avoided.



A signal line or a power source line must not cross or go near the oscillation circuit line as shown in the left figure to prevent the induction from these lines and perform the correct oscillation. The resistance among XTAL, EXTAL and other pins should be over 10MΩ.

Figure 15 Note for Board Design of the Oscillation Circuit

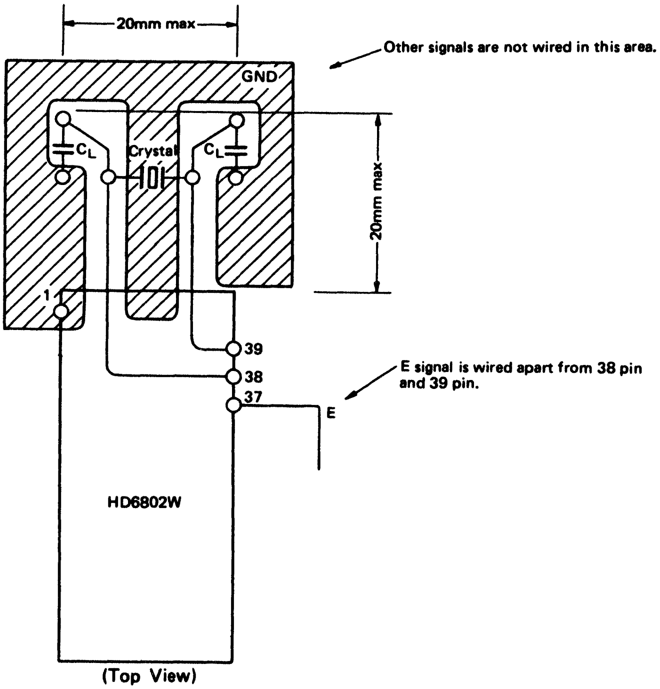

20mm max
Other signals are not wired in this area.
GND
C_L
Crystal
C_L
20mm max
1
39
38
37
E
E signal is wired apart from 38 pin and 39 pin.
HD6802W
(Top View)

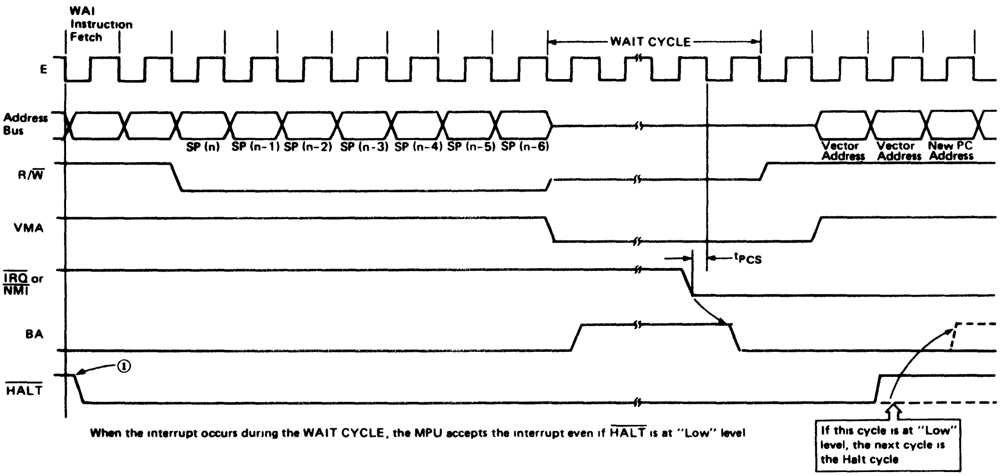
Figure 16 Example of Board Design Using the Crystal Oscillator

■ NOTE FOR THE RELATION BETWEEN WAI INSTRUCTION AND HALT OPERATION OF HD6802W

When $\overline{\text{HALT}}$ input signal is asserted to "Low" level, the MPU will be halted after the execution of the current instruction except WAI instruction.

The "Halt" signal is not accepted after the fetch cycle of the WAI instruction (See ① in Fig. 17). In the case of the "WAI" instruction, the MPU enters the "WAIT" cycle after stacking the internal registers and outputs the "High" level on the BA line.

When an interrupt request signal is input to the MPU, the MPU accepts the interrupt regardless the "Halt" signal and releases the "WAIT" state and outputs the interrupt's vector address. If the "Halt" signal is "Low" level, the MPU halts after the fetch of new PC contents. The sequense is shown below.



When the interrupt occurs during the WAIT CYCLE, the MPU accepts the interrupt even if $\overline{\text{HALT}}$ is at "Low" level

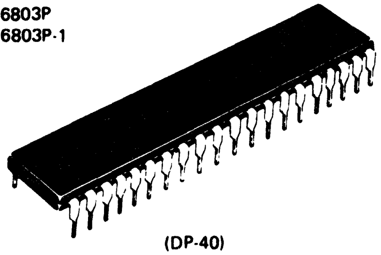
If this cycle is at "Low" level, the next cycle is the Halt cycle

Figure 17 HD6802W WAIT CYCLE & $\overline{\text{HALT}}$ Request

HD6803, HD6803-1
MPU (Micro Processing Unit)

The HD6803 MPU is an 8-bit micro processing unit which is compatible with the HMCS6800 family of parts. The HD6803 MPU is object code compatible with the HD6800 with improved execution times of key instructions plus several new 16-bit and 8-bit instruction including an 8 × 8 unsigned multiply with 16-bit result. The HD6803 MPU can be expanded to 65k bytes. The HD6803 MPU is TTL compatible and requires one +0.5 volt power supply. The HD6803 MPU has 128 bytes of RAM, Serial Communications Interface (S.C.I.), and parallel I/O as well as a three function 16-bit timer. Features and Block Diagram of the HD6803 include the following:

HD6803P
HD6803P-1

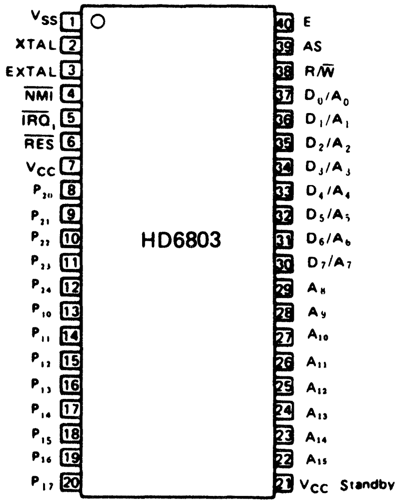


(DP-40)

■ FEATURES

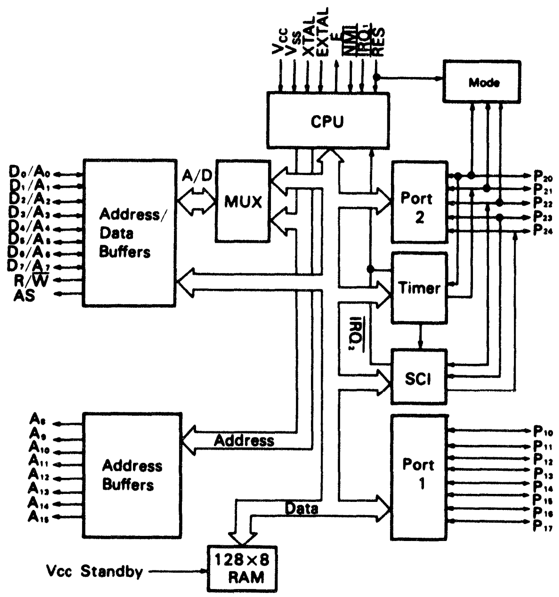
- Expanded HMCS6800 Instruction Set
- 8 × 8 Multiply
- On-Chip Serial Communications Interface (S.C.I.)
- Object Code Compatible with The HD6800 MPU
- 16-Bit Timer
- Expandable to 65k Bytes
- Multiplexed Address and Data
- 128 Bytes of RAM (64 Bytes Retainable On Power Down)
- 13 Parallel I/O Lines
- Internal Clock/Divided-By-Four
- TTL Compatible Inputs and Outputs
- Interrupt Capability
- Compatible with MC6803 and MC6803-1

■ PIN ARRANGEMENT



(Top View)

■ BLOCK DIAGRAM



■ TYPE OF PRODUCTS

Type No.	Bus Timing
HD6803	1.0MHz
HD6803-1	1.25MHz

■ ABSOLUTE MAXIMUM RATINGS

Item	Symbol	Value	Unit
Supply Voltage	V_{CC} *	-0.3 ~ +7.0	V
Input Voltage	V_{in} *	-0.3 ~ +7.0	V
Operating Temperature	T_{opr}	0 ~ + 70	°C
Storage Temperature	T_{stg}	- 55 ~ +150	°C

* With respect to V_{SS} (SYSTEM GND)

[NOTE] Permanent LSI damage may occur if maximum ratings are exceeded. Normal operation should be under recommended operating conditions. If these conditions are exceeded, it could affect reliability of LSI.

■ ELECTRICAL CHARACTERISTICS

● DC CHARACTERISTICS (V_{CC} =5.0V±5%, V_{SS} = 0V, Ta = 0~+70°C, unless otherwise noted.)

Item		Symbol	Test Condition	min	typ	max	Unit
Input "High" Voltage	$\overline{RES}$	V_{IH}		4.0	–	V_{CC}	V
	Other Inputs*			2.0	–	V_{CC}	
Input "Low" Voltage	All Inputs*	V_{IL}		-0.3	–	0.8	V
Input Load Current	EXTAL	$\|I_{in}\|$	V_{in} = 0 ~ V_{CC}	–	–	0.8	mA
Input Leakage Current	$\overline{NMI}$, $\overline{IRQ_1}$, $\overline{RES}$	$\|I_{in}\|$	V_{in} = 0 ~ 5.25V	–	–	2.5	μA
Three State (Offset) Leakage Current	P_{10} ~ P_{17}, D_0/A_0 ~ D_7/A_7	$\|I_{TSI}\|$	V_{in} = 0.5 ~ 2.4V	–	–	10	μA
	P_{20} ~ P_{24}			–	–	100	
Output "High" Voltage	D_0/A_0 ~ D_7/A_7	V_{OH}	I_{LOAD} = -205 μA	2.4	–	–	V
	A_8 ~ A_{15}, E, R/$\overline{W}$, AS		I_{LOAD} = -145 μA	2.4	–	–	
	Other Outputs		I_{LOAD} = -100 μA	2.4	–	–	
Output "Low" Voltage	All Outputs	V_{OL}	I_{LOAD} = 1.6 mA	–	–	0.5	V
Darlington Drive Current	P_{10} ~ P_{17}	$-I_{OH}$	V_{out} = 1.5V	1.0	–	10.0	mA
Power Dissipation		P_D		–	–	1200	mW
Input Capacitance	D_0/A_0 ~ D_7/A_7	C_{in}	V_{in} = 0V, Ta = 25°C, f = 1.0 MHz	–	–	12.5	pF
	Other Inputs			–	–	10.0	
V_{CC} Standby	Powerdown	V_{SBB}		4.0	–	5.25	V
	Operating	V_{SB}		4.75	–	5.25	
Standby Current	Powerdown	I_{SBB}	V_{SBB} = 4.0V	–	–	8.0	mA

*Except Mode Programming Levels.

- **AC CHARACTERISTICS**

BUS TIMING (V_{CC} = 5.0V ± 5%, V_{SS} = 0V, Ta = 0 ~ +70°C, unless otherwise noted.)

Item		Symbol	Test Condi-tion	HD6803			HD6803-1			Unit
				min	typ	max	min	typ	max	
Cycle Time		t_{cyc}	Fig. 1	1	–	10	0.8	–	10	μs
Address Strobe Pulse Width "High" *		PW_{ASH}		200	–	–	150	–	–	ns
Address Strobe Rise Time		t_{ASr}		5	–	50	5	–	50	ns
Address Strobe Fall Time		t_{ASf}		5	–	50	5	–	50	ns
Address Strobe Delay Time *		t_{ASD}		60	–	–	30	–	–	ns
Enable Rise Time		t_{Er}		5	–	50	5	–	50	ns
Enable Fall Time		t_{Ef}		5	–	50	5	–	50	ns
Enable Pulse Width "High" Time *		PW_{EH}		450	–	–	340	–	–	ns
Enable Pulse Width "Low" Time *		PW_{EL}		450	–	–	350	–	–	ns
Address Strobe to Enable Delay Time *		t_{ASED}		60	–	–	30	–	–	ns
Address Delay Time		t_{AD}		–	–	260	–	–	260	ns
Address Delay Time for Latch *		t_{ADL}		–	–	270	–	–	260	ns
Data Set-up Write Time		t_{DSW}		225	–	–	115	–	–	ns
Data Set-up Read Time		t_{DSR}		80	–	–	70	–	–	ns
Data Hold Time	Read	t_{HR}		10	–	–	10	–	–	ns
	Write	t_{HW}		20	–	–	20	–	–	
Address Set-up Time for Latch *		t_{ASL}		60	–	–	50	–	–	ns
Address Hold Time for Latch		t_{AHL}		20	–	–	20	–	–	ns
Address Hold Time		t_{AH}		20	–	–	20	–	–	ns
Peripheral Read Access Time (Multiplexed Bus)*		(t_{ACCM})		–	–	(600)	–	–	(420)	ns
Oscillator stabilization Time		t_{RC}	Fig. 8	100	–	–	100	–	–	ms
Processor Control Set-up Time		t_{PCS}	Fig. 7,8	200	–	–	200	–	–	ns

*These timings change in approximate proportion to t_{cyc}. The figures in this characteristics represent those when t_{cyc} is minimum (= in the highest speed operation).

PERIPHERAL PORT TIMING (V_{CC} = 5.0V ± 5%, V_{SS} = 0V, Ta = 0 ~ +70°C, unless otherwise noted.)

Item		Symbol	Test Condition	min	typ	max	Unit
Peripheral Data Setup Time	Port 1, 2	t_{PDSU}	Fig. 2	200	–	–	ns
Peripheral Data Hold Time	Port 1, 2	t_{PDH}	Fig. 2	200	–	–	ns
Delay Time, Enable Negative Transition to Peripheral Data Valid	Port 1, 2*	t_{PWD}	Fig. 3	–	–	400	ns

* Except P_{21}

TIMER, SCI TIMING (V_{CC} = 5.0V ±5%, V_{SS} = 0V, Ta = 0 ~ +70°C, unless otherwise noted.)

Item	Symbol	Test Condition	min	typ	max	Unit
Timer Input Pulse Width	t_{PWT}		$2t_{cyc}$+200	–	–	ns
Delay Time, Enable Positive Transition to Timer Out	t_{TOD}	Fig. 4	–	–	600	ns
SCI Input Clock Cycle	t_{Scyc}		1	–	–	t_{cyc}
SCI Input Clock Pulse Width	t_{PWSCK}		0.4	–	0.6	t_{Scyc}

MODE PROGRAMMING (V_{CC} = 5.0V ±5%, V_{SS} = 0V, Ta = 0 ~ +70°C, unless otherwise noted.)

Item		Symbol	Test Condition	min	typ	max	Unit
Mode Programming Input "Low" Voltage		V_{MPL}	Fig. 5	–	–	1.7	V
Mode Programming Input "High" Voltage		V_{MPH}		4.0	–	–	V
$\overline{RES}$ "Low" Pulse Width		PW_{RSTL}		3.0	–	-	t_{cyc}
Mode Programming Set-up Time		t_{MPS}		2.0	–	–	t_{cyc}
Mode Programming Hold Time	$\overline{RES}$ Rise Time ≧ 1μs	t_{MPH}		0	–	–	ns
	$\overline{RES}$ Rise Time < 1μs			100	–	–	

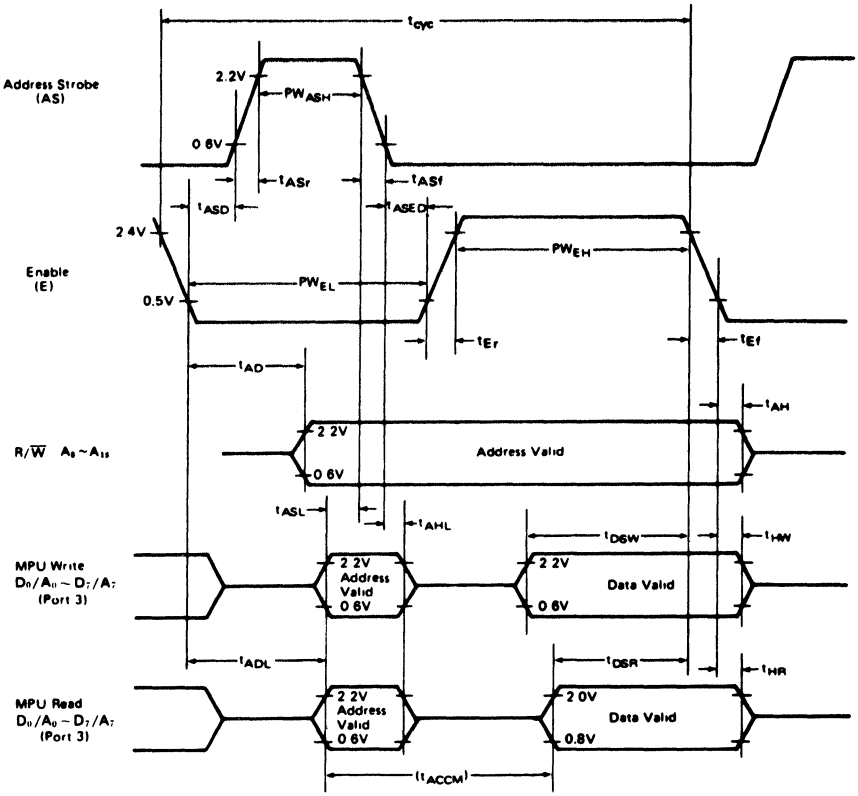


Figure 1 Expanded Multiplexed Bus Timing

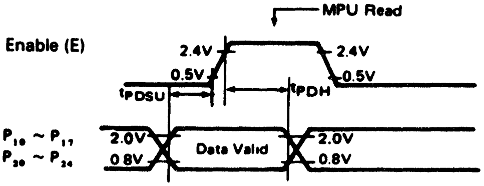


Figure 2 Data Set-up and Hold Times (MPU Read)

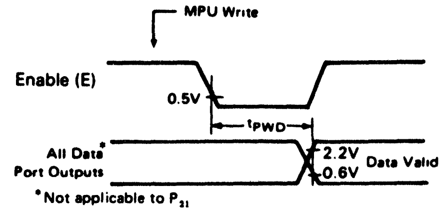


Figure 3 Port Data Delay Timing (MPU Write)

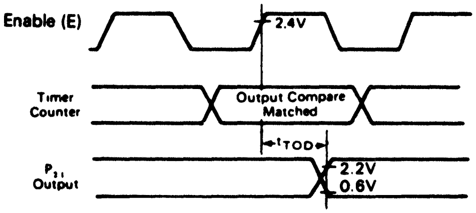


Figure 4 Timer Output Timing

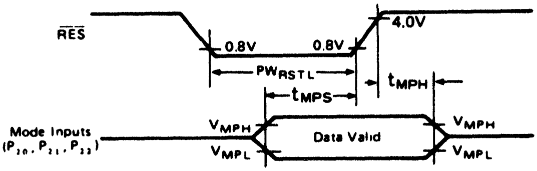
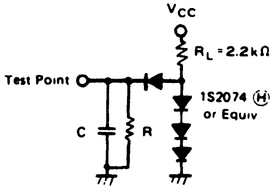
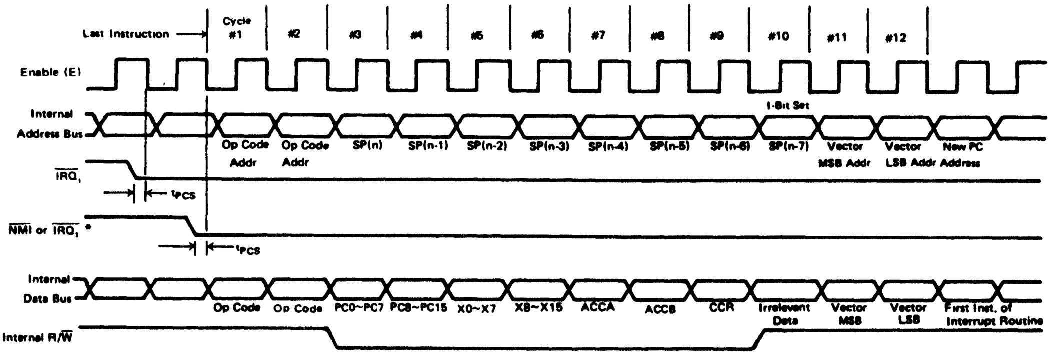


Figure 5 Mode Programming Timing



C = 90 pF for D_0/A_0 ~ D_7/A_7, A_8 ~ A_{15}, E, AS, R/$\overline{W}$
= 30 pF for P_{10} ~ P_{17}, P_{20} ~ P_{24}
R = 12 kΩ for D_0/A_0 ~ D_7/A_7, A_8 ~ A_{15}, E, AS, R/$\overline{W}$
= 24 kΩ for P_{10} ~ P_{17}, P_{20} ~ P_{24}
TTL Load

Figure 6 Bus Timing Test Load



* IRQ2 , Internal interrupt

Figure 7 Interrupt Sequence

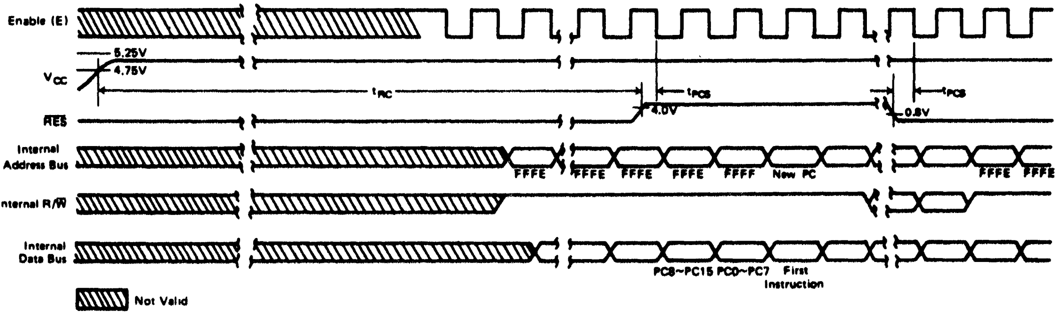


Figure 8 Reset Timing

■ **SIGNAL DESCRIPTIONS**

• Vcc and Vss

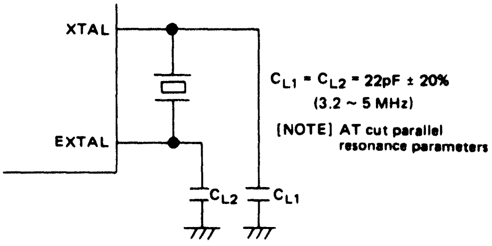
These two pins are used to supply power and ground to the chip. The voltage supplied will be +5 volts ±5%.

• XTAL and EXTAL

These connections are for a parallel resonant fundamental crystal, AT cut. Devide-by-4 circuitry is included with the internal clock, so a 4 MHz crystal may be used to run the system at 1 MHz. The devide-by-4 circuitry allows for use of the inexpensive 3.58 MHz Color TV crystal for non-time critical applications. Two 22pF capacitors are needed from the two crystal pins to ground to insure reliable operation. An example of the crystal interface is shown in Fig. 9. EXTAL may be driven by an external TTL compatible source with a 45% to 55% duty cycle. It will devided by 4 any frequency less than or equal to 5 MHz. XTAL must be grounded if an external clock is used.

Nominal Crystal Parameter

Crystal / Item	4 MHz	5 MHz
C_O	7pF max.	4.7pF max.
R_S	60Ω max.	30Ω typ.



$C_{L1} = C_{L2} = 22pF \pm 20\%$
(3.2 ~ 5 MHz)
[NOTE] AT cut parallel resonance parameters

Figure 9 Crystal Interface

- **V_{CC} Standby**

This pin will supply +5 volts ±5% to the standby RAM on the chip. The first 64 bytes of RAM will be maintained in the power down mode with 8 mA current max. The circuit of figure 13 can be utilized to assure that V_{CC} Standby does not go below V_{SBB} during power down.

To retain information in the RAM during power down the following procedure is necessary:

1) Write "0" into the RAM enable bit, RAME. RAME is bit 6 of the RAM Control Register at location $0014. This disables the standby RAM, thereby protecting it at power down.
2) Keep V_{CC} Standby greater than V_{SBB}.

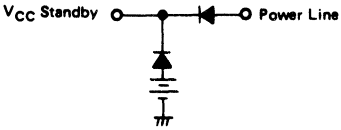


Figure 10 Battery Backup for V_{CC} Standby

- **Reset ($\overline{\text{RES}}$)**

This input is used to reset and start the MPU from a power down condition, resulting from a power failure or an initial startup of the processor. On power up, the reset must be held "Low" for at least 100 ms. When reset during operation, $\overline{\text{RES}}$ must be held "Low" at least 3 clock cycles.

When a "High" level is detected, the CPU does the following;

1) All the higher order address lines will be forced "High".
2) I/O Port 2 bits, 2, 1, and 0 are latched into programmed control bits PC2, PC1 and PC0.
3) The last two ($FFFE, $FFFF) locations in memory will be used to load the program addressed by the program counter.
4) The interrupt mask bit is set. Clear before the CPU can recognize maskable interrupts.

- **Enable (E)**

This supplies the external clock for the rest of the system when the internal oscillator is used. It is a single phase, TTL compatible clock, and will be the divide-by-4 result of the crystal oscillator frequency. It will drive one TTL load and 90 pF capacitance.

- **Non-Maskable Interrupt ($\overline{\text{NMI}}$)**

When the falling edge of the input signal is detected at this pin, the CPU begins non-maskable interrupt sequence internally. As with interrupt Request signal, the processor will complete the current instruction that is being executed before it recognizes the $\overline{\text{NMI}}$ signal. The interrupt mask bit in the Condition Code Register has no effect on $\overline{\text{NMI}}$.

In response to an $\overline{\text{NMI}}$ interrupt, the Index Register, Program Counter, Accumulators, and Condition Code Register are stored on the stack. At the end of the sequence, a 16-bit address will be loaded that points to a vectoring address located in memory locations $FFFC and $FFFD. An address loaded at these locations causes the CPU to branch to a non-maskable interrupt service routine in memory.

A 3.3 kΩ external resistor to V_{CC} should be used for wire-OR and optimum control of interrupts.

Inputs $\overline{\text{IRQ}_1}$ and $\overline{\text{NMI}}$ are hardware interrupt lines that are sampled during E and will start the interrupt routine on the $\bar{\text{E}}$ following the completion of an instruction.

- **Interrupt Request ($\overline{\text{IRQ}_1}$)**

This level sensitive input requests that an interrupt sequence be generated within the machine. The processor will complete the current instruction before it recognizes the request. At that time, if the interrupt mask bit in the Condition Code Register is not set, the machine will begin an interrupt sequence. The Index Register, Program Counter, Accumulators, and Condition Code Register are stored on the stack. Next the CPU will respond to the interrupt request by setting the interrupt mask bit "High" so that no further maskable interrupts may occur. At the end of the cycle, a 16-bit address will be loaded that points to a vectoring address which is located in memory locations $FFF8 and $FFF9. An address loaded at these locations causes the CPU to branch to an interrupt routine in memory.

The $\overline{\text{IRQ}_1}$ requires a 3.3 kΩ external resistor to V_{CC} which should be used for wire-OR and optimum control of interrupts. Internal Interrupts will use an internal interrupt line ($\overline{\text{IRQ}_2}$). This interrupt will operate the same as $\overline{\text{IRQ}_1}$ except that it will use the vector address of $FFF0 through $FFF7. $\overline{\text{IRQ}_1}$ will have priority to $\overline{\text{IRQ}_2}$ if both occur at the same time. The Interrupt Mask Bit in the condition code register masks both interrupts (See Table 1).

Table 1 Interrupt Vector Location

	Vector		Interrupt
	MSB	LSB	
Highest Priority	FFFE	FFFF	$\overline{\text{RES}}$
	FFFC	FFFD	$\overline{\text{NMI}}$
	FFFA	FFFB	Software Interrupt (SWI)
	FFF8	FFF9	$\overline{\text{IRQ}_1}$
	FFF6	FFF7	ICF (Input Capture)
	FFF4	FFF5	OCF (Output Compare)
	FFF2	FFF3	TOF (Timer Overflow)
Lowest Priority	FFF0	FFF1	SCI (RDRF + ORFE + TDRE)

- **Read/Write (R/$\overline{\text{W}}$)**

This TTL compatible output signals the peripherals and memory devices whether the CPU is in a Read ("High") or a Write ("Low") state. The normal standby state of this signal is Read ("High"). This output can drive one TTL load and 90pF capacitance.

- **Address Strobe (AS)**

In the expanded multiplexed mode of operation, address strobe is output on this pin. This signal is used to latch the 8 LSB's of address which are multiplexed with data on D_0/A_0 to D_7/A_7. An 8-bit latch is utilized in conjunction with Address Strobe, as shown in figure 11. So D_0/A_0 to D_7/A_7 can become data bus during the E pulse. The timing for this signal is shown in Figure 1 of Bus Timing. This signal is also used to disable the address from the multiplexed bus allowing a deselect time, t_{ASD} before the data is enabled to the bus.

- **PORTS**

There are two I/O ports on the HD6803 MPU; one 8-bit port and one 5-bit port. Each port has an associated write only Data Direction Register which allows each I/O line to be programmed to act as an input or an output*. A "1" in the corresponding Data Direction Register bit will cause that I/O line to be an output. A "0" in the corresponding Data Direction Register bit will cause that I/O line to be an input. There are two ports: Port 1, Port 2. Their addresses and the addresses of their Data Direction registers are given in Table 2.

* The only exception is bit 1 of Port 2, which can either be data input or Timer output

Table 2 Port and Data Direction Register Addresses

Ports	Port Address	Data Direction Register Address
I/O Port 1	$0002	$0000
I/O Port 2	$0003	$0001

- **I/O Port 1**

This is an 8-bit port whose individual bits may be defined as inputs or outputs by the corresponding bit in its data direction register. The 8 output buffers have three-state capability, allowing them to enter a high impedance state when the peripheral data lines are used as inputs. In order to be read properly, the voltage on the input lines must be greater than 2.0 V for a logic "1" and less than 0.8 V for a logic "0". As outputs, these lines are TTL compatible and may also be used as a source of up to 1 mA at 1.5 V to directly drive a Darlington base. After reset, the I/O lines are configured as inputs.

- **I/O Port 2**

This port has five lines that may be defined as inputs or outputs by its data direction register. The 5 output buffers have three-state capability, allowing them to enter a high impedance state when used as an input. In order to be read properly, the voltage on the input lines must be greater than 2.0 V for a logic "1" and less than 0.8 V for a logic "0". As outputs, this port has no internal pullup resistors but will drive TTL inputs directly. For driving CMOS inputs, external pullup resistors are required. After reset, the I/O lines are configured as inputs. Three pins on Port 2 (pin 8, 9 and 10 of the chip) are requested to set following values (Table 3) during reset. The values of above three pins during reset are latched into the three MSBs (Bit 5, 6 and 7) of Port 2 which are read only.

Port 2 can be configured as I/O and provides access to the Serial Communications Interface and the Timer. Bit 1 is the only pin restricted to data input or Timer output.

Table 3 The Values of three pins

Pin Number	Value
8	L
9	H
10	L

[NOTES] L; Logical "0"
H; Logical "1"

■ BUS

- **Data/Address Lines (D_0/A_0 ~ D_7/A_7)**

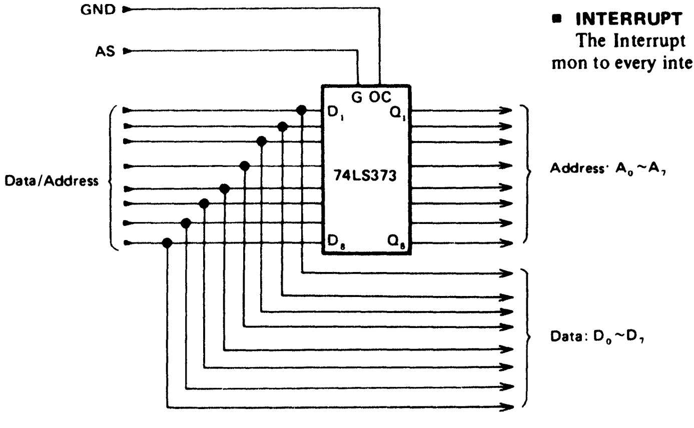
Since the data bus is multiplexed with the lower order address bus in Data/Address, latches are required to latch those address bits. The 74LS373 Transparent Octal D-type latch can be used with the HD6803 to latch the least significant address byte. Figure 11 shows how to connect the latch to the HD6803. The output control to the 74LS373 may be connected to ground.

- **Address Lines (A_8 ~ A_{15})**

Each line is TTL compatible and can drive one TTL load and 90 pF. After reset, these pins become output for upper order address lines (A_8 to A_{15}).

■ INTERRUPT FLOWCHART

The Interrupt flowchart is depicted in Figure 16 and is common to every interrupt excluding reset.



Function Table

Output Control	Enable G	Enable D	Output Q
L	H	H	H
L	H	L	L
L	L	X	Q_0
H	X	X	Z

Figure 11 Latch Connection

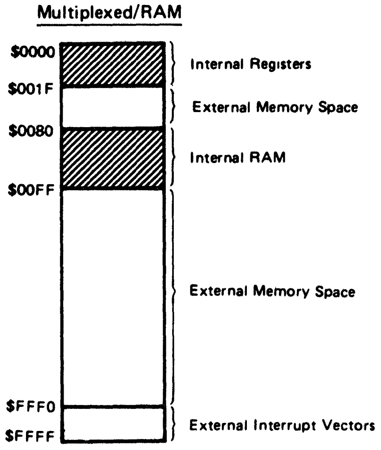
■ MEMORY MAP

The MPU can provide up to 65k byte address space. A memory map is shown in Figure 12. The first 32 locations are reserved for the MPU's internal register area, as shown in Table 4 with exceptions as indicated.

Table 4 Internal Register Area

Register	Address
Port 1 Data Direction Register**	00
Port 2 Data Direction Register**	01
Port 1 Data Register	02
Port 2 Data Register	03
Not Used	04*
Not Used	05*
Not Used	06*
Not Used	07*
Timer Control and Status Register	08
Counter (High Byte)	09
Counter (Low Byte)	0A
Output Compare Register (High Byte)	0B
Output Compare Register (Low Byte)	0C
Input Capture Register (High Byte)	0D
Input Capture Register (Low Byte)	0E
Not Used	0F*
Rate and Mode Control Register	10
Transmit/Receive Control and Status Register	11
Receive Data Register	12
Transmit Data Register	13
RAM Control Register	14
Reserved	15-1F

* External Address
** 1; Output, 0; Input



(NOTE)
Excludes the following addresses which may be used externally. $04, $05, $06, $07, and $0F.

Figure 12 HD6803 Memory Map

■ PROGRAMMABLE TIMER

The HD6803 contains an on-chip 16-bit programmable timer which may be used to measure an input waveform while independently generating an output waveform. Pulse widths for both input and output signals may vary from a few microseconds to many seconds. The timer hardware consists of

- an 8-bit control and status register,
- a 16-bit free running counter,
- a 16-bit output compare register,
- a 16-bit input capture register

A block diagram of the timer registers is shown in Figure 13.

● Free Running Counter ($0009:$000A)

The key element in the programmable timer is a 16-bit free running counter which is driven to increasing values by E (Enable). The counter value may be read by the CPU software at any time. The counter is cleared to zero by reset and may be considered a read-only register with one exception. Any CPU write to the counter's address ($09) will always result in preset value of $FFF8 being loaded into the counter regardless of the value involved in the write. This preset figure is intended for testing operation of the part, but may be of value in some applications.

● Output Compare Register ($000B:$000C)

The Output Compare Register is a 16-bit read/write register which is used to control an output waveform. The contents of this register are constantly compared with the current value of the free running counter. When a match is found, a flag is set (OCF) in the Timer Control and Status Register (TCSR) and the current value of the Output Level bit (OLVL) in the TCSR is clocked to the Output Level Register. Providing the Data Direction Register for Port 2, Bit 1 contains a "1" (Output), the output level register value will appear on the pin for Port 2 Bit 1. The values in the Output Compare Register and Output Level bit may then be changed to control the output level on the next compare value. The Output Compare Register is set to $FFFF during reset. The Compare function is inhibited for one cycle following a write to the high byte of the Output Compare Register to insure a valid 16-bit value is in the register before a compare is made.

● Input Capture Register ($000D:$000E)

The Input Capture Register is a 16-bit read-only register used to store the current value of the free running counter when the proper transition of an external input signal occurs. The input transition change required to trigger the counter transfer is controlled by the input Edge bit (IEDG) in the TCSR. The Data Direction Register bit for Port 2 Bit 0, should* be clear (zero) in order to gate in the external input signal to the edge detect unit in the timer.

The input pulse width must be at least two E-cycles to ensure an input capture under all conditions.

* With Port 2 Bit 0 configured as an output and set to "1", the external input will still be seen by the edge detect unit.

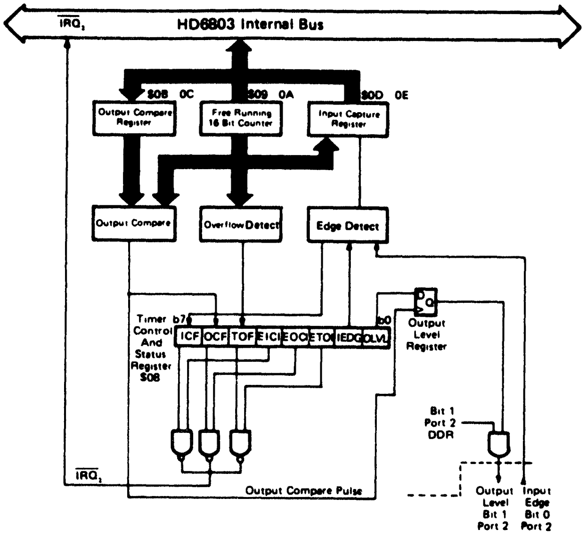


Figure 13 Block Diagram of Programmable Timer

Timer Control and Status Register

7	6	5	4	3	2	1	0	
ICF	OCF	TOF	EICI	EOCI	ETOI	IEDG	OLVL	$0008

- **Timer Control and Status Register (TCSR) ($0008)**

The Timer Control and Status Register consists of an 8-bit register of which all 8 bits are readable but only the low order 5 bits may be written. The upper three bits contain read-only timer status information and indicate the followings:

- a proper transition has taken place on the input pin with a subsequent transfer of the current counter value to the input capture register.
- a match has been found between the value in the free running counter and the output compare register, and when $0000 is in the free running counter.

Each of the flags may be enabled onto the HD6803 internal bus ($\overline{IRQ_2}$) with an individual Enable bit in the TCSR. If the I-bit in the HD6803 Condition Code register has been cleared, a prior vectored interrupt will occur corresponding to the flag bit(s) set. A description for each bit follows:

Bit 0 **OLVL** Output Level – This value is clocked to the output level register on a successful output compare. If the DDR for Port 2 bit 1 is set, the value will appear on the output pin.

Bit 1 **IEDG** Input Edge – This bit controls which transition of an input will trigger a transfer of the counter to the input capture register. The DDR for Port 2 Bit 0 must be clear for this function to operate. IEDG = 0 Transfer takes place on a negative edge ("High"-to-"Low" transition).
IEDG = 1 Transfer takes place on a positive edge ("Low"-to-"High" transition).

Bit 2 **ETOI** Enable Timer Overflow Interrupt – When set, this bit enables $\overline{IRQ_2}$ to occur on the internal bus for a TOF interrupt; when clear the interrupt is inhibited.

Bit 3 **EOCI** Enable Output Compare Interrupt – When set, this bit enables $\overline{IRQ_2}$ to appear on the internal bus for an output compare interrupt; when clear the interrupt is inhibited.

Bit 4 **EICI** Enable input Capture Interrupt – When set, this bit enables $\overline{IRQ_2}$ to occur on the internal bus for an input capture interrupt; when clear the interrupt is inhibited.

Bit 5 **TOF** Timer Overflow Flag – This read-only bit is set when the counter contains $FFFF. It is cleared by a read of the TCSR (with TOF set) followed by an CPU read of the Counter ($09).

Bit 6 **OCF** Output Compare Flag – This read-only bit is set when a match is found between the output compare register and the free running counter. It is cleared by a read of the TCSR (with OCF set) followed by an CPU write to the output compare register ($0B or $0C).

Bit 7 **ICF** Input Capture Flag – This read-only status bit is set by a proper transition on the input, it is cleared by a read of the TCSR (with ICF set) followed by an CPU read of the Input Capture Register ($0D).

■ SERIAL COMMUNICATIONS INTERFACE

The HD6803 contains a full-duplex asynchronous serial communications interface (SCI) on chip. The controller comprises a transmitter and a receiver which operate independently or each other but in the same data format and at the same data rate. Both transmitter and receiver communicate with the CPU via the data bus and with the outside world via pins 2, 3, and 4 of Port 2. The hardware, software, and registers are explained in the following paragraphs.

● Wake-Up Feature

In a typical multi-processor application, the software protocol will usually contain a destination address in the initial byte(s) of the message. In order to permit non-selected MPU's to ignore the remainder of the message, a wake-up feature is included whereby all further interrupt processing may be optionally inhibited until the beginning of the next message. When the next message appears, the hardware re-enables (or "wakes-up") for the next message. The "wake-up" is automatically triggered by a string of ten consecutive 1's which indicates an idle transmit line. The software protocol must provide for the short idle period between any two consecutive messages.

● Programmable Options

The following features of the HD6803 serial I/O section are programmable:

- format – standard mark/space (NRZ)
- Clock – external or internal
- baud rate – one of 4 per given CPU ϕ_2 clock frequency or external clock ×8 input
- wake-up feature – enabled or disabled
- Interrupt requests – enabled or masked individually for transmitter and receiver data registers
- clock output – internal clock enabled or disabled to Port 2 (Bit 2)
- Port 2 (bits 3 and 4) – dedicated or not dedicated to serial I/O individually for transmitter and receiver.

● Serial Communications Hardware

The serial communications hardware is controlled by 4 registers as shown in Figure 14. The registers include:

- an 8-bit control and status register
- a 4-bit rate and mode control register (write only)
- an 8-bit read only receive data register and
- an 8-bit write only transmit data register.

In addition to the four registers, the serial I/O section utilizes bit 3 (serial input) and bit 4 (serial output) of Port 2. Bit 2 of Port 2 is utilized if the internal-clock-out or external-clock-in options are selected.

Transmit/Receive Control and Status (TRCS) Register

The TRCS register consists of an 8-bit register of which all 8 bits may be read while only bits 0~4 may be written. The register is initialized to $20 by reset. The bits in the TRCS register are defined as follows:

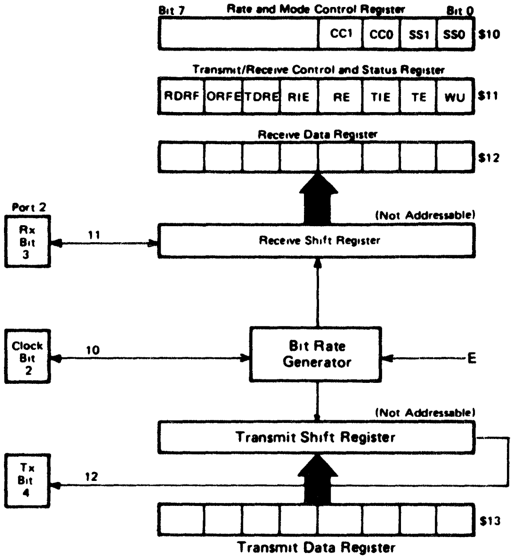


Figure 14 Serial I/O Registers

Bit 0 **WU** "Wake-up" on Next Message – set by HD6803 software and cleared by hardware on receipt of ten consecutive 1's or reset of RE flag. It should be noted that RE flag should be set in advance of CPU set of WU flag.

Bit 1 **TE** Transmit Enable – set by HD6803 to produce preamble of nine consecutive 1's and to enable gating of transmitter output to Port 2, bit 4 regardless of the DDR value corresponding to this bit; when clear, serial I/O has no effect on Port 2 bit 4.

TE set should be after at least one bit time of data transmit rate from the set-up of transmit data rate and mode.

Bit 2 **TIE** Transmit Interrupt Enable – when set, will permit an $\overline{IRQ_2}$ interrupt to occur when bit 5 (TDRE) is set; when clear, the TDRE value is masked from the bus.

Bit 3 **RE** Receiver Enable – when set, gates Port 2 bit 3 to input of receiver regardless of DDR value for this bit; when clear, serial I/O has no effect on Port 2 bit 3.

Bit 4 **RIE** Receiver Interrupt Enable – when set, will permit an $\overline{IRQ_2}$ interrupt to occur when bit 7 (RDRF) or bit 6 (ORFE) is set; when clear, the interrupt is masked.

Transmit/Receive Control and Status Register

7	6	5	4	3	2	1	0	
RDRF	ORFE	TDRE	RIE	RE	TIE	TE	WU	ADDR $0011

Bit 5 **TDRE** Transmit Data Register Empty – set by hardware when a transfer is made from the transmit data register to the output shift register. The TDRE bit is cleared by reading the status register, then writing a new byte into the transmit data register, TDRE is initialized to 1 by reset.

Bit 6 **ORFE** Over-Run-Framing Error – set by hardware when an overrun or framing error occurs (receive only).

An overrun is defined as a new byte received with last byte still in Data Register/Buffer. A framing error has occured when the byte boundaries in bit stream are not synchronized to bit counter. If WU-flag is set, the ORFE bit will not be set. The ORFE bit is cleard by reading the status register, then reading the Receive Data Register, or by reset.

Bit 7 **RDRF** Receiver Data Register Full-set by hardware when a transfer from the input shift register to the receiver data register is made. If WU-flag is set, the RDRF bit will not be set. The RDRF bit is cleared by reading the status register, then reading the Receive Data Register, or by reset.

Rate and Mode Control Register

7	6	5	4	3	2	1	0
X	X	X	X	CC1	CC0	SS1	SS0

ADDR : $0010

Rate and Mode Control Register (RMCR)

The Rate and Mode Control register controls the following serial I/O variables:

- Baud rate
- format
- clocking source,
- Port 2 bit 2 configuration

The register consists of 4 bits all of which are write-only and cleared by reset. The 4 bits in the register may be considered as a pair of 2-bit fields. The two low order bits control the bit rate for internal clocking and the remaining twq bits control the format and clock select logic. The register definition is as follows:

Bit 0 **SSO**
Bit 1 **SS1** } Speed Select – These bits select the Baud rate for the internal clock. The four rates which may be selected are a function of the CPU ϕ_2 clock frequency. Table 5 lists the available Baud rates.

Bit 2 **CCO**
Bit 3 **CC1** } Clock Control and Format Select – this 2-bit field controls the format and clock select logic. Table 6 defines the bit field.

Table 5 SCI Bit Times and Rates

	XTAL	2.4576 MHz	4.0 MHz	4.9152 MHz*
SS1 : SS0	E	614.4 kHz	1.0 MHz	1.2288 MHz
0 0	E ÷ 16	26 μs/38,400 Baud	16 μs/62,500 Baud	13.0 μs/76,800 Baud
0 1	E ÷ 128	208 μs/4,800 Baud	128 μs/7812.5 Baud	104.2 μs/9,600 Baud
1 0	E ÷ 1024	1.67 ms/600 Baud	1.024 ms/976.6 Baud	833.3 μs/1,200 Baud
1 1	E ÷ 4096	6.67 ms/150 Baud	4.096 ms/244.1 Baud	3.33 ms/300 Baud

* HD6803-1 Only

Table 6 SCI Format and Clock Source Control

CC1 : CC0	Format	Clock Source	Port 2 Bit 2	Port 2 Bit 3	Port 2 Bit 4
0 0	–	–	–	–	–
0 1	NRZ	Internal	Not Used	**	**
1 0	NRZ	Internal	Output*	**	**
1 1	NRZ	External	Input	**	**

* Clock output is available regardless of values for bits RE and TE
** Bit 3 is used for serial input if RE = "1" in TRCS, bit 4 is used for serial output if TE = "1" in TRCS.

Internally Generated Clock

If the user wishes for the serial I/O to furnish a clock, the following requirements are applicable:

- the values of RE and TE are immaterial.
- CC1, CC0 must be set to 10
- the maximum clock rate will be E – 16.
- the clock will be at 1x the bit rate and will have a rising edge at mid-bit.

Externally Generated Clock

If the user wishes to provide an external clock for the serial I/O, the following requirements are applicable:

- the CC1, CC0, field in the Rate and Mode Control Register must be set to 11,
- the external clock must be set to 8 times (×8) the desired baud rate and
- the maximum external clock frequency is 1.0 MHz.

• Serial Operations

The serial I/O hardware should be initialized by the HD6803 software prior to operation. This sequence will normally consist of;

- writing the desired operation control bits to the Rate and Mode Control Register and
- writing the desired operational control bits in the Transmit/Receive Control and Status Register.

The Transmitter Enable (TE) and Receiver Enable (RE) bits may be left set for dedicated operations.

Transmit Operations

The transmit operation is enabled by the TE bit in the Transmit/Receive Control and Status Register. This bit when set, gates the output of the serial transmit shift register to Port 2 Bit 4 and takes unconditional control over the Data Direction Register value for Port 2, Bit 4.

Following a $\overline{RES}$ the user should configure both the Rate and Mode Control Register and the Transmit/Receive Control and Status Register for desired operation. Setting the TE bit during this procedure initiates the serial output by first transmitting a nine-bit preamble of 1's. Following the preamble, internal synchronization is established and the transmitter section is ready for operation.

At this point one of two situation exist:

1) if the Transmit Data Register is empty (TDRE = 1), a continuous string of ones will be sent indicating an idle line, or,
2) if data has been loaded into the Transmit Data Register (TDRE = 0), the word is transferred to the output shift register and transmission of the data word will begin.

During the data transmit, the 0 start bit is first transmitted. Then the 8 data bits (beginning with bit 0) followed by the stop bit, are transmitted. When the Transmitter Data Register has been emptied, the hardware sets the TDRE flag bit.

If the HD6803 fails to respond to the flag within the proper time, (TDRE is still set when the next normal transfer from the parallel data register to the serial output register should occur) then a 1 will be sent (instead of a 0) at "Start" bit time, followed by more 1's until more data is supplied to the data register. No 0's will be sent while TDRE remains a 1.

Receive Operation

The receive operation is enabled by the RE bit which gates in the serial input through Port 2, Bit 3. The receiver section operation is conditioned by the contents of the Transmit/Receive Control and Status Register and the Rate and Mode Control Register.

The receiver bit interval is divided into 8 sub-intervals for internal synchronization. In the NRZ Mode, the received bit stream is synchronized by the first 0 (space) encountered.

The approximate center of each bit time is strobed during the next 10 bits. If the tenth bit is not a 1 (stop bit) a framing error is assumed, and bit ORFE is set. If the tenth bit as a 1, the data is transferred to the Receive Data Register, and interrupt flag RDRF is set. If RDRF is still set at the next tenth bit time, ORFE will be set, indicating an overrun has occurred. When the HD6803 responds to either flag (RDRF or ORFE) by reading the status register followed by reading the Data Register, RDRF (or ORFE) will be cleared.

■ RAM CONTROL REGISTER

This register, which is addressed at S0014, gives status information about the standby RAM. A 0 in the RAM enable bit (RAME) will disable the standby RAM, thereby protecting it at power down if V_{CC} Standby is held greater than V_{SBB} volts, as explained previously in the signal description for V_{CC} Standby.

RAM Control Register

$0014	STBY PWR	RAME	X	X	X	X	X	X

Bit 0 Not used.
Bit 1 Not used.
Bit 2 Not used.
Bit 3 Not used.
Bit 4 Not used.
Bit 5 Not used.

Bit 6 **RAME** The RAM Enable control bit allows the user the ability to disable the standby RAM. This bit is set to a logic "1" by $\overline{RES}$ which enables the standby RAM and can be written to one or zero under program control. When the RAM is disabled, data is read from external memory.

Bit 7 **STBY PWR** The Standby Power bit is cleared when the standby voltage is removed. This bit is a read/write status flag that the user can read which indicates that the standby RAM voltage has been applied, and the data in the standby RAM is valid.

■ GENERAL DESCRIPTION OF INSTRUCTION SET

The HD6803 is upward object code compatible with the HD6800 as it implements the full HMCS6800 instruction set. The execution times of key instructions have been reduced to increase throughout. In addition, new instructions have been added; these include 16-bit operations and a hardware multiply.

Included in the instruction set section are the following:

- CPU Programming Model–Figure 15.
- Addressing modes
- Accumulator and memory instructions – Table 7
- New instructions
- Index register and stack manipulations instructions – Table 8
- Jump and branch instructions – Table 9
- Condition code register manipulation instructions – Table 10
- Instructions Execution times in machine cycles – Table 11
- Summary of cycle by cycle operation – Table 12
- Summary of undefined instructions – Table 13

• CPU Programming Model

The programming model for the HD6803 is shown in Figure 15. The double (D) accumulator is physically the same as the Accumulator A concatenated with the Accumulator B so that any operation using accumulator D will destroy information in A and B.

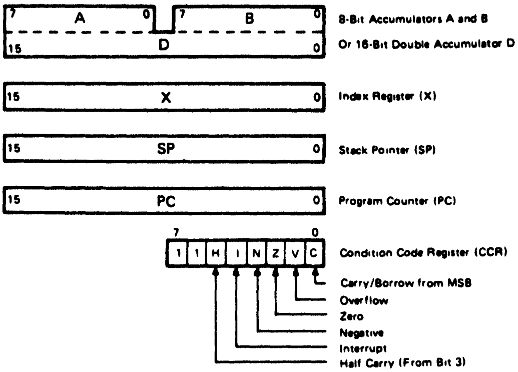


Figure 15 CPU Programming Model

- **CPU Addressing Modes**

The HD6803 8-bit micro processing unit has seven address modes that can be used by a programmer, with the addressing mode a function of both the type of instruction and the coding within the instruction. A summary of the addressing modes for a particular instruction can be found in Table 11 along with the associated instruction execution time that is given in machine cycles. With a clock frequency of 4 MHz, these times would be microseconds.

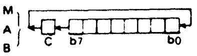
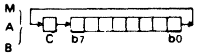
Accumulator (ACCX) Addressing

In accumulator only addressing, either accumulator A or accumulator B is specified. These are one-byte instructions.

Immediate Addressing

In immediate addressing, the operand is contained in the second byte of the instruction except LDS and LDX which have the operand in the second and third bytes of the instruction. The CPU addresses this location when it fetches the immediate instruction for execution. These are two or three-byte instructions.

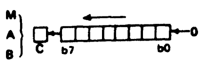
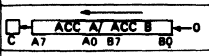
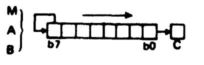
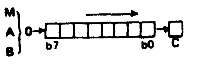
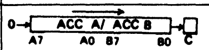
Table 7 Accumulator & Memory Instructions

Operations	Mnemonic	Addressing Modes															Boolean/ Arithmetic Operation	Condition Code Register					
		IMMED			DIRECT			INDEX			EXTEND			IMPLIED				5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#		H	I	N	Z	V	C
Add	ADDA	8B	2	2	9B	3	2	AB	4	2	BB	4	3				A + M → A	↕	●	↕	↕	↕	↕
	ADDB	CB	2	2	DB	3	2	EB	4	2	FB	4	3				B + M → B	↕	●	↕	↕	↕	↕
Add Double	ADDD	C3	4	3	D3	5	2	E3	6	2	F3	6	3				A B + M M + 1 → A B	●	●	↕	↕	↕	↕
Add Accumulators	ABA													1B	2	1	A + B → A	↕	●	↕	↕	↕	↕
Add With Carry	ADCA	89	2	2	99	3	2	A9	4	2	B9	4	3				A + M + C → A	↕	●	↕	↕	↕	↕
	ADCB	C9	2	2	D9	3	2	E9	4	2	F9	4	3				B + M + C → B	↕	●	↕	↕	↕	↕
AND	ANDA	84	2	2	94	3	2	A4	4	2	B4	4	3				A·M → A	●	●	↕	↕	R	●
	ANDB	C4	2	2	D4	3	2	E4	4	2	F4	4	3				B·M → B	●	●	↕	↕	R	●
Bit Test	BIT A	85	2	2	95	3	2	A5	4	2	B5	4	3				A·M	●	●	↕	↕	R	●
	BIT B	C5	2	2	D5	3	2	E5	4	2	F5	4	3				B·M	●	●	↕	↕	R	●
Clear	CLR							6F	6	2	7F	6	3				00 → M	●	●	R	S	R	R
	CLRA													4F	2	1	00 → A	●	●	R	S	R	R
	CLRB													5F	2	1	00 → B	●	●	R	S	R	R
Compare	CMPA	81	2	2	91	3	2	A1	4	2	B1	4	3				A − M	●	●	↕	↕	↕	↕
	CMPB	C1	2	2	D1	3	2	E1	4	2	F1	4	3				B − M	●	●	↕	↕	↕	↕
Compare Accumulators	CBA													11	2	1	A − B	●	●	↕	↕	↕	↕
Complement, 1's	COM							63	6	2	73	6	3				$\overline{M}$ → M	●	●	↕	↕	R	S
	COMA													43	2	1	$\overline{A}$ → A	●	●	↕	↕	R	S
	COMB													53	2	1	$\overline{B}$ → B	●	●	↕	↕	R	S
Complement, 2's	NEG							60	6	2	70	6	3				00 − M → M	●	●	↕	↕	①	②
(Negate)	NEGA													40	2	1	00 − A → A	●	●	↕	↕	①	②
	NEGB													50	2	1	00 − B → B	●	●	↕	↕	①	②
Decimal Adjust, A	DAA													19	2	1	Converts binary add of BCD characters into BCD format	●	●	↕	↕	↕	③
Decrement	DEC							6A	6	2	7A	6	3				M − 1 → M	●	●	↕	↕	④	●
	DECA													4A	2	1	A − 1 → A	●	●	↕	↕	④	●
	DECB													5A	2	1	B − 1 → B	●	●	↕	↕	④	●
Exclusive OR	EORA	88	2	2	98	3	2	A8	4	2	B8	4	3				A ⊕ M → A	●	●	↕	↕	R	●
	EORB	C8	2	2	D8	3	2	E8	4	2	F8	4	3				B ⊕ M → B	●	●	↕	↕	R	●
Increment	INC							6C	6	2	7C	6	3				M + 1 → M	●	●	↕	↕	⑤	●
	INCA													4C	2	1	A + 1 → A	●	●	↕	↕	⑤	●
	INCB													5C	2	1	B + 1 → B	●	●	↕	↕	⑤	●
Load Accumulator	LDAA	86	2	2	96	3	2	A6	4	2	B6	4	3				M → A	●	●	↕	↕	R	●
	LDAB	C6	2	2	D6	3	2	E6	4	2	F6	4	3				M → B	●	●	↕	↕	R	●
Load Double Accumulator	LDD	CC	3	3	DC	4	2	EC	5	2	FC	5	3				M + 1 → B, M → A	●	●	↕	↕	R	●
Multiply Unsigned	MUL													3D	10	1	A × B → A B	●	●	●	●	●	⑪
OR, Inclusive	ORAA	8A	2	2	9A	3	2	AA	4	2	BA	4	3				A + M → A	●	●	↕	↕	R	●
	ORAB	CA	2	2	DA	3	2	EA	4	2	FA	4	3				B + M → B	●	●	↕	↕	R	●
Push Data	PSHA													36	3	1	A → Msp, SP − 1 → SP	●	●	●	●	●	●
	PSHB													37	3	1	B → Msp, SP − 1 → SP	●	●	●	●	●	●
Pull Data	PULA													32	4	1	SP + 1 → SP, Msp → A	●	●	●	●	●	●
	PULB													33	4	1	SP + 1 → SP, Msp → B	●	●	●	●	●	●
Rotate Left	ROL							69	6	2	79	6	3					●	●	↕	↕	⑥	↕
	ROLA													49	2	1		●	●	↕	↕	⑥	↕
	ROLB													59	2	1		●	●	↕	↕	⑥	↕
Rotate Right	ROR							66	6	2	76	6	3					●	●	↕	↕	⑥	↕
	RORA													46	2	1		●	●	↕	↕	⑥	↕
	RORB													56	2	1		●	●	↕	↕	⑥	↕

The Condition Code Register notes are listed after Table 10

(Continued)

Table 7 Accumulator & Memory Instructions (Continued)

Operations	Mnemonic	Addressing Modes															Boolean/ Arithmetic Operation	Condition Code Register					
		IMMED.			DIRECT			INDEX			EXTEND			IMPLIED				5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#		H	I	N	Z	V	C
Shift Left Arithmetic	ASL							68	6	2	78	6	3					●	●	↕	↕	⑥	↕
	ASLA													48	2	1		●	●	↕	↕	⑥	↕
	ASLB													58	2	1		●	●	↕	↕	⑥	↕
Double Shift Left, Arithmetic	ASLD													05	3	1		●	●	↕	↕	⑥	↕
Shift Right Arithmetic	ASR							67	6	2	77	6	3					●	●	↕	↕	⑥	↕
	ASRA													47	2	1		●	●	↕	↕	⑥	↕
	ASRB													57	2	1		●	●	↕	↕	⑥	↕
Shift Right Logical	LSR							64	6	2	74	6	3					●	●	R	↕	⑥	↕
	LSRA													44	2	1		●	●	R	↕	⑥	↕
	LSRB													54	2	1		●	●	R	↕	⑥	↕
Double Shift Right Logical	LSRD													04	3	1		●	●	R	↕	⑥	↕
Store Accumulator	STAA				97	3	2	A7	4	2	B7	4	3				A → M	●	●	↕	↕	R	●
	STAB				D7	3	2	E7	4	2	F7	4	3				B → M	●	●	↕	↕	R	●
Store Double Accumulator	STD				DD	4	2	ED	5	2	FD	5	3				A → M B → M + 1	●	●	↕	↕	R	●
Subtract	SUBA	80	2	2	90	3	2	A0	4	2	B0	4	3				A – M → A	●	●	↕	↕	↕	↕
	SUBB	C0	2	2	D0	3	2	E0	4	2	F0	4	3				B – M → B	●	●	↕	↕	↕	↕
Double Subtract	SUBD	83	4	3	93	5	2	A3	6	2	B3	6	3				A . B – M : M + 1 → A : B	●	●	↕	↕	↕	↕
Subtract Accumulators	SBA													10	2	1	A – B → A	●	●	↕	↕	↕	↕
Subtract With Carry	SBCA	82	2	2	92	3	2	A2	4	2	B2	4	3				A – M – C → A	●	●	↕	↕	↕	↕
	SBCB	C2	2	2	D2	3	2	E2	4	2	F2	4	3				B – M – C → B	●	●	↕	↕	↕	↕
Transfer Accumulators	TAB													16	2	1	A → B	●	●	↕	↕	R	●
	TBA													17	2	1	B → A	●	●	↕	↕	R	●
Test Zero or Minus	TST							6D	6	2	7D	6	3				M – 00	●	●	↕	↕	R	R
	TSTA													4D	2	1	A – 00	●	●	↕	↕	R	R
	TSTB													5D	2	1	B – 00	●	●	↕	↕	R	R

The Condition Code Register notes are listed after Table 10.

Direct Addressing

In direct addressing, the address of the operand is contained in the second byte of the instruction. Direct addressing allows the user to directly address the lowest 256 bytes in the machine i.e., locations zero through 255. Enhanced execution times are achieved by storing data in these locations. In most configurations, it should be a random access memory. These are two-byte instructions.

Extended Addressing

In extended addressing, the address contained in the second byte of the instruction is used as the higher 8-bits of the address of the operand. The third byte of the instruction is used as the lower 8-bits of the address for the operand. This is an absolute address in memory. These are three-byte instructions.

Indexed Addressing

In indexed addressing, the address contained in the second byte of the instruction is added to the index register's lowest 8-bits in the CPU. The carry is then added to the higher order 8-bits of the index register. This result is then used to address memory. The modified address is held in a temporary address register so there is no change to the index register. These are two-byte instructions.

Implied Addressing

In the implied addressing mode the instruction gives the address (i.e., stack pointer, index register, etc.). These are one-byte instructions.

Relative Addressing

In relative addressing, the address contained in the second byte of the instruction is added to the program counter's lowest 8-bits plus two. The carry or borrow is then added to the high 8-bits. This allows the user to address data within a range of -126 to +129 bytes of the present instruction. These are two-byte instructions.

- **New Instructions**

In addition to the existing 6800 Instruction Set, the following new instructions are incorporated in the HD6803 Microcomputer.

ABX	Adds the 8-bit unsigned accumulator B to the 16-bit X-Register taking into account the possible carry out of the low order byte of the X-Register.
ADDD	Adds the double precision ACCD* to the double precision value M:M+1 and places the results in ACCD.
ASLD	Shifts all bits of ACCD one place to the left. Bit 0 is loaded with zero. The C bit is loaded from the most significant bit of ACCD.
LDD	Loads the contents of double precision memory location into the double accumulator A:B. The condition codes are set according to the data.
LSRD	Shifts all bits of ACCD one place to the right. Bit 15 is loaded with zero. The C bit is loaded from the least significant bit to ACCD.
MUL	Multiplies the 8 bits in accumulator A with the 8 bits in accumulator B to obtain a 16-bit unsigned number in A:B, ACCA contains MSB of result.
PSHX	The contents of the index register is pushed onto the stack at the address contained in the stack pointer. The stack pointer is decremented by 2.
PULX	The index register is pulled from the stack beginning at the current address contained in the stack pointer +1. The stack pointer is incremented by 2 in total.
STD	Stores the contents of double accumulator A:B in memory. The contents of ACCD remain unchanged.
SUBD	Subtracts the contents of M:M + 1 from the contents of double accumulator AB and places the result in ACCD.
BRN	Never branches. If effect, this instruction can be considered a two byte NOP (No operation) requiring three cycles for execution.
CPX	Internal processing modified to permit its use with any conditional branch instruction.

*ACCD' is the 16 bit register (A B) formed by concatenating the A and B accumulators The A-accumulator is the most significant byte.

Table 8 Index Register and Stack Manipulation Instructions

Pointer Operations	Mnemonic	Addressing Modes															Boolean/ Arithmetic Operation	Condition Code Register					
		IMMED.			DIRECT			INDEX			EXTND			IMPLIED				5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#		H	I	N	Z	V	C
Compare Index Reg	CPX	8C	4	3	9C	5	2	AC	6	2	BC	6	3				X − M M + 1	●	●	↕	↕	↕	↕
Decrement Index Reg	DEX													09	3	1	X − 1 → X	●	●	●	↕	●	●
Decrement Stack Pntr	DES													34	3	1	SP − 1 → SP	●	●	●	●	●	●
Increment Index Reg	INX													08	3	1	X + 1 → X	●	●	●	↕	●	●
Increment Stack Pntr	INS													31	3	1	SP + 1 → SP	●	●	●	●	●	●
Load Index Reg	LDX	CE	3	3	DE	4	2	EE	5	2	FE	5	3				M → X_H, (M + 1) → X_L	●	●	⑦	↕	R	●
Load Stack Pntr	LDS	8E	3	3	9E	4	2	AE	5	2	BE	5	3				M → SP_H, (M+1) → SP_L	●	●	⑦	↕	R	●
Store Index Reg	STX				DF	4	2	EF	5	2	FF	5	3				X_H → M, X_L → (M + 1)	●	●	⑦	↕	R	●
Store Stack Pntr	STS				9F	4	2	AF	5	2	BF	5	3				SP_H → M, SP_L → (M + 1)	●	●	⑦	↕	R	●
Index Reg → Stack Pntr	TXS													35	3	1	X − 1 → SP	●	●	●	●	●	●
Stack Pntr → Index Reg	TSX													30	3	1	SP + 1 → X	●	●	●	●	●	●
Add	ABX													3A	3	1	B + X → X	●	●	●	●	●	●
Push Data	PSHX													3C	4	1	X_L → M_{sp}, SP − 1 → SP X_H → M_{sp}, SP − 1 → SP	●	●	●	●	●	●
Pull Data	PULX													38	5	1	SP + 1 → SP, M_{sp} → X_H SP + 1 → SP, M_{sp} → X_L	●	●	●	●	●	●

The Condition Code Register notes are listed after Table 10

Table 9 Jump and Branch Instructions

Operations	Mnemonic	Addressing Modes															Branch Test	Condition Code Register					
		RELATIVE			DIRECT			INDEX			EXTND			IMPLIED				5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#		H	I	N	Z	V	C
Branch Always	BRA	20	3	2													None	●	●	●	●	●	●
Branch Never	BRN	21	3	2													None	●	●	●	●	●	●
Branch If Carry Clear	BCC	24	3	2													C = 0	●	●	●	●	●	●
Branch If Carry Set	BCS	25	3	2													C = 1	●	●	●	●	●	●
Branch If = Zero	BEQ	27	3	2													Z = 1	●	●	●	●	●	●
Branch If ≥ Zero	BGE	2C	3	2													N ⊕ V = 0	●	●	●	●	●	●
Branch If > Zero	BGT	2E	3	2													Z + (N ⊕ V) = 0	●	●	●	●	●	●
Branch If Higher	BHI	22	3	2													C + Z = 0	●	●	●	●	●	●
Branch If ≤ Zero	BLE	2F	3	2													Z + (N ⊕ V) = 1	●	●	●	●	●	●
Branch If Lower Or Same	BLS	23	3	2													C + Z = 1	●	●	●	●	●	●
Branch If < Zero	BLT	2D	3	2													N ⊕ V = 1	●	●	●	●	●	●
Branch If Minus	BMI	2B	3	2													N = 1	●	●	●	●	●	●
Branch If Not Equal Zero	BNE	26	3	2													Z = 0	●	●	●	●	●	●
Branch If Overflow Clear	BVC	28	3	2													V = 0	●	●	●	●	●	●
Branch If Overflow Set	BVS	29	3	2													V = 1	●	●	●	●	●	●
Branch If Plus	BPL	2A	3	2													N = 0	●	●	●	●	●	●
Branch To Subroutine	BSR	8D	6	2														●	●	●	●	●	●
Jump	JMP							6E	3	2	7E	3	3					●	●	●	●	●	●
Jump To Subroutine	JSR				9D	5	2	AD	6	2	BD	6	3					●	●	●	●	●	●
No Operation	NOP													01	2	1	Advances Prog. Cntr. Only	●	●	●	●	●	●
Return From Interrupt	RTI													3B	10	1		⑧					
Return From Subroutine	RTS													39	5	1		●	●	●	●	●	●
Software Interrupt	SWI													3F	12	1		●	S	●	●	●	●
Wait for Interrupt	WAI													3E	9	1		●	⑨	●	●	●	●

2

Table10 Condition Code Register Manipulation Instructions

Operations	Mnemonic	AddressingModes			Boolean Operation	Condition Code Register					
		IMPLIED				5	4	3	2	1	0
		OP	~	#		H	I	N	Z	V	C
Clear Carry	CLC	0C	2	1	0 → C	●	●	●	●	●	R
Clear Interrupt Mask	CLI	0E	2	1	0 → I	●	R	●	●	●	●
Clear Overflow	CLV	0A	2	1	0 → V	●	●	●	●	R	●
Set Carry	SEC	0D	2	1	1 → C	●	●	●	●	●	S
Set Interrupt Mask	SEI	0F	2	1	1 → I	●	S	●	●	●	●
Set Overflow	SEV	0B	2	1	1 → V	●	●	●	●	S	●
Accumulator A → CCR	TAP	06	2	1	A → CCR	⑩					
CCR → Accumulator A	TPA	07	2	1	CCR → A	●	●	●	●	●	●

Condition Code Register Notes: (Bit set it test is true and cleared otherwise)

① (Bit V) Test. Result = 10000000?
② (Bit C) Test Result ≠ 00000000?
③ (Bit C) Test Decimal value of most significant BCD Character greater than nine? (Not cleared if previously set)
④ (Bit V) Test. Operand = 10000000 prior to execution?
⑤ (Bit V) Test Operand = 01111111 prior to execution?
⑥ (Bit V) Test: Set equal to result of N ⊕ C after shift has occurred.
⑦ (Bit N) Test· Result less than zero? (Bit 15 = 1)
⑧ (All) Load Condition Code Register from Stack. (See Special Operations)
⑨ (Bit I) Set when interrupt occurs. If previously set, a Non-Maskable Interrupt is required to exit the wait state.
⑩ (All) Set according to the contents of Accumulator A
⑪ (Bit C) Set equal to result of Bit 7 (ACCB)

Table 11 Instruction Execution Times in Machine Cycle

	ACCX	Imme-diate	Direct	Ex-tended	In-dexed	Im-plied	Re-lative		ACCX	Imme-diate	Direct	Ex-tended	In-dexed	Im-plied	Re-lative
ABA	•	•	•	•	•	2	•	INX	•	•	•	•	•	3	•
ABX	•	•	•	•	•	3	•	JMP	•	•	•	3	3	•	•
ADC	•	2	3	4	4	•	•	JSR	•	•	5	6	6	•	•
ADD	•	2	3	4	4	•	•	LDA	•	2	3	4	4	•	•
ADDD	•	4	5	6	6	•	•	LDD	•	3	4	5	5	•	•
AND	•	2	3	4	4	•	•	LDS	•	3	4	5	5	•	•
ASL	2	•	•	6	6	•	•	LDX	•	3	4	5	5	•	•
ASLD	•	•	•	•	•	3	•	LSR	2	•	•	6	6	•	•
ASR	2	•	•	6	6	•	•	LSRD	•	•	•	•	•	3	•
BCC	•	•	•	•	•	•	3	MUL	•	•	•	•	•	10	•
BCS	•	•	•	•	•	•	3	NEG	2	•	•	6	6	•	•
BEQ	•	•	•	•	•	•	3	NOP	•	•	•	•	•	2	•
BGE	•	•	•	•	•	•	3	ORA	•	2	3	4	4	•	•
BGT	•	•	•	•	•	•	3	PSH	3	•	•	•	•	•	•
BHI	•	•	•	•	•	•	3	PSHX	•	•	•	•	•	4	•
BIT	•	2	3	4	4	•	•	PUL	4	•	•	•	•	•	•
BLE	•	•	•	•	•	•	3	PULX	•	•	•	•	•	5	•
BLS	•	•	•	•	•	•	3	ROL	2	•	•	6	6	•	•
BLT	•	•	•	•	•	•	3	ROR	2	•	•	6	6	•	•
BMI	•	•	•	•	•	•	3	RTI	•	•	•	•	•	10	•
BNE	•	•	•	•	•	•	3	RTS	•	•	•	•	•	5	•
BPL	•	•	•	•	•	•	3	SBA	•	•	•	•	•	2	•
BRA	•	•	•	•	•	•	3	SBC	•	2	3	4	4	•	•
BRN	•	•	•	•	•	•	3	SEC	•	•	•	•	•	2	•
BSR	•	•	•	•	•	•	6	SEI	•	•	•	•	•	2	•
BVC	•	•	•	•	•	•	3	SEV	•	•	•	•	•	2	•
BVS	•	•	•	•	•	•	3	STA	•	•	3	4	4	•	•
CBA	•	•	•	•	•	2	•	STD	•	•	4	5	5	•	•
CLC	•	•	•	•	•	2	•	STS	•	•	4	5	5	•	•
CLI	•	•	•	•	•	2	•	STX	•	•	4	5	5	•	•
CLR	2	•	•	6	6	•	•	SUB	•	2	3	4	4	•	•
CLV	•	•	•	•	•	2	•	SUBD	•	4	5	6	6	•	•
CMP	•	2	3	4	4	•	•	SWI	•	•	•	•	•	12	•
COM	2	•	•	6	6	•	•	TAB	•	•	•	•	•	2	•
CPX	•	4	5	6	6	•	•	TAP	•	•	•	•	•	2	•
DAA	•	•	•	•	•	2	•	TBA	•	•	•	•	•	2	•
DEC	2	•	•	6	6	•	•	TPA	•	•	•	•	•	2	•
DES	•	•	•	•	•	3	•	TST	2	•	•	6	6	•	•
DEX	•	•	•	•	•	3	•	TSX	•	•	•	•	•	3	•
EOR	•	2	3	4	4	•	•	TXS	•	•	•	•	•	3	•
INC	2	•	•	6	6	•	•	WAI	•	•	•	•	•	9	•
INS	•	•	•	•	•	3	•								

- **Summary of Cycle by Cycle Operation**

Table 12 provides a detailed description of the information present on the Address Bus, Data Bus, and the Read/Write line (R/$\overline{W}$) during each cycle for each instruction.

This information is useful in comparing actual with expected results during debug of both software and hardware as the control program is executed. The information is categorized in groups according to addressing mode and number of cycles per instruction. (In general, instructions with the same addressing mode and number of cycles execute in the same manner; exceptions are indicated in the table).

Table 12 Cycle by Cycle Operation

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/$\overline{W}$ Line	Data Bus
IMMEDIATE					
ADC EOR ADD LDA AND ORA BIT SBC CMP SUB	2	1	Op Code Address	1	Op Code
		2	Op Code Address + 1	1	Operand Data
LDS LDX LDD	3	1	Op Code Address	1	Op Code
		2	Op Code Address + 1	1	Operand Data (High Order Byte)
		3	Op Code Address + 2	1	Operand Data (Low Order Byte)
CPX SUBD ADDD	4	1	Op Code Address	1	Op Code
		2	Op Code Address + 1	1	Operand Data (High Order Byte)
		3	Op Code Address + 2	1	Operand Data (Low Order Byte)
		4	Address Bus FFFF	1	Low Byte of Restart Vector
DIRECT					
ADC EOR ADD LDA AND ORA BIT SBC CMP SUB	3	1	Op Code Address	1	Op Code
		2	Op Code Address + 1	1	Address of Operand
		3	Address of Operand	1	Operand Data
STA	3	1	Op Code Address	1	Op Code
		2	Op Code Address + 1	1	Destination Address
		3	Destination Address	0	Data from Accumulator
LDS LDX LDD	4	1	Op Code Address	1	Op Code
		2	Op Code Address + 1	1	Address of Operand
		3	Address of Operand	1	Operand Data (High Order Byte)
		4	Operand Address + 1	1	Operand Data (Low Order Byte)
STS STX STD	4	1	Op Code Address	1	Op Code
		2	Op Code Address + 1	1	Address of Operand
		3	Address of Operand	0	Register Data (High Order Byte)
		4	Address of Operand + 1	0	Register Data (Low Order Byte)
CPX SUBD ADDD	5	1	Op Code Address	1	Op Code
		2	Op Code Address + 1	1	Address of Operand
		3	Operand Address	1	Operand Data (High Order Byte)
		4	Operand Address + 1	1	Operand Data (Low Order Byte)
		5	Address Bus FFFF	1	Low Byte of Restart Vector
JSR	5	1	Op Code Address	1	Op Code
		2	Op Code Address + 1	1	Irrelevant Data
		3	Subroutine Address	1	First Subroutine Op Code
		4	Stack Pointer	0	Return Address (Low Order Byte)
		5	Stack Pointer + 1	0	Return Address (High Order Byte)

(Continued)

Table 12 Cycle by Cycle Operation (Continued)

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/$\overline{W}$ Line	Data Bus
INDEXED					
JMP	3	1	Op Code Address	1	Op Code
		2	Op Code Address + 1	1	Offset
		3	Address Bus FFFF	1	Low Byte of Restart Vector
ADC EOR	4	1	Op Code Address	1	Op Code
ADD LDA		2	Op Code Address + 1	1	Offset
AND ORA		3	Address Bus FFFF	1	Low Byte of Restart Vector
BIT SBC		4	Index Register Plus Offset	1	Operand Data
CMP SUB					
STA	4	1	Op Code Address	1	Op Code
		2	Op Code Address + 1	1	Offset
		3	Address Bus FFFF	1	Low Byte of Restart Vector
		4	Index Register Plus Offset	0	Operand Data
LDS	5	1	Op Code Address	1	Op Code
LDX		2	Op Code Address + 1	1	Offset
LDD		3	Address Bus FFFF	1	Low Byte of Restart Vector
LDD		4	Index Register Plus Offset	1	Operand Data (High Order Byte)
		5	Index Register Plus Offset + 1	1	Operand Data (Low Order Byte)
STS	5	1	Op Code Address	1	Op Code
STX		2	Op Code Address + 1	1	Offset
STD		3	Address Bus FFFF	1	Low Byte of Restart Vector
		4	Index Register Plus Offset	0	Operand Data (High Order Byte)
		5	Index Register Plus Offset + 1	0	Operand Data (Low Order Byte)
ASL LSR	6	1	Op Code Address	1	Op Code
ASR NEG		2	Op Code Address + 1	1	Offset
CLR ROL		3	Address Bus FFFF	1	Low Byte of Restart Vector
COM ROR		4	Index Register Plus Offset	1	Current Operand Data
DEC TST*		5	Address Bus FFFF	1	Low Byte of Restart Vector
INC		6	Index Register Plus Offset	0	New Operand Data
CPX	6	1	Op Code Address	1	Op Code
SUBD		2	Op Code Address + 1	1	Offset
ADDD		3	Address Bus FFFF	1	Low Byte of Restart Vector
		4	Index Register + Offset	1	Operand Data (High Order Byte)
		5	Index Register + Offset + 1	1	Operand Data (Low Order Byte)
		6	Address Bus FFFF	1	Low Byte of Restart Vector
JSR	6	1	Op Code Address	1	Op Code
		2	Op Code Address + 1	1	Offset
		3	Address Bus FFFF	1	Low Byte of Restart Vector
		4	Index Register + Offset	1	First Subroutine Op Code
		5	Stack Pointer	0	Return Address (Low Order Byte)
		6	Stack Pointer − 1	0	Return Address (High Order Byte)

* In the TST instruction, R/$\overline{W}$ line of the sixth cycle is "1" level, and AB = FFFF, DB = Low Byte of Reset Vector.

(Continued)

Table 12 Cycle by Cycle Operation (Continued)

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/$\overline{W}$ Line	Data Bus
EXTENDED					
JMP	3	1	Op Code Address	1	Op Code
		2	Op Code Address + 1	1	Jump Address (High Order Byte)
		3	Op Code Address + 2	1	Jump Address (Low Order Byte)
ADC EOR	4	1	Op Code Address	1	Op Code
ADD LDA		2	Op Code Address + 1	1	Address of Operand (High Order Byte)
AND ORA		3	Op Code Address + 2	1	Address of Operand (Low Order Byte)
BIT SBC		4	Address of Operand	1	Operand Data
CMP SUB					
STA	4	1	Op Code Address	1	Op Code
		2	Op Code Address + 1	1	Destination Address (High Order Byte)
		3	Op Code Address + 2	1	Destination Address (Low Order Byte)
		4	Operand Destination Address	0	Data from Accumulator
LDS	5	1	Op Code Address	1	Op Code
LDX		2	Op Code Address + 1	1	Address of Operand (High Order Byte)
LDD		3	Op Code Address + 2	1	Address of Operand (Low Order Byte)
		4	Address of Operand	1	Operand Data (High Order Byte)
		5	Address of Operand + 1	1	Operand Data (Low Order Byte)
STS	5	1	Op Code Address	1	Op Code
STX		2	Op Code Address + 1	1	Address of Operand (High Order Byte)
STD		3	Op Code Address + 2	1	Address of Operand (Low Order Byte)
		4	Address of Operand	0	Operand Data (High Order Byte)
		5	Address of Operand + 1	0	Operand Data (Low Order Byte)
ASL LSR	6	1	Op Code Address	1	Op Code
ASR NEG		2	Op Code Address + 1	1	Address of Operand (High Order Byte)
CLR ROL		3	Op Code Address + 2	1	Address of Operand (Low Order Byte)
COM ROR		4	Address of Operand	1	Current Operand Data
DEC TST*		5	Address Bus FFFF	1	Low Byte of Restart Vector
INC		6	Address of Operand	0	New Operand Data
CPX	6	1	Op Code Address	1	Op Code
SUBD		2	Op Code Address + 1	1	Operand Address (High Order Byte)
ADDD		3	Op Code Address + 2	1	Operand Address (Low Order Byte)
		4	Operand Address	1	Operand Data (High Order Byte)
		5	Operand Address + 1	1	Operand Data (Low Order Byte)
		6	Address Bus FFFF	1	Low Byte of Restart Vector
JSR	6	1	Op Code Address	1	Op Code
		2	Op Code Address + 1	1	Address of Subroutine (High Order Byte)
		3	Op Code Address + 2	1	Address of Subroutine (Low Order Byte)
		4	Subroutine Starting Address	1	Op Code of Next Instruction
		5	Stack Pointer	0	Return Address (Low Order Byte)
		6	Stack Pointer − 1	0	Return Address (High Order Byte)

* In the TST instruction, R/$\overline{W}$ line of the sixth cycle is "1" level, and AB = FFFF, DB = Low Byte of Reset Vector

(Continued)

Table 12 Cycle by Cycle Operation (Continued)

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/$\overline{W}$ Line	Data Bus
IMPLIED					
ABA DAA SEC ASL DEC SEI ASR INC SEV CBA LSR TAB CLC NEG TAP CLI NOP TBA CLR ROL TPA CLV ROR TST COM SBA	2	1	Op Code Address	1	Op Code
		2	Op Code Address + 1	1	Op Code of Next Instruction
ABX	3	1	Op Code Address	1	Op Code
		2	Op Code Address + 1	1	Irrelevant Data
		3	Address Bus FFFF	1	Low Byte of Restart Vector
ASLD	3	1	Op Code Address	1	Op Code
LSRD		2	Op Code Address + 1	1	Irrelevant Data
		3	Address Bus FFFF	1	Low Byte of Restart Vector
DES	3	1	Op Code Address	1	Op Code
INS		2	Op Code Address + 1	1	Op Code of Next Instruction
		3	Previous Register Contents	1	Irrelevant Data
INX	3	1	Op Code Address	1	Op Code
DEX		2	Op Code Address + 1	1	Op Code of Next Instruction
		3	Address Bus FFFF	1	Low Byte of Restart Vector
PSHA	3	1	Op Code Address	1	Op Code
PSHB		2	Op Code Address + 1	1	Op Code of Next Instruction
		3	Stack Pointer	0	Accumulator Data
TSX	3	1	Op Code Address	1	Op Code
		2	Op Code Address + 1	1	Op Code of Next Instruction
		3	Stack Pointer	1	Irrelevant Data
TXS	3	1	Op Code Address	1	Op Code
		2	Op Code Address + 1	1	Op Code of Next Instruction
		3	Address Bus FFFF	1	Low Byte of Restart Vector
PULA	4	1	Op Code Address	1	Op Code
PULB		2	Op Code Address + 1	1	Op Code of Next Instruction
		3	Stack Pointer	1	Irrelevant Data
		4	Stack Pointer + 1	1	Operand Data from Stack
PSHX	4	1	Op Code Address	1	Op Code
		2	Op Code Address + 1	1	Irrelevant Data
		3	Stack Pointer	0	Index Register (Low Order Byte)
		4	Stack Pointer − 1	0	Index Register (High Order Byte)
PULX	5	1	Op Code Address	1	Op Code
		2	Op Code Address + 1	1	Irrelevant Data
		3	Stack Pointer	1	Irrelevant Data
		4	Stack Pointer + 1	1	Index Register (High Order Byte)
		5	Stack Pointer +2	1	Index Register (Low Order Byte)
RTS	5	1	Op Code Address	1	Op Code
		2	Op Code Address + 1	1	Irrelevant Data
		3	Stack Pointer	1	Irrelevant Data
		4	Stack Pointer + 1	1	Address of Next Instruction (High Order Byte)
		5	Stack Pointer + 2	1	Address of Next Instruction (Low Order Byte)
WAI**	9	1	Op Code Address	1	Op Code
		2	Op Code Address + 1	1	Op Code of Next Instruction
		3	Stack Pointer	0	Return Address (Low Order Byte)
		4	Stack Pointer − 1	0	Return Address (High Order Byte)

(Continued)

Table 12 Cycle by Cycle Operation (Continued)

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/W̄ Line	Data Bus
WAI**		5	Stack Pointer – 2	0	Index Register (Low Order Byte)
		6	Stack Pointer – 3	0	Index Register (High Order Byte)
		7	Stack Pointer – 4	0	Contents of Accumulator A
		8	Stack Pointer – 5	0	Contents of Accumulator B
		9	Stack Pointer – 6	0	Contents of Cond. Code Register
MUL	10	1	Op Code Address	1	Op Code
		2	Op Code Address + 1	1	Irrelevant Data
		3	Address Bus FFFF	1	Low Byte of Restart Vector
		4	Address Bus FFFF	1	Low Byte of Restart Vector
		5	Address Bus FFFF	1	Low Byte of Restart Vector
		6	Address Bus FFFF	1	Low Byte of Restart Vector
		7	Address Bus FFFF	1	Low Byte of Restart Vector
		8	Address Bus FFFF	1	Low Byte of Restart Vector
		9	Address Bus FFFF	1	Low Byte of Restart Vector
		10	Address Bus FFFF	1	Low Byte of Restart Vector
RTI	10	1	Op Code Address	1	Op Code
		2	Op Code Address + 1	1	Irrelevant Data
		3	Stack Pointer	1	Irrelevant Data
		4	Stack Pointer + 1	1	Contents of Cond. Code Reg. from Stack
		5	Stack Pointer + 2	1	Contents of Accumulator B from Stack
		6	Stack Pointer + 3	1	Contents of Accumulator A from Stack
		7	Stack Pointer + 4	1	Index Register from Stack (High Order Byte)
		8	Stack Pointer + 5	1	Index Register from Stack (Low Order Byte)
		9	Stack Pointer + 6	1	Next Instruction Address from Stack (High Order Byte)
		10	Stack Pointer + 7	1	Next Instruction Address from Stack (Low Order Byte)
SWI	12	1	Op Code Address	1	Op Code
		2	Op Code Address + 1	1	Irrelevant Data
		3	Stack Pointer	0	Return Address (Low Order Byte)
		4	Stack Pointer – 1	0	Return Address (High Order Byte)
		5	Stack Pointer – 2	0	Index Register (Low Order Byte)
		6	Stack Pointer – 3	0	Index Register (High Order Byte)
		7	Stack Pointer – 4	0	Contents of Accumulator A
		8	Stack Pointer – 5	0	Contents of Accumulator B
		9	Stack Pointer – 6	0	Contents of Cond. Code Register
		10	Stack Pointer – 7	1	Irrelevant Data
		11	Vector Address FFFA (Hex)	1	Address of Subroutine (High Order Byte)
		12	Vector Address FFFB (Hex)	1	Address of Subroutine (Low Order Byte)

(Continued)

** While the MPU is in the "Wait" state, its bus state will appear as a series of MPU reads of an address which is seven locations less than the original contents of the Stack Pointer. Contrary to the HD6800, none of the ports are driven to the high impedance state by a WAI instruction.

Table 12 Cycle by Cycle Operation (Continued)

RELATIVE

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/$\overline{W}$ Line	Data Bus
BCC BHT BNE BCS BLE BPL BEQ BLS BRA BGE BLT BVC BGT BMT BVS BRN	3	1 2 3	Op Code Address Op Code Address + 1 Address Bus FFFF	1 1 1	Op Code Branch Offset Low Byte of Restart Vector
BSR	6	1 2 3 4 5 6	Op Code Address Op Code Address + 1 Address Bus FFFF Subroutine Starting Address Stack Pointer Stack Pointer − 1	1 1 1 1 0 0	Op Code Branch Offset Low Byte of Restart Vector Op Code of Next Instruction Return Address (Low Order Byte) Return Address (High Order Byte)

- **Summary of Undefined Instruction Operations**

The HD6803 has 36 underfined instructions. When these are carried out, the contents of Register and Memory in MPU change at random.

When the op codes (4E, 5E) are used to execute. the MPU continues to increase the program counter and it will not stop until the Reset signal enters. These op codes are used to test the LSI.

Table 13 Op codes Map

HD6803 MICROPROCESSOR INSTRUCTIONS

OP CODE						ACC A	ACC B	IND	EXT	ACCA or SP				ACCB or X				
										IMM	DIR	IND	EXT	IMM	DIR	IND	EXT	
LO \ HI		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111	
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0000	0		SBA	BRA	TSX	NEG				SUB								0
0001	1	NOP	CBA	BRN	INS					CMP								1
0010	2			BHI	PULA (+1)					SBC								2
0011	3			BLS	PULB (+1)	COM				•	SUBD (+2)			•	ADDD (+2)			3
0100	4	LSRD (+1)		BCC	DES	LSR				AND								4
0101	5	ASLD (+1)		BCS	TXS					BIT								5
0110	6	TAP	TAB	BNE	PSHA	ROR				LDA								6
0111	7	TPA	TBA	BEQ	PSHB	ASR					STA				STA			7
1000	8	INX (+1)		BVC	PULX (+2)	ASL				EOR								8
1001	9	DEX (+1)	DAA	BVS	RTS (+2)	ROL				ADC								9
1010	A	CLV		BPL	ABX	DEC				ORA								A
1011	B	SEV	ABA	BMI	RTI (+7)					ADD								B
1100	C	CLC		BGE	PSHX (+1)	INC				•	CPX (+2)			•	LDD (+1)			C
1101	D	SEC		BLT	MUL (+7)	TST				BSR (+4)	JSR (+2)			* (+1)	STD (+1)			D
1110	E	CLI		BGT	WAI (+6)	**		JMP (-3)		•	LDS (+1)			•	LDX (+1)			E
1111	F	SEI		BLE	SWI (+9)	CLR				* (+1)	STS (+1)			* (+1)	STX (+1)			F
BYTE/CYCLE		1/2	1/2	2/3	1/3	1/2	1/2	2/6	3/6	2/2	2/3	2/4	3/4	2/2	2/3	2/4	3/4	

[NOTES] 1) Undefined Op codes are marked with ▨.

2) ( ) indicate that the number in parenthesis must be added to the cycle count for that instruction.

3) The instructions shown below are all 3 bytes and are marked with "*".
Immediate addressing mode of SUBD, CPX, LDS, ADDD, LDD and LDX instructions, and undefined op codes (8F, CD, CF).

4) The Op codes (4E, 5E) are 1 byte/∞ cycles instructions, and are marked with "**"

HITACHI

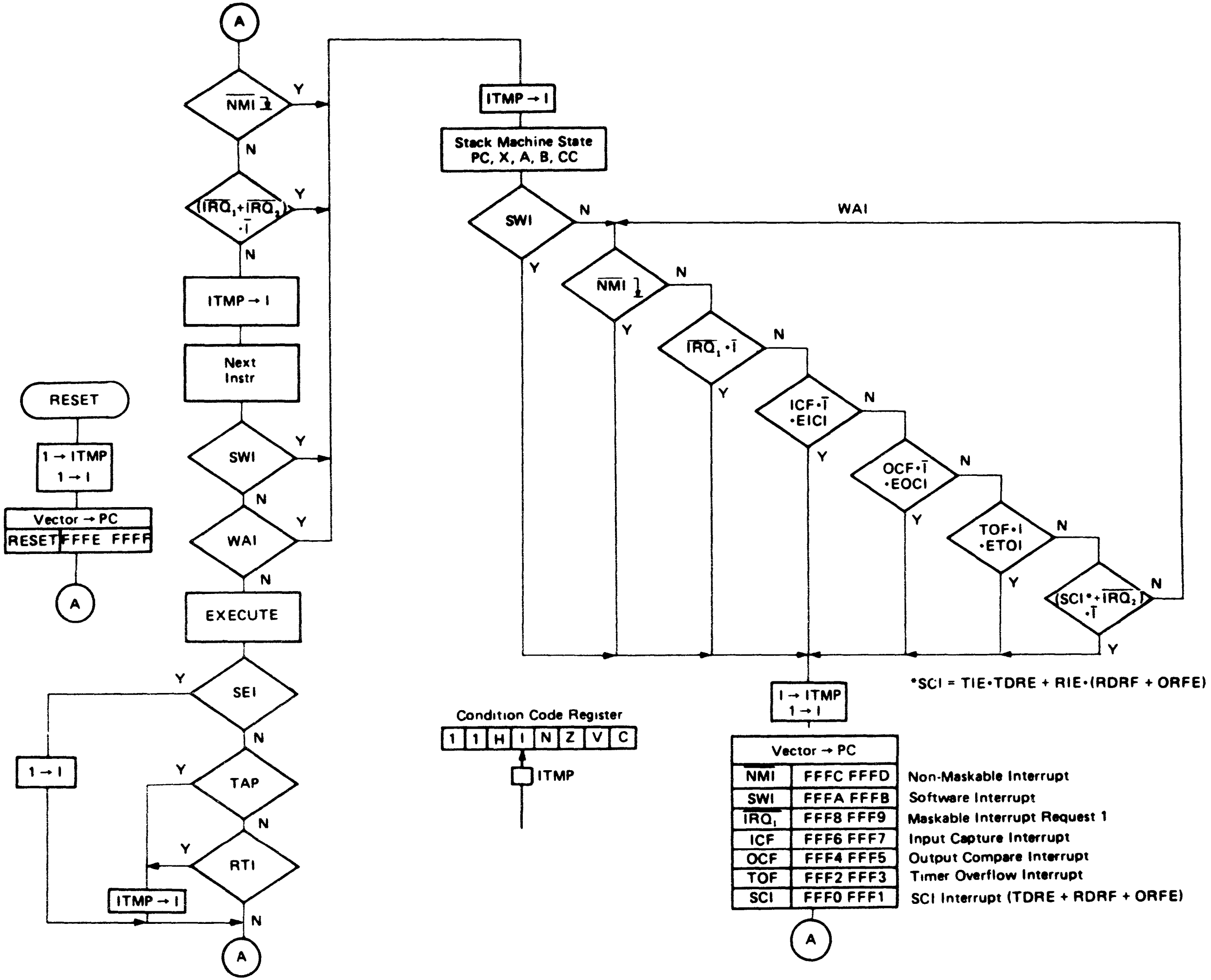

A
NMI ⤓
Y
N
(IRQ₁+IRQ₂)·Ī
Y
N
ITMP → I
Next Instr
SWI
Y
N
WAI
Y
N
EXECUTE
SEI
Y
N
1 → I
TAP
Y
N
RTI
Y
N
ITMP → I
A
RESET
1 → ITMP
1 → I
Vector → PC
RESET FFFE FFFF
A
ITMP → I
Stack Machine State PC, X, A, B, CC
SWI
N
Y
WAI
NMI ⤓
N
Y
IRQ₁·Ī
N
Y
ICF·Ī ·EICI
N
Y
OCF·Ī ·EOCI
N
Y
TOF·I ·ETOI
N
Y
(SCI*+IRQ₂)·Ī
N
Y
*SCI = TIE·TDRE + RIE·(RDRF + ORFE)
I → ITMP
1 → I
Condition Code Register
1 1 H I N Z V C
ITMP
Vector → PC
NMI FFFC FFFD Non-Maskable Interrupt
SWI FFFA FFFB Software Interrupt
IRQ₁ FFF8 FFF9 Maskable Interrupt Request 1
ICF FFF6 FFF7 Input Capture Interrupt
OCF FFF4 FFF5 Output Compare Interrupt
TOF FFF2 FFF3 Timer Overflow Interrupt
SCI FFF0 FFF1 SCI Interrupt (TDRE + RDRF + ORFE)
A

Figure 16 Interrupt Flowchart

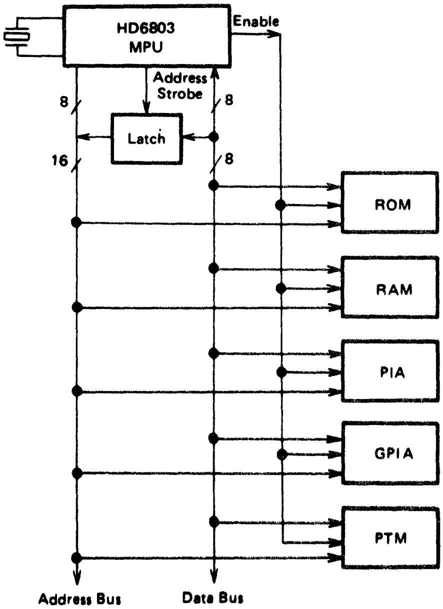


Figure 17 HD6803 MPU Expanded Multiplexed Bus

- **Caution for the HD6803 Family SCI, TIMER Status Flag**

The flags shown in Table 14 are cleared by reading/writing (flag reset condition 2) the data register corresponding to each flag after reading the status register (flag reset condition 1).

To clear the flag correctly, take the following procedure:

1. Read the status register.
2. Test the flag.
3. Read the data register.

Table 14 Status Flag Reset Conditions

	Status Flag	Flag Reset Condition 1 (Status Register)	Flag Reset Condition 2 (Data Register)
TIMER	ICF	When each flag is "1", TRCSR/Read	ICR/Read
	OCF		OCR/Write
	TOF		TC/Read
SCI	RDRF	When each flag is "1", TRCSR/Read	RDR/Read
	ORFE		
	TDRE		TDR/Write

HD6809, HD68A09, HD68B09 MPU (Micro Processing Unit)

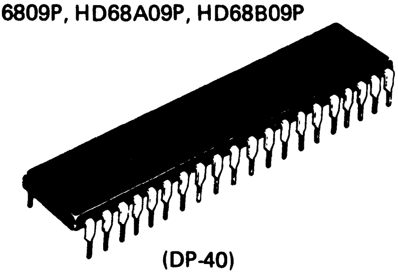
The HD6809 is a revolutionary high performance 8-bit microprocessor which supports modern programming techniques such as position independence, reentrancy, and modular programming.

This third-generation addition to the HMCS6800 family has major architectural improvements which include additional registers, instructions and addressing modes.

The basic instructions of any computer are greatly enhanced by the presence of powerful addressing modes. The HD6809 has the most complete set of addressing modes available on any 8-bit microprocessor today.

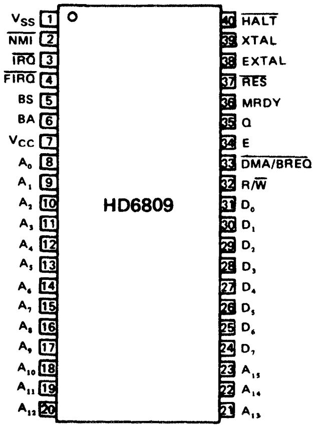
The HD6809 has hardware and software features which make it an ideal processor for higher level language execution or standard controller applications.

HD6809P, HD68A09P, HD68B09P



(DP-40)

■ PIN ARRANGEMENT



(Top View)

HD6800 COMPATIBLE

- Hardware – Interfaces with All HMCS6800 Peripherals
- Software – Upward Source Code Compatible Instruction Set and Addressing Modes

■ ARCHITECTURAL FEATURES

- Two 16-bit Index Registers
- Two 16-bit Indexable Stack Pointers
- Two 8-bit Accumulators can be Concatenated to Form One 16-Bit Accumulator
- Direct Page Register Allows Direct Addressing Throughout Memory

■ HARDWARE FEATURES

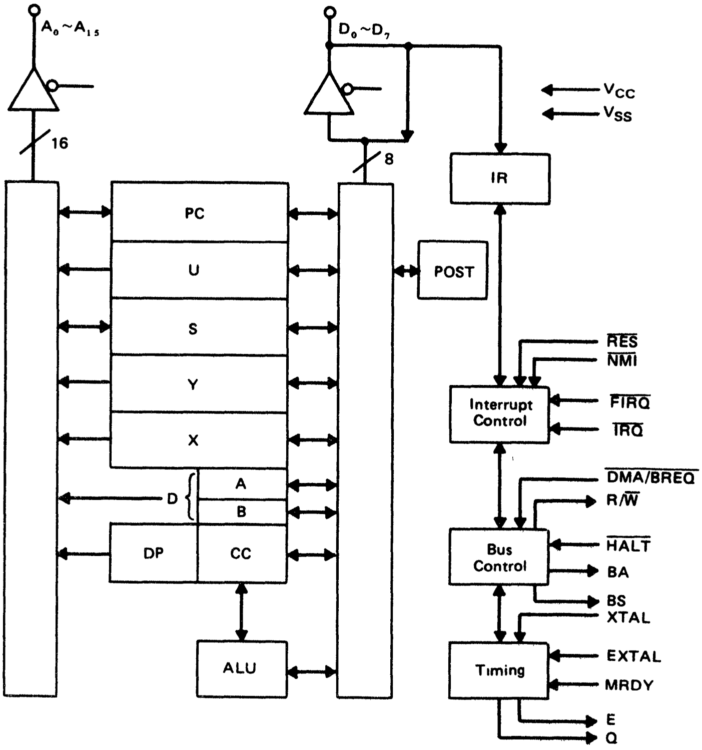
- On Chip Oscillator
- $\overline{\text{DMA/BREQ}}$ Allows DMA Operation or Memory Refresh
- Fast Interrupt Request Input Stacks Only Condition Code Register and Program Counter
- MRDY Input Extends Data Access Times for Use With Slow Memory
- Interrupt Acknowledge Output Allows Vectoring By Devices
- SYNC Acknowledge Output Allows for Synchronization to External Event
- Single Bus-Cycle RESET
- Single 5-Volt Supply Operation
- NMI Blocked After RESET Until After First Load of Stack Pointer
- Early Address Valid Allows Use With Slower Memories
- Early Write-Data for Dynamic Memories
- Compatible with MC6809, MC68A09 and MC68B09

■ SOFTWARE FEATURES

- 10 Addressing Modes
  - HMCS6800 Upward Compatible Addressing Modes
  - Direct Addressing Anywhere in Memory Map
  - Long Relative Branches
  - Program Çounter Relative
  - True Indirect Addressing
  - Expanded Indexed Addressing:

  - 0, 5, 8, or 16-bit Constant Offsets
  - 8, or 16-bit Accumulator Offsets
  - Auto-Increment/Decrement by 1 or 2
- Improved Stack Manipulation
- 1464 Instructions with Unique Addressing Modes
- 8 x 8 Unsigned Multiply
- 16-bit Arithmetic
- Transfer/Exchange All Registers
- Push/Pull Any Registers or Any Set of Registers
- Load Effective Address

■ BLOCK DIAGRAM



■ ABSOLUTE MAXIMUM RATINGS

Item	Symbol	Value	Unit
Supply Voltage	V_{CC}*	-0.3 ~ +7.0	V
Input Voltage	V_{in}*	-0.3 ~ +7.0	V
Operating Temperature	T_{opr}	-20 ~ +75	°C
Storage Temperature	T_{stg}	-55 ~ +150	°C

* With respect to V_{SS} (SYSTEM GND)

(NOTE) Permanent LSI damage may occur if maximum ratings are exceeded. Normal operation should be under recommended operating conditions. If these conditions are exceeded, it could affect reliability of LSI.

■ RECOMMENDED OPERATING CONDITIONS

Item	Symbol		min	typ	max	Unit
Supply Voltage	V_{CC}*		4.75	5.0	5.25	V
Input Voltage	V_{IL}*		-0.3	–	0.8	V
	V_{IH}*	Logic (Ta = 0 ~ +75°C)	2.0	–	V_{CC}	V
		Logic (Ta = -20 ~ 0°C)	2.2	–	V_{CC}	
		$\overline{RES}$	4.0	–	V_{CC}	
Operating Temperature	T_{opr}		-20	25	75	°C

* With respect to V_{SS} (SYSTEM GND)

■ ELECTRICAL CHARACTERISTICS

● DC CHARACTERISTICS (V_{CC}=5V±5%, V_{SS} = 0V, Ta = -20~+75°C, unless otherwise noted.)

Item		Symbol	Test Condition	HD6809			HD68A09			HD68B09			Unit
				min	typ*	max	min	typ*	max	min	typ*	max	
Input "High" Voltage	Except $\overline{RES}$	V_{IH}	Ta = 0 ~ +75°C	2.0	–	V_{CC}	2.0	–	V_{CC}	2.0	–	V_{CC}	V
			Ta = -20 ~ 0°C	2.2	–	V_{CC}	2.2	–	V_{CC}	2.2	–	V_{CC}	
	$\overline{RES}$			4.0	–	V_{CC}	4.0	–	V_{CC}	4.0	–	V_{CC}	
Input "Low" Voltage		V_{IL}		-0.3	–	0.8	-0.3	–	0.8	-0.3	–	0.8	V
Input Leakage Current	Except EXTAL, XTAL	Iin	Vin=0~5.25V, V_{CC}=max	-2.5	–	2.5	-2.5	–	2.5	-2.5	–	2.5	μA
Three State (Off State) Input Current	D_0~D_7	I_{TSI}	Vin=0.4~2.4V, V_{CC}=max	-10	–	10	-10	–	10	-10	–	10	μA
	A_0~A_{15}, $R/\overline{W}$			-100	–	100	-100	–	100	-100	–	100	
Output "High" Voltage	D_0~D_7	V_{OH}	I_{LOAD}=-205μA, V_{CC}=min	2.4	–	–	2.4	–	–	2.4	–	–	V
	A_0~A_{15}, $R/\overline{W}$, Q, E		I_{LOAD}=-145μA, V_{CC}=min	2.4	–	–	2.4	–	–	2.4	–	–	
	BA, BS		I_{LOAD}=-100μA, V_{CC}=min	2.4	–	–	2.4	–	–	2.4	–	–	
Output "Low" Voltage		V_{OL}	I_{LOAD}=2mA	–	–	0.5	–	–	0.5	–	–	0.5	V
Power Dissipation		P_D		–	–	1.0	–	–	1.0	–	–	1.0	W
Input Capacitance	D_0~D_7	Cin	Vin=0V, Ta=25°C, f=1MHz	–	10	15	–	10	15	–	10	15	pF
	Except D_0~D_7			–	7	10	–	7	10	–	7	10	
Output Capacitance	A_0~A_{15}, $R/\overline{W}$, BA, BS	Cout		–	–	12	–	–	12	–	–	12	pF

*Ta=25°C, V_{CC}=5V

- **AC CHARACTERISTICS (V_{CC}=5V±5%, V_{SS} = 0V, Ta = -20~+75°C, unless otherwise noted.)**

1. CLOCK TIMING

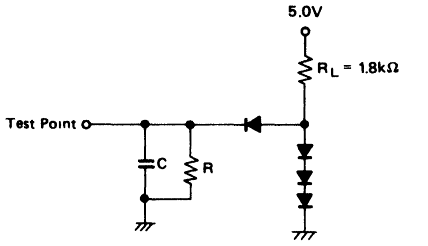
Item	Symbol	Test Condition	HD6809			HD68A09			HD68B09			Unit
			min	typ	max	min	typ	max	min	typ	max	
Frequency of Operation (Crystal or External Input)	f_{XTAL}	Fig. 2, Fig. 3	0.4	–	4	0.4	–	6	0.4	–	8	MHz
Cycle Time	t_{cyc}		1000	–	10000	667	–	10000	500	–	10000	ns
Total Up Time	t_{UT}		975	–	–	640	–	–	480	–	–	ns
Processor Clock "High"	t_{PWEH}		450	–	15500	280	–	15700	220	–	15700	ns
Processor Clock "Low"	t_{PWEL}		430	–	5000	280	–	5000	210	–	5000	ns
E Rise and Fall Time	t_{Er}, t_{Ef}		–	–	25	–	–	25	–	–	20	ns
E_{Low} to Q_{High} Time	t_{AVS}		200	–	250	130	–	165	80	–	125	ns
Q Clock "High"	t_{PWQH}		450	–	5000	280	–	5000	220	–	5000	ns
Q Clock "Low"	t_{PWQL}		450	–	15500	280	–	15700	220	–	15700	ns
Q Rise and Fall Time	t_{Qr}, t_{Qf}		–	–	25	–	–	25	–	–	20	ns
Q_{Low} to E Falling	t_{QE}		200	–	–	133	–	–	100	–	–	ns

2. BUS TIMING

Item	Symbol	Test Condition	HD6809			HD68A09			HD68B09			Unit
			min	typ	max	min	typ	max	min	typ	max	
Address Delay	t_{AD}	Fig. 2, Fig. 3	–	–	200	–	–	140	–	–	110	ns
Address Valid to Q_{High}	t_{AQ}		50	–	–	25	–	–	15	–	–	ns
Peripheral Read Access Time (t_{UT}–t_{AD}–t_{DSR}=t_{ACC})	t_{ACC}		695	–	–	440	–	–	330	–	–	ns
Data Set Up Time (Read)	t_{DSR}		80	–	–	60	–	–	40	–	–	ns
Input Data Hold Time	t_{DHR}		10	–	–	10	–	–	10	–	–	ns
Address Hold Time A_0~A_{15}, $R/\overline{W}$	t_{AH}	Fig. 2, Fig. 3 Ta=0~+75°C	20	–	–	20	–	–	20	–	–	ns
		Fig. 2, Fig. 3 Ta=-20~0°C	10	–	–	10	–	–	10	–	–	ns
Data Delay Time (Write)	t_{DDW}	Fig. 3	–	–	200	–	–	140	–	–	110	ns
Output Hold Time	t_{DHW}	Fig. 3 Ta=0~+75°C	30	–	–	30	–	–	30	–	–	ns
		Fig. 3 Ta=-20~0°C	20	–	–	20	–	–	20	–	–	ns

3. PROCESSOR CONTROL TIMING

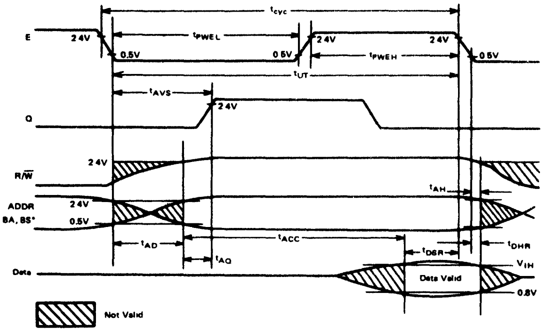
Item	Symbol	Test Condition	HD6809			HD68A09			HD68B09			Unit
			min	typ	max	min	typ	max	min	typ	max	
MRDY Set Up Time	t_{PCSM}	Fig. 6~Fig. 10 Fig. 14, Fig. 15	125	–	–	125	–	–	110	–	–	ns
Interrupts Set Up Time	t_{PCS}		200	–	–	140	–	–	110	–	–	ns
$\overline{HALT}$ Set Up Time	t_{PCSH}		200	–	–	140	–	–	110	–	–	ns
$\overline{RES}$ Set Up Time	t_{PCSR}		200	–	–	140	–	–	110	–	–	ns
$\overline{DMA/BREQ}$ Set Up Time	t_{PCSD}		125	–	–	125	–	–	110	–	–	ns
Processor Control Rise and Fall Time	t_{PCr}, t_{PCf}		–	–	100	–	–	100	–	–	100	ns
Crystal Oscillator Start Time	t_{RC}		–	–	50	–	–	30	–	–	30	ms



- C = 30pF (BA, BS)
  130pF (D_0 ~ D_7, E, Q)
  90pF (A_0 ~ A_{15}, R/$\overline{W}$)
- R = 11kΩ (D_0 ~ D_7)
  16kΩ (A_0 ~ A_{15}, E, Q, R/$\overline{W}$)
  24kΩ (BA, BS)

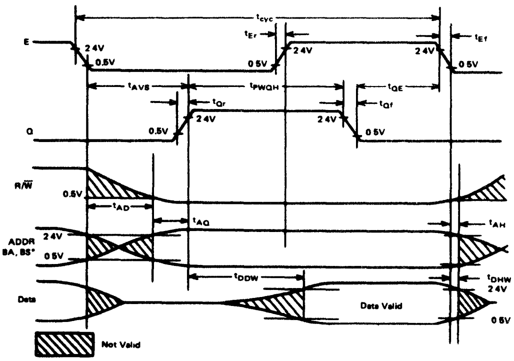
All diodes are 1S2074Ⓗor equivalent.
C includes Stray Capacitance.

Figure 1 Bus Timing Test Load



*Hold time for BA, BS not specified.

Figure 2 Read Data from Memory or Peripherals



*Hold time for BA, BS not specified.

Figure 3 Write Data to Memory or Peripherals

■ PROGRAMMING MODEL

As shown in Figure 4, the HD6809 adds three registers to the set available in the HD6800. The added registers include a Direct Page Register, the User Stack pointer and a second Index Register.

● Accumulators (A, B, D)

The A and B registers are general purpose accumulators which are used for arithmetic calculations and manipulation of data.

Certain instructions concatenate the A and B registers to form a single 16-bit accumulator. This is referred to as the D register, and is formed with the A register as the most significant byte.

● Direct Page Register (DP)

The Direct Page Register of the HD6809 serves to enhance the Direct Addressing Mode. The content of this register appears at the higher address outputs (A_8~A_{15}) during Direct Addressing Instruction execution. This allows the direct mode to be used at any place in memory, under program control. To ensure HD6800 compatibility, all bits of this register are cleared during Processor Reset.

- **Index Registers (X, Y)**

The Index Registers are used in indexed mode of addressing. The 16-bit address in this register takes part in the calculation of effective addresses. This address may be used to point to data directly or may be modified by an optional constant or register offset. During some indexed modes, the contents of the index register are incremented or decremented to point to the next item of tabular type data. All four pointer registers (X, Y, U, S) may be used as index registers.

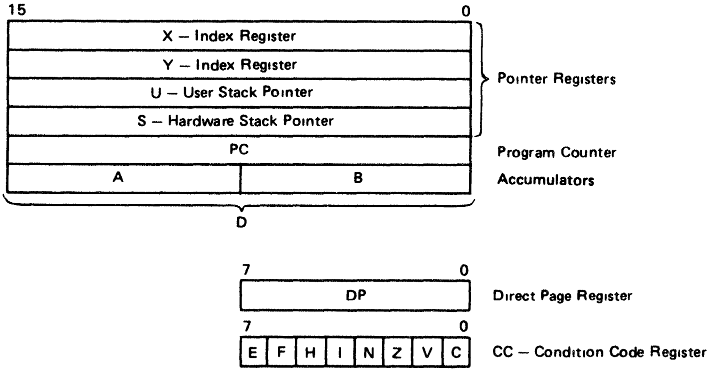


Figure 4 Programming Model of The Microprocessing Unit

- **Stack Pointer (U, S)**

The Hardware Stack Pointer (S) is used automatically by the processor during subroutine calls and interrupts. The stack pointers of the HD6809 point to the top of the stack, in contrast to the HD6800 stack pointer, which pointed to the next free location on the stack. The User Stack Pointer (U) is controlled exclusively by the programmer thus allowing arguments to be passed to and from subroutines with ease. Both Stack Pointers have the same indexed mode addressing capabilities as the X and Y registers, but also support Push and Pull instructions. This allows the HD6809 to be used efficiently as a stack processor, greatly enhancing its ability to support higher level languages and modular programming.

- **Program Counter**

The Program Counter is used by the processor to point to the address of the next instruction to be executed by the processor. Relative Addressing is provided allowing the Program Counter to be used like an index register in some situations.

- **Condition Code Register**

The Condition Code Register defines the State of the Processor at any given time. See Fig. 5.

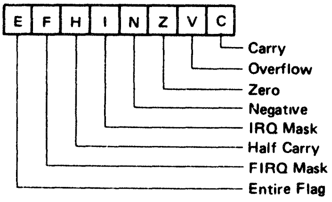


Figure 5 Condition Code Register Format

■ CONDITION CODE REGISTER DESCRIPTION

- **Bit 0 (C)**

Bit 0 is the carry flag, and is usually the carry from the binary ALU. C is also used to represent a 'borrow' from subtract like instructions (CMP, NEG, SUB, SBC) and is the complement of the carry from the binary ALU.

- **Bit 1 (V)**

Bit 1 is the overflow flag, and is set to a one by an operation which causes a signed two's complement arithmetic overflow. This overflow is detected in an operation in which the carry from the MSB in the ALU does not match the carry from the MSB-1.

- **Bit 2 (Z)**

Bit 2 is the zero flag, and is set to a one if the result of the previous operation was identically zero.

- **Bit 3 (N)**

Bit 3 is the negative flag, which contains exactly the value of the MSB of the result of the preceding operation. Thus, a negative two's-complement result will leave N set to a one.

- **Bit 4 (I)**

Bit 4 is the $\overline{IRQ}$ mask bit. The processor will not recognize interrupts from the $\overline{IRQ}$ line if this bit is set to a one. $\overline{NMI}$, $\overline{FIRQ}$, $\overline{IRQ}$, $\overline{RES}$, and SWI all are set I to a one; SWI2 and SWI3 do not affect I.

- **Bit 5 (H)**

Bit 5 is the half-carry bit, and is used to indicate a carry·from bit 3 in the ALU as a result of an 8-bit addition only (ADC or ADD). This bit is used by the DAA instruction to perform a BCD decimal add adjust operation. The state of this flag is undefined in all subtract-like instructions.

- **Bit 6 (F)**

Bit 6 is the $\overline{\text{FIRQ}}$ mask bit. The processor will not recognize interrupts from the $\overline{\text{FIRQ}}$ line if this bit is a one. $\overline{\text{NMI}}$, $\overline{\text{FIRQ}}$, SWI, and $\overline{\text{RES}}$ all set F to a one. $\overline{\text{IRQ}}$, SWI2 and SWI3 do not affect F.

- **Bit 7 (E)**

Bit 7 is the entire flag, and when set to a one indicates that the complete machine state (all the registers) was stacked, as opposed to the subset state (PC and CC). The E bit of the stacked CC is used on a return from interrupt (RTI) to determine the extent of the unstacking. Therefore, the current E left in the Condition Code Register represents past action.

■ SIGNAL DESCRIPTION

- **Power (V_{SS}, V_{CC})**

Two pins are used tŏ supply power to the part: V_{SS} is ground or 0 volts, while V_{CC} is +5.0V ±5%.

- **Address Bus (A_0~A_{15})**

Sixteen pins are used to output address information from the MPU onto the Address Bus. When the processor does not require the bus for a data transfer, it will output address $FFFF_{16}$, R/$\overline{\text{W}}$ = "High", and BS = "Low"; this is a "dummy access" or $\overline{\text{VMA}}$ cycle. Addresses are valid on the rising edge of Q (see Figs. 2 and 3). All address bus drivers are made high impedance when output Bus Availalbe (BA) is "High". Each pin will drive one Schottky TTL load or four LS TTL loads, and typically 90 pF.

- **Data Bus (D_0~D_7)**

These eight pins provide communication with the system bi-directional data bus. Each pin will drive one Schottky TTL load or four LS TTL loads, and typically 130 pF.

- **Read/Write (R/$\overline{\text{W}}$)**

This signal indicates the direction of data transfer on the data bus. A "Low" indicates that the MPU is writing data onto the data bus. R/$\overline{\text{W}}$ is made high impedance when BA is "High". R/$\overline{\text{W}}$ is valid on the rising edge of Q. Refer to Figs. 2 and 3.

- **Reset ($\overline{\text{RES}}$)**

A "Low" level on this Schmitt-trigger input for greater than one bus cycle will reset the MPU, as shown in Fig. 6. The Reset vectors are fetched from locations $FFFE_{16}$ and $FFFF_{16}$ (Table 1) when Interrupt Acknowledge is true, ($\overline{\text{BA}}$ • BS=1). During initial power-on, the Reset line should be held "Low" until the clock oscillator is fully operational. See Fig. 7.

Because the HD6809 Reset pin has a Schmitt-trigger input with a threshold voltage higher than that of standard peripherals, a simple R/C network may be used to reset the entire system. This higher threshold voltage ensures that all peripherals are out of the reset state before the Processor.

Table 1 Memory Map for Interrupt Vectors

Memory Map For Vector Locations		Interrupt Vector Description
MS	LS	
FFFE	FFFF	$\overline{\text{RES}}$
FFFC	FFFD	$\overline{\text{NMI}}$
FFFA	FFFB	SWI
FFF8	FFF9	$\overline{\text{IRQ}}$
FFF6	FFF7	$\overline{\text{FIRQ}}$
FFF4	FFF5	SWI2
FFF2	FFF3	SWI3
FFF0	FFF1	Reserved

- **$\overline{\text{HALT}}$**

A "Low" level on this input pin will cause the MPU to stop running at the end of the present instruction and remain halted indefinitely without loss of data. When halted, the BA output is driven "High" indicating the buses are high impedance. BS is also "High" which indicates the processor is in the Halt or Bus Grant state. While halted, the MPU will not respond to external real-time requests ($\overline{\text{FIRQ}}$, $\overline{\text{IRQ}}$) although $\overline{\text{DMA/BREQ}}$ will always be accepted, and $\overline{\text{NMI}}$ or $\overline{\text{RES}}$ will be latched for later response. During the Halt state Q and E continue to run normally. If the MPU is not running ($\overline{\text{RES}}$, $\overline{\text{DMA/BREQ}}$), a halted state (BA•BS=1) can be achieved by pulling $\overline{\text{HALT}}$ "Low" while $\overline{\text{RES}}$ is still "Low". If $\overline{\text{DAM/BREQ}}$ and $\overline{\text{HALT}}$ are both pulled "Low", the processor will reach the last cycle of the instruction (by reverse cycle stealing) where the machine will then become halted. See Figs. 8 and 16.

- **Bus Available, Bus Status (BA, BS)**

The BA output is an indication of an internal control signal which makes the MOS buses of the MPU high impedance. This signal does not imply that the bus will be available for more than one cycle. When BA goes "Low", an additional dead cycle will elapse before the MPU acquires the bus.

The BS output signal, when decoded with BA, represents the MPU state (valid with leading edge of Q).

Table 2 MPU State Definition

BA	BS	MPU State
0	0	Normal (Running)
0	1	Interrupt or RESET Acknowledge
1	0	SYNC Acknowledge
1	1	HALT or Bus Grant

Interrupt Acknowledge is indicated during both cycles of a hardware-vector-fetch ($\overline{\text{RES}}$, $\overline{\text{NMI}}$, $\overline{\text{FIRQ}}$, $\overline{\text{IRQ}}$, SWI, SWI2, SWI3). This signal, plus decoding of the lower four address lines, can provide the user with an indication of which interrupt level is being serviced and allow vectoring by device. See Table 1.

Sync Acknowledge is indicated while the MPU is waiting for external synchronization on an interrupt line.

Halt/Bus Grant is true when the HD6809 is in a Halt or Bus Grant condition.

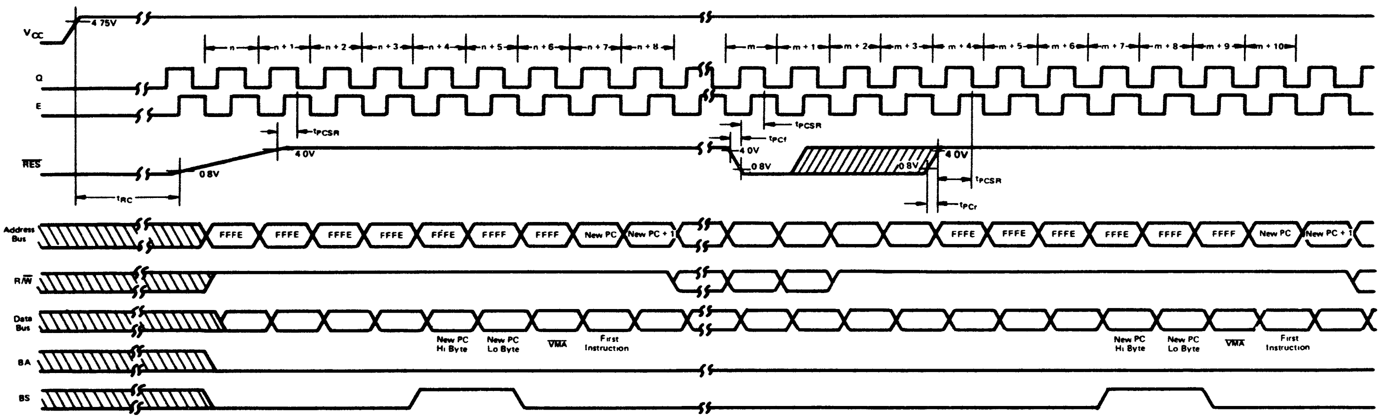
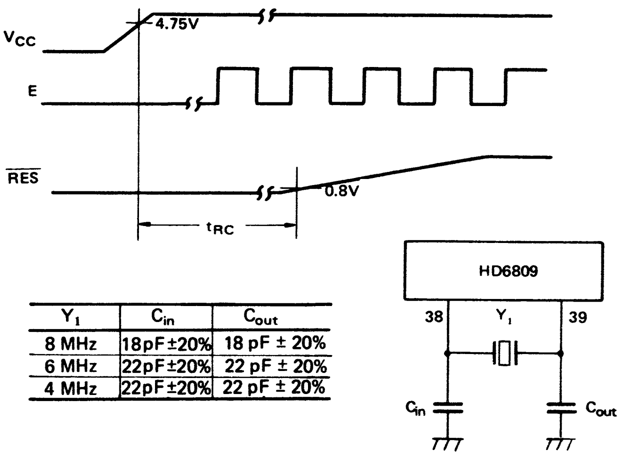


Figure 6 RES Timing



Y_1	C_{in}	C_{out}
8 MHz	18pF±20%	18 pF ± 20%
6 MHz	22pF±20%	22 pF ± 20%
4 MHz	22pF±20%	22 pF ± 20%

Figure 7 Crystal Connections and Oscillator Start Up

- **Non Maskable Interrupt ($\overline{\text{NMI}}$)***

A negative edge on this input requests that a non-maskable interrupt sequence be generated. A non-maskable interrupt cannot be inhibited by the program, and also has a higher priority than $\overline{\text{FIRQ}}$, $\overline{\text{IRQ}}$ or software interrupts. During recognition of an $\overline{\text{NMI}}$, the entire machine state is saved on the hardware stack. After reset, an $\overline{\text{NMI}}$ will not be recognized until the first program load of the Hardware Stack Pointer (S). The pulse width of $\overline{\text{NMI}}$ "Low" must be at least one E cycle. If the $\overline{\text{NMI}}$ input does not meet the minimum set up with respect to Q, the interrupt will not be recognized until the next cycle. See Fig. 9.

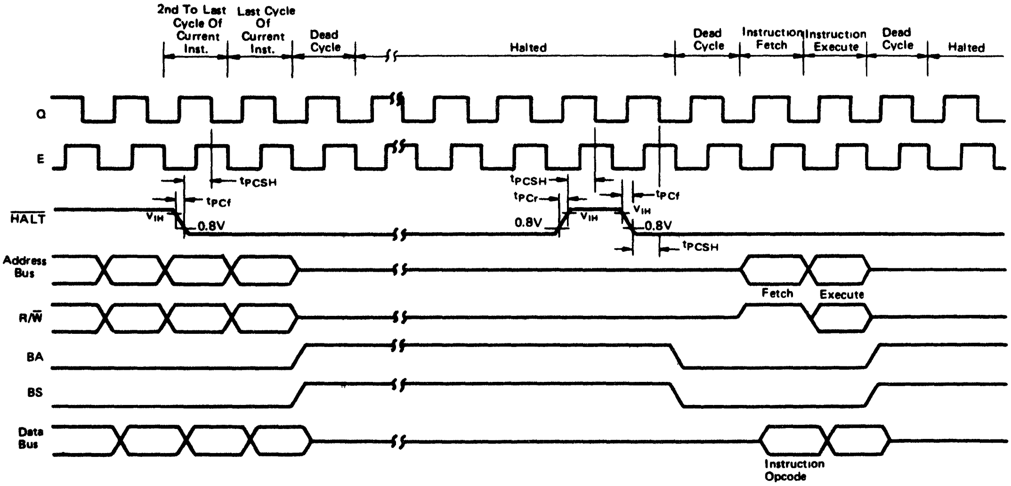


Figure 8 $\overline{\text{HALT}}$ and Single Instruction Execution for System Debug

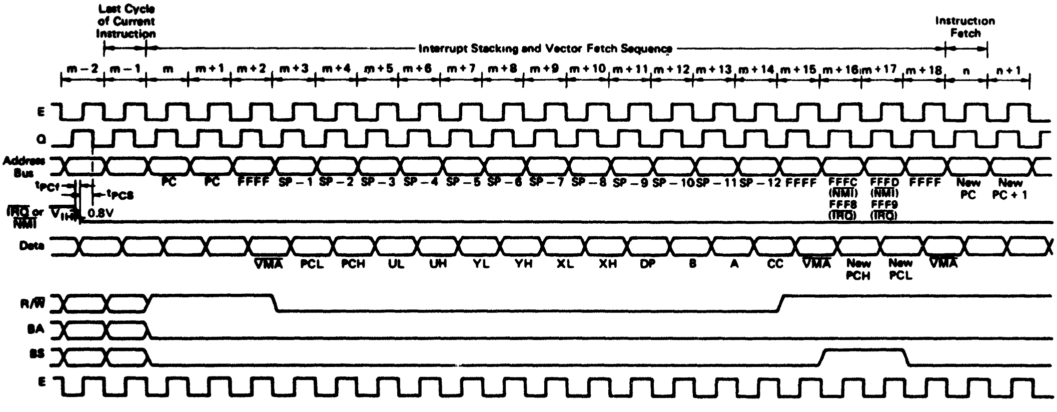


Figure 9 $\overline{\text{IRQ}}$ and $\overline{\text{NMI}}$ Interrupt Timing

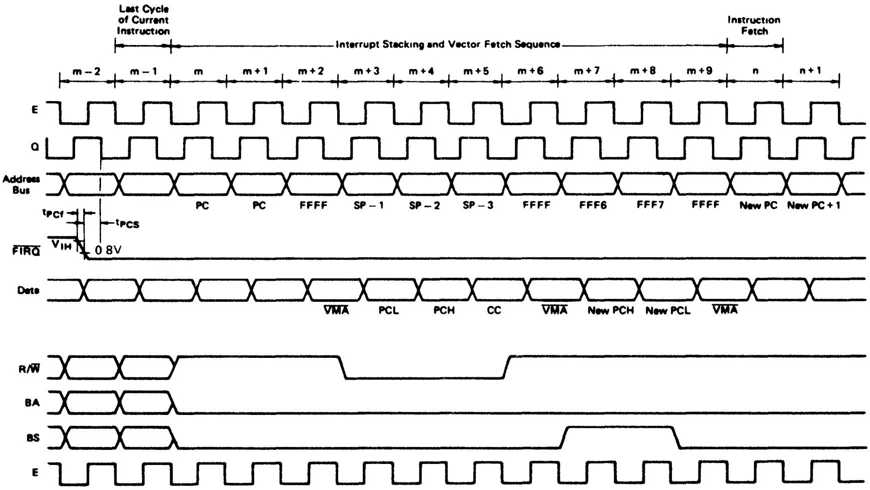


Figure 10 FIRQ Interrupt Timing

- **Fast-Interrupt Request (FIRQ)***

A "Low" level on this input pin will initiate a fast interrupt sequence provided its mask bit (F) in the CC is clear. This sequence has priority over the standard Interrupt Request (IRQ), and is fast in the sense that it stacks only the contents of the condition code register and the program counter. The interrupt service routine should clear the source of the interrupt before doing an RTI. See Fig. 10.

- **Interrupt Request (IRQ)***

A "Low" level input on this pin will initiate an interrupt Request sequence provided the mask bit (I) in the CC is clear. Since IRQ stacks the entire machine state it provides a slower response to interrupts than FIRQ. IRQ also has a lower priority than FIRQ. Again, the interrupt service routine should clear the source of the interrupt before doing an RTI. See Fig. 9.

* NMI, FIRQ, and IRQ requests are sampled on the falling edge of Q. One cycle is required for synchronization before these interrupts are recognized. The pending interrupt(s) will not be serviced until completion of the current instruction unless a SYNC or CWAI condition is present. If IRQ and FIRQ do not remain "Low" until completion of the current instruction they may not be recognized. However, NMI is latched and need only remain "Low" for one cycle.

- **XTAL, EXTAL**

These inputs are used to connect the on-chip oscillator to an external parallel-resonant crystal. Alternately, the pin EXTAL may be used as a TTL level input for external timing by grounding XTAL. The crystal or external frequency is four times the bus frequency. See Fig. 7. Proper RF layout techniques should be observed in the layout of printed circuit boards.

< NOTE FOR BOARD DESIGN OF THE OSCILLATION CIRCUIT >

In designing the board, the following notes should be taken when the crystal oscillator is used.

1) Crystal oscillator and load capacity Cin, Cout must be placed near the LSI as much as possible.

[Normal oscillation may be disturbed when external noise is induced to pin 38 and 39.]

2) Pin 38 and 39 signal line should be wired apart from other signal line as much as possible. Don't wire them in parallel.

[Normal oscillation may be disturbed when E or Q signal is feedbacked to pin 38 and 39.]

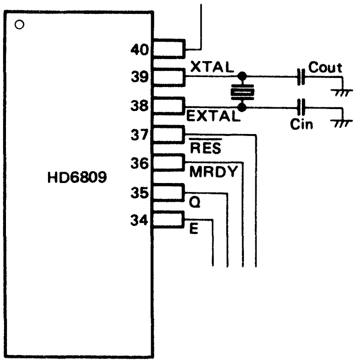


Figure 11 Board Design of the Oscillation Circuit.

<THE FOLLOWING DESIGN MUST BE AVOIDED>

A signal line or a power source line must not cross or go near the oscillation circuit line as shown in Fig. 12 to prevent the induction from these lines and perform the correct oscillation. The resistance among XTAL, EXTAL and other pins should be over 10MΩ.

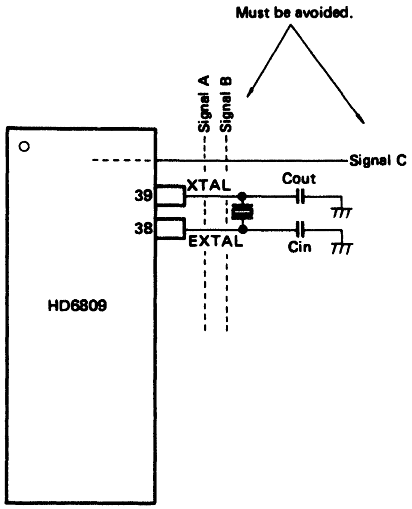


Figure 12 Example of Normal Oscillation may be Disturbed.

- **E, Q**

E is similar to the HD6800 bus timing signal ϕ_2; Q is a quadrature clock signal which leads E. Q has no parallel on the HD6800. Addresses from the MPU will be valid with the leading edge of Q. Data is latched on the falling edge of E. Timing for E and Q is shown in Fig. 13.

- **MRDY**

This input control signal allows stretching of E and Q to extend data-access time. E and Q operate normally while MRDY is "High". When MRDY is "Low", E and Q may be stretched in integral multiples of quarter (1/4) bus cycles, thus allowing interface to slow memories, as shown in Fig. 14. A maximum

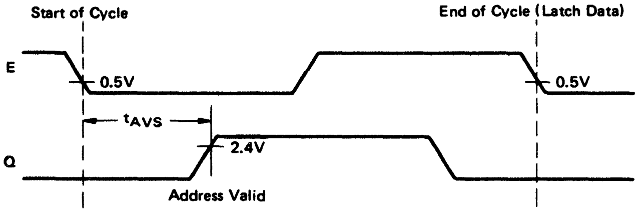


Figure 13 E/Q Relationship

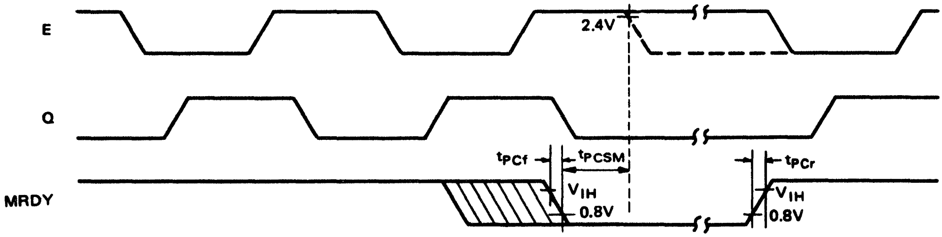


Figure 14 MRDY Timing

stretch is 10 microseconds. During nonvalid memory access ($\overline{\text{VMA}}$ cycles) MRDY has no effect on stretching E and Q; this inhibits slowing the processor during "don't care" bus accesses. MRDY may also be used to stretch clocks (for slow memory) when bus control has been transferred to an external device (through the use of $\overline{\text{HALT}}$ and $\overline{\text{DMA/BREQ}}$).

Also MRDY has effect on stretching E and Q during Dead Cycle.

• $\overline{\text{DMA/BREQ}}$

The $\overline{\text{DMA/BREQ}}$ input provides a method of suspending execution and acquiring the MPU bus for another use, as shown in Fig. 15. Typical uses include DMA and dynamic memory refresh.

Transition of $\overline{\text{DMA/BREQ}}$ should occur during Q. A "Low" level on this pin will stop instruction execution at the end of the current cycle. The MPU will acknowledge $\overline{\text{DMA/BREQ}}$ by setting BA and BS to "High" level. The requesting device will now have up to 15 bus cycles before the MPU retrieves the bus for self-refresh. Self-refresh requires one bus cycle with a leading and trailing dead cycle. See Fig. 16.

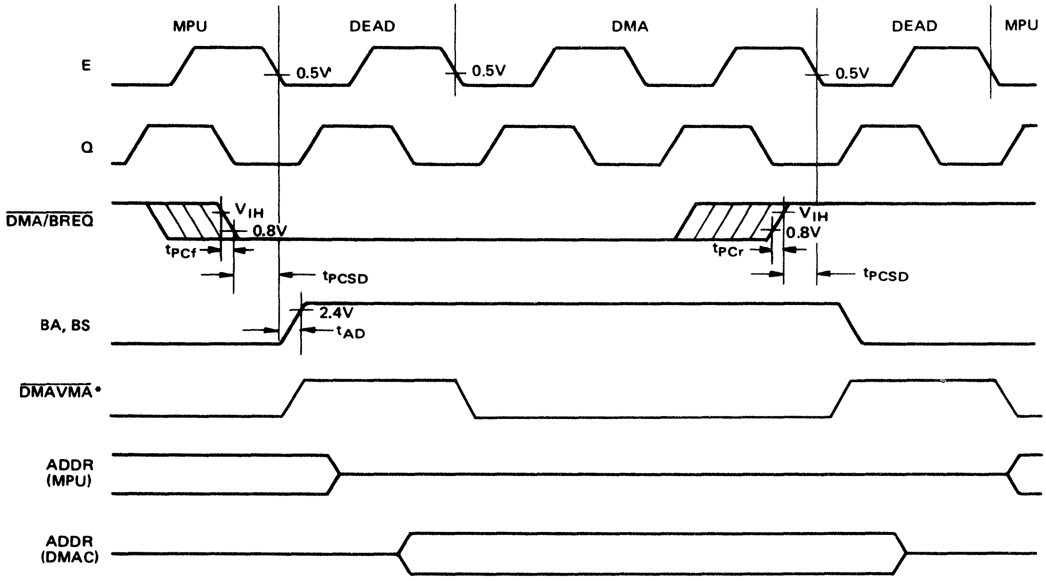
Typically, the DMA controller will request to use the bus by asserting $\overline{\text{DMA/BREQ}}$ pin "Low" on the leading edge of E. When the MPU replies by setting BA and BS to a one, that cycle will be a dead cycle used to transfer bus mastership to the DMA controller.

False memory accesses may be prevented during and dead cycles by developing a system $\overline{\text{DMAVMA}}$ signal which is "Low" in any cycle when BA has changed.

When BA goes "Low" (either as a result of $\overline{\text{DMA/BREQ}}$ = "High" or MPU self-refresh), the DMA device should be taken off the bus. Another dead cycle will elapse before the MPU accesses memory, to allow transfer of bus mastership without contention.

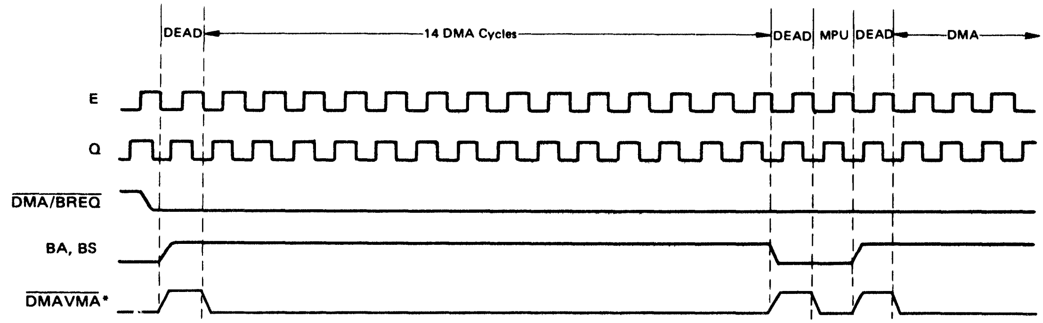
▪ MPU OPERATION

During normal operation, the MPU fetches an instruction from memory and then executes the requested function. This



*$\overline{\text{DMAVMA}}$ is a signal which is developed externally, but is a system requirement for DMA.

Figure 15 Typical DMA Timing (<14 Cycles)



*$\overline{\text{DMAVMA}}$ is a signal which is developed externally, but is a system requirement for DMA.

Figure 16 Auto – Refresh DMA Timing
(Reverse Cycle Stealing)

sequence begins at $\overline{\text{RES}}$ and is repeated indefinitely unless altered by a special instruction or hardware occurrence. Software instructions that alter normal MPU operation are: SWI, SWI2, SWI3, CWAI, RTI and SYNC. An interrupt, $\overline{\text{HALT}}$ or $\overline{\text{DMA/BREQ}}$ can also alter the normal execution of instructions. Fig. 17 illustrates the flow chart for the HD6809.

■ ADDRESSING MODES

The basic instructions of any computer are greatly enhanced by the presence of powerful addressing modes. The HD6809 has the most complete set of addressing modes available on any microcomputer today. For example, the HD6809 has 59 basic instructions; however, it recognizes 1464 different variations of instructions and addressing modes. The addressing modes support modern programming techniques. The following addressing modes are available on the HD6809:

(1) Implied (Includes Accumulator)
(2) Immediate
(3) Extended
(4) Extended Indirect
(5) Direct
(6) Register
(7) Indexed
  Zero-Offset
  Constant Offset
  Accumulator Offset
  Auto Increment/Decrement
(8) Indexed Indirect
(9) Relative
(10) Program Counter Relative

● Implied (Includes Accumulator)

In this addressing mode, the opcode of the instruction contains all the address information necessary. Examples of Implied Addressing are: ABX, DAA, SWI, ASRA, and CLRB.

● Immediate Addressing

In Immediate Addressing, the effective address of the data is the location immediately following the opcode (i.e., the data to be used in the instruction immediately follows the opcode of the instruction). The HD6809 uses both 8 and 16-bit immediate values depending on the size of argument specified by the opcode. Examples of instructions with Immediate Addressing are:

```
LDA #$20
LDX #$F000
LDY #CAT
```

(NOTE) # signifies Immediate addressing, $ signifies hexadecimal value.

● Extended Addressing

In Extended Addressing, the contents of the two bytes immediately following the opcode fully specify the 16-bit effective address used by the instruction. Note that the address generated by an extended instruction defines an absolute address and is not position independent. Examples of Extended Addressing include:

```
LDA   CAT
STX   MOUSE
LDD   $2000
```

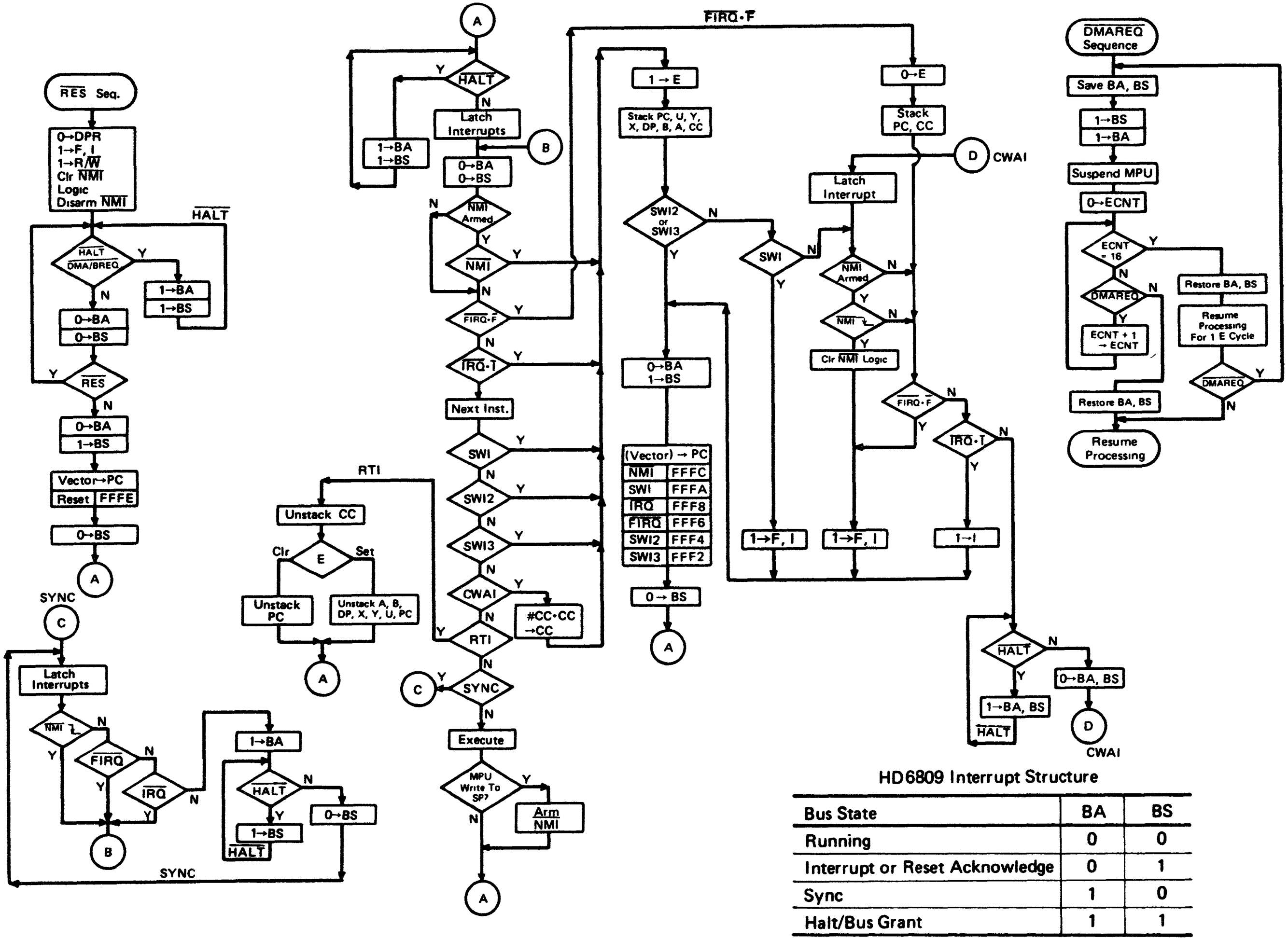
● Extended Indirect

As a special case of indexed addressing (discussed below), "1" level of indirection may be added to Extended Addressing. In Extended Indirect, the two bytes following the postbyte of an Indexed instruction contain the address of the data.

```
LDA   [CAT]
LDX   [$FFFE]
STU   [DOG]
```

● Direct Addressing

Direct addressing is similar to extended addressing except that only one byte of address follows the opcode. This byte specifies the lower 8-bit of the address to be used. The upper 8-bit of the address are supplied by the direct page register. Since only one byte of address is required in direct addressing, this mode requires less memory and executes faster than extended addressing. Of course, only 256 locations (one page) can be



HD6809 Interrupt Structure

Bus State	BA	BS
Running	0	0
Interrupt or Reset Acknowledge	0	1
Sync	1	0
Halt/Bus Grant	1	1

(NOTE) Asserting RES will result in entering the reset sequence from any point in the flow chart.

Figure 17 Flowchart for HD6809 Instruction

accessed without redefining the contents of the DP register. Since the DP register is set to $00 on Reset, direct addressing on the HD6809 is compatible with direct addressing on the HD6800. Indirection is not allowed in direct addressing. Some examples of direct addressing are:

LDA	$30
SETDP	$10 (Assembler directive)
LDB	$1030
LDD	<CAT

(NOTE) < is an assembler directive which forces direct addressing.

- **Register Addressing**

Some opcodes are followed by a byte that defines a register or set of registers to be used by the instruction. This is called a postbyte. Some examples of register addressing are:

TFR	X, Y	Transfers X into Y
EXG	A, B	Exchanges A with B
PSHS	A, B, X, Y	Push Y, X, B and A onto S
PULU	X, Y, D	Pull D, X, and Y from U

- **Indexed Addressing**

In all indexed addressing, one of the pointer registers (X, Y, U, S, and sometimes PC) is used in a calculation of the effective address of the operand to be used by the instruction. Five basic types of indexing are available and are discussed below. The postbyte of an indexed instruction specifies the basic type and variation of the addressing mode as well as the pointer register to be used. Fig. 18 lists the legal formats for the postbyte. Table 3 gives the assembler form and the number of cycles and bytes

Post-Byte Register Bit								Indexed Addressing Mode
7	6	5	4	3	2	1	0	
0	R	R	×	×	×	×	×	EA = ,R + 5 Bit Offset
1	R	R	0	0	0	0	0	,R +
1	R	R	0/1	0	0	0	1	,R + +
1	R	R	0	0	0	1	0	, -R
1	R	R	0/1	0	0	1	1	, - - R
1	R	R	0/1	0	1	0	0	EA = ,R + 0 Offset
1	R	R	0/1	0	1	0	1	EA = ,R + ACCB Offset
1	R	R	0/1	0	1	1	0	EA = ,R + ACCA Offset
1	R	R	0/1	1	0	0	0	EA = , R + 8 Bit Offset
1	R	R	0/1	1	0	0	1	EA = ,R + 16 Bit Offset
1	R	R	0/1	1	0	1	1	EA = ,R + D Offset
1	×	×	0/1	1	1	0	0	EA= ,PC + 8 Bit Offset
1	×	×	0/1	1	1	0	1	EA = ,PC + 16 Bit Offset
1	R	R	1	1	1	1	1	EA = [,Address]

Addressing Mode Field

Indirect Field
(Sigh bit when b7 = 0)
{0 .... Non Indirect
{1 .... Indirect

Register Field : RR
00 = X
01 = Y
10 = U
11 = S

× = Don't Care

Figure 18 Index Addressing Postbyte Register Bit Assignments

Table 3 Indexed Addressing Mode

Type	Forms	Non Indirect				Indirect			
		Assembler Form	Postbyte OP Code	+ ~	+ #	Assembler Form	Postbyte OP Code	+ ~	+ #
Constant Offset From R (2's Complement Offsets)	No Offset	,R	1RR00100	0	0	[,R]	1RR10100	3	0
	5 Bit Offset	n, R	0RRnnnnn	1	0	defaults to 8-bit			
	8 Bit Offset	n, R	1RR01000	1	1	[n, R]	1RR11000	4	1
	16 Bit Offset	n, R	1RR01001	4	2	[n, R]	1RR11001	7	2
Accumulator Offset From R (2's Complement Offsets)	A Register Offset	A, R	1RR00110	1	0	[A, R]	1RR10110	4	0
	B Register Offset	B, R	1RR00101	1	0	[B, R]	1RR10101	4	0
	D Register Offset	D, R	1RR01011	4	0	[D, R]	1RR11011	7	0
Auto Increment/Decrement R	Increment By 1	,R +	1RR00000	2	0	not allowed			
	Increment By 2	,R + +	1RR00001	3	0	[,R + +]	1RR10001	6	0
	Decrement By 1	, - R	1RR00010	2	0	not allowed			
	Decrement By 2	, - - R	1RR00011	3	0	[, - - R]	1RR10011	6	0
Constant Offset From PC (2's Complement Offsets)	8 Bit Offset	n, PCR	1xx01100	1	1	[n,PCR]	1xx11100	4	1
	16 Bit Offset	n, PCR	1xx01101	5	2	[n, PCR]	1xx11101	8	2
Extended Indirect	16 Bit Address	–	–	–	–	[n]	10011111	5	2

R = X, Y, U or S
× = Don't Care

RR:
00 = X
01 = Y
10 = U
11 = S

$\overset{+}{\sim}$ and $\overset{+}{\#}$ indicate the number of additional cycles and bytes for the particular variation.

added to the basic values for indexed addressing for each variation.

Zero-Offset Indexed

In this mode, the selected pointer register contains the effective address of the data to be used by the instruction. This is the fastest indexing mode.

Examples are:

```
LDD    0,X
LDA    S
```

Constant Offset Indexed

In this mode, a two's-complement offset and the contents of one of the pointer registers are added to form the effective address of the operand. The pointer register's initial content is unchanged by the addition.

Three sizes of offsets are available:

5-bit (−16 to +15)
8-bit (−128 to +127)
16-bit (−32768 to +32767)

The two's complement 5-bit offset is included in the postbyte and, therefore, is most efficient in use of bytes and cycles. The two's complement 8-bit offset is contained in a single byte following the postbyte. The two's complement 16-bit offset is in the two bytes following the postbyte. In most cases the programmer need not be concerned with the size of this offset since the assembler will select the optimal size automatically.

Examples of constant-offset indexing are:

```
LDA    23,X
LDX    -2,S
LDY    300,X
LDU    CAT,Y
```

Accumulator-Offset Indexed

This mode is similar to constant offset indexed except that the two's-complement value in one of the accumulators (A, B or D) and the contents of one of the pointer registers are added to form the effective address of the operand. The contents of both the accumulator and the pointer register are unchanged by the addition. The postbyte specifies which accumulator to use as an offset and no additional bytes are required. The advantage of an accumulator offset is that the value of the offset can be calculated by a program at run-time.

Some examples are:

```
LDA    B,Y
LDX    D,Y
LEAX   B,X
```

Auto Increment/Decrement Indexed

In the auto increment addressing mode, the pointer register contains the address of the operand. Then, after the pointer register is used it is incremented by one or two. This addressing mode is useful in stepping through tables, moving data, or for the creation of software stacks. In auto decrement, the pointer register is decremented prior to use as the address of the data. The use of auto decrement is similar to that of auto increment; but the tables, etc., are scanned from the "High" to "Low" addresses.· The size of the increment/decrement can be either one or two to allow for tables of either 8 or 16-bit data to be accessed and is selectable by the programmer. The pre-decrement, post-increment nature of these modes allow them to be used to create additional software stacks that behave identically to the U and S stacks.

Some examples of the auto increment/decrement addressing modes are:

```
LDA    ,X+
STD    ,Y++
LDB    ,-Y
LDX    ,--S
```

Care should be taken in performing operations on 16-bit pointer registers (X, Y, U, S) where the same register is used to calculate the effective address.

Consider the following instruction:

STX 0, X + + (X initialized to 0)

The desired result is to store a 0 in locations $0000 and $0001 then increment X to point to $0002. In reality, the following occurs:

```
0 → temp        calculate the EA; temp is a holding register
X + 2 → X       perform autoincrement
X → (temp)      do store operation
```

• Indexed Indirect

All of the indexing modes with the exception of auto increment/decrement by one, or a ±4-bit offset may have an additional level of indirection specified. In indirect addressing, the effective address is contained at the location specified by the contents of the Index register plus any offset. In the example below, the A accumulator is loaded indirectly using an effective address calculated from the Index register and an offset.

```
        Before Execution
        A = XX (don't care)
        X = $F000
$0100   LDA [$10,X]          EA is now $F010

$F010   $F1                  $F150 is now the
$F011   $50                  new EA

$F150   $AA
        After Execution
        A = $AA Actual Data Loaded
        X = $F000
```

All modes of indexed indirect are included except those which are meaningless (e.g., auto increment/decrement by 1 indirect). Some examples of indexed indirect are:

```
LDA    [,X]
LDD    [10,S]
LDA    [B,Y]
LDD    [,X++]
```

• Relative Addressing

The byte(s) following the branch opcode is (are) treated as a signed offset which may be added to the program counter. If the branch condition is true then the calculated address (PC + signed offset) is loaded into the program counter. Program execution continues at the new location as indicated by the PC; short (1 byte offset) and long (2 bytes offset) relative addressing modes are available. All of memory can be reached in long relative addressing as an effective address is interpreted modulo 2^{16}. Some examples of relative addressing are:

```
         BEQ     CAT      (short)
         BGT     DOG      (short)
CAT      LBEQ    RAT      (long)
DOG      LBGT    RABBIT   (long)
```

```
           •
           •
           •
RAT        NOP
RABBIT     NOP
```

- **Program Counter Relative**

The PC can be used as the pointer register with 8 or 16-bit signed offsets. As in relative addressing, the offset is added to the current PC to create the effective address. The effective address is then used as the address of the operand or data. Program Counter Relative Addressing is used for writing position independent programs. Tables related to a particular routine will maintain the same relationship after the routine is moved, if referenced relative to the Program Counter. Examples are:

```
LDA    CAT, PCR
LEAX   TABLE, PCR
```

Since program counter relative is a type of indexing, an additional level of indirection is available.

```
LDA    [CAT, PCR]
LDU    [DOG, PCR]
```

■ HD6809 INSTRUCTION SET

The instruction set of the HD6809 is similar to that of the HD6800 and is upward compatible at the source code level. The number of opcodes has been reduced from 72 to 59, but because of the expanded architecture and additional addressing modes, the number of available opcodes (with different addressing modes) has risen from 197 to 1464.

Some of the new instructions and addressing modes are described in detail below:

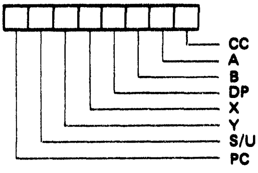
- **PSHU/PSHS**

The push instructions have the capability of pushing onto either the hardware stack (S) or user stack (U) any single register, or set of registers with a single instruction.

- **PULU/PULS**

The pull instructions have the same capability of the push instruction, in reverse order. The byte immediately following the push or pull opcode determines which register or registers are to be pushed or pulled. The actual PUSH/PULL sequence is fixed; each bit defines a unique register to push or pull, as shown in below.

PUSH/PULL POST BYTE



```
    ← Pull Order          Push Order →
PC    U    Y    X    DP    B    A    CC
FFFF...← increasing memory address .....0000
PC    S    Y    X    DP    B    A    CC
```

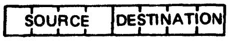
- **TFR/EXG**

Within the HD6809, any register may be transferred to or exchanged with another of like-size; i.e., 8-bit to 8-bit or 16-bit to 16-bit. Bits 4-7 of postbyte define the source register, while bits 0-3 represent the destination register. Three are denoted as follows:

0000	D	0101	PC
0001	X	1000	A
0010	Y	1001	B
0011	U	1010	CC
0100	S	1011	DP

(NOTE) All other combinations are undefined and INVALID.

TRANSFER/EXCHANGE POST BYTE



- **LEAX/LEAY/LEAU/LEAS**

The LEA (Load Effective Address) works by calculating the effective address used in an indexed instruction and stores that address value, rather than the data at that address, in a pointer register. This makes all the features of the internal addressing hardware available to the programmer. Some of the implications of this instruction are illustrated in Table 4.

The LEA instruction also allows the user to access data in a position independent manner. For example:

```
        LEAX      MSG1, PCR
        LBSR      PDATA (Print message routine)
          •
          •
MSG1 FCC          'MESSAGE'
```

This sample program prints: 'MESSAGE'. By writing MSG1, PCR, the assembler computes the distance between the present address and MSG1. This result is placed as a constant into the LEAX instruction which will be indexed from the PC value at the time of execution. No matter where the code is located, when it is executed, the computed offset from the PC will put the absolute address of MSG1 into the X pointer register. This code is totally position independent.

The LEA instructions are very powerful and use an internal holding register (temp). Care must be exercised when using the LEA instructions with the autoincrement and autodecrement addressing modes due to the sequence of internal operations. The LEA internal sequence is outlined as follows:

LEAa, b+ (any of the 16-bit pointer registers X, Y, U or S may be substituted for a and b.)
1. b → temp (calculate the EA)
2. b + 1 → b (modify b, postincrement)
3. temp → a (load a)

LEAa, – b
1. b – 1 → temp (calculate EA with predecrement)
2. b – 1 → b (modify b, predecrement)
3. temp → a (load a)

Autoincrement-by-two and autodecrement-by-two instructions work similarly. Note that LEAX, X+ does not change X, however LEAX, –X does decrement X. LEAX 1, X should be used to increment X by one.

Table 4 LEA Examples

Instruction	Operation	Comment
LEAX 10, X	X + 10 → X	Adds 5-bit constant 10 to X
LEAX 500, X	X + 500 → X	Adds 16-bit constant 500 to X
LEAY A, Y	Y + A → Y	Adds 8-bit accumulator to Y
LEAY D, Y	Y + D → Y	Adds 16-bit D accumulator to Y
LEAU –10, U	U – 10 → U	Subtracts 10 from U
LEAS –10, S	S – 10 → S	Used to reserve area on stack
LEAS 10, S	S + 10 → S	Used to 'clean up' stack
LEAX 5, S	S + 5 → X	Transfers as well as adds

• MUL

Multiplies the unsigned binary numbers in the A and B accumulator and places the unsigned result into the 16-bit D accumulator. This unsigned multiply also allows multiple-precision multiplications.

Long And Short Relative Branches

The HD6809 has the capability of program counter relative branching throughout the entire memory map. In this mode, if the branch is to be taken, the 8 or 16-bit signed offset is added to the value of the program counter to be used as the effective address. This allows the program to branch anywhere in the 64k memory map. Position independent code can be easily generated through the use of relative branching. Both short (8-bit) and long (16-bit) branches are available.

• SYNC

After encountering a Sync instruction, the MPU enters a Sync state, stops processing instructions and waits for an interrupt. If the pending interrupt is non-maskable ($\overline{\text{NMI}}$) or maskable ($\overline{\text{FIRQ}}$, $\overline{\text{IRQ}}$) with its mask bit (F or I) clear, the processor will clear the Syne state and perform the normal interrupt stacking and service routine. Since $\overline{\text{FIRQ}}$ and $\overline{\text{IRQ}}$ are not edge-triggered, a "Low" level with a minimum duration of three bus cycles is required to assure that the interrupt will be taken. If the pending interrupt is maskable ($\overline{\text{FIRQ}}$, $\overline{\text{IRQ}}$) with its mask bit (F or I) set, the processor will clear the Sync state and continue processing by executing the next inline instruction. Fig. 19 depicts Sync timing.

Software Interrupts

A Software Interrupt is an instruction which will cause an interrupt, and its associated vector fetch. These Software Interrupts are useful in operating system calls, software debugging, trace operations, memory mapping, and software development systems. Three levels of SWI are available on this HD6809, and are prioritized in the following order: SWI, SWI2, SWI3.

16-Bit Operation

The HD6809 has the capability of processing 16-bit data. These instructions include loads, stores, compares, adds, subtracts, transfers, exchanges, pushes and pulls.

■ CYCLE-BY-CYCLE OPERATION

The address bus cycle-by-cycle performance chart illustrates the memory-access sequence corresponding to each possible instruction and addressing mode in the HD6809. Each instruction begins with an opcode fetch. While that opcode is being internally decoded, the next program byte is always fetched. (Most instructions will use the next byte, so this technique considerably speeds throughput.) Next, the operation of each opcode will follow the flow chart. $\overline{\text{VMA}}$ is an indication of FFFF$_{16}$ on the address bus, R/$\overline{\text{W}}$="High" and BS="Low". The following examples illustrate the use of the chart; see Fig. 20.

Example 1: LBSR (Branch Taken)
Before Execution SP = F000

```
                          •
                          •
                          •
$8000                     LBSR       CAT
                          •
                          •
                          •
$A000       CAT           •
```

CYCLE-BY-CYCLE FLOW

Cycle #	Address	Data	R/$\overline{\text{W}}$	Description
1	8000	17	1	Opcode Fetch
2	8001	1F	1	Offset High Byte
3	8002	FD	1	Offset Low Byte
4	FFFF	*	1	$\overline{\text{VMA}}$ Cycle
5	FFFF	*	1	$\overline{\text{VMA}}$ Cycle
6	A000	*	1	Computed Branch Address
7	FFFF	*	1	$\overline{\text{VMA}}$ Cycle
8	EFFF	03	0	Stack Low Order Byte of Return Address
9	EFFE	80	0	Stack High Order Byte of Return Address

Example 2: DEC (Extended)

```
$8000          DEC           $A000
$A000          FCB           $80
```

CYCLE-BY-CYCLE FLOW

Cycle #	Address	Data	R/$\overline{\text{W}}$	Description
1	8000	7A	1	Opcode Fetch
2	8001	A0	1	Operand Address, High Byte
3	8002	00	1	Operand Address, Low Byte
4	FFFF	*	1	$\overline{\text{VMA}}$ Cycle
5	A000	80	1	Read the Data
6	FFFF	*	1	$\overline{\text{VMA}}$ Cycle
7	A000	7F	0	Store the Decremented Data

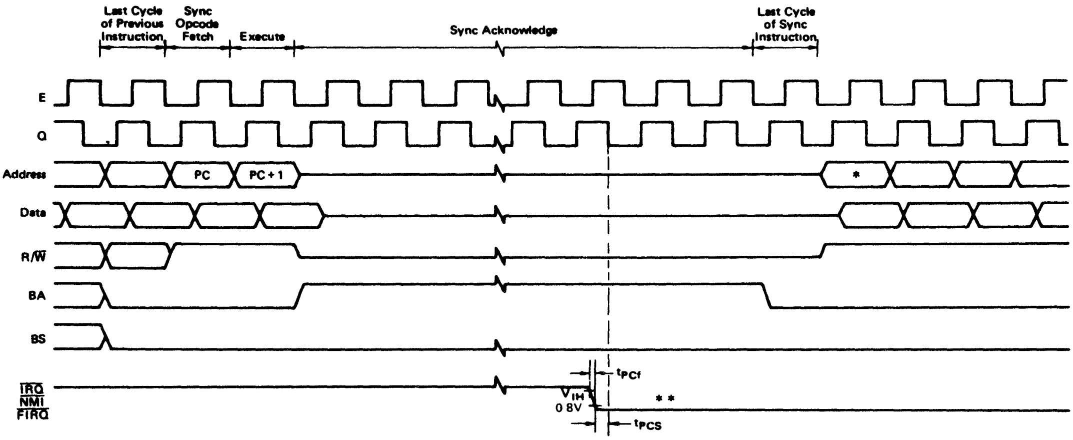
* The data bus has the data at that particular address.

■ HD6809 INSTRUCTION SET TABLES

The instructions of the HD6809 have been broken down into five different categories. They are as follows:

8-Bit operation (Table 5)
16-Bit operation (Table 6)
Index register/stack pointer instructions (Table 7)
Relative branches (long or short) (Table 8)
Miscellaneous instructions (Table 9)

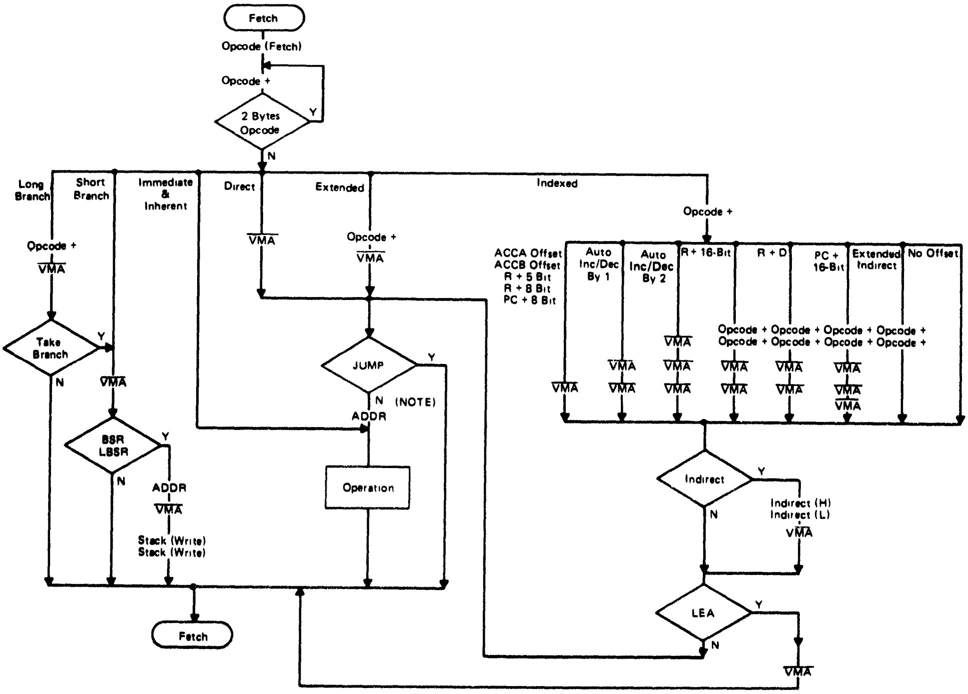
HD6809 instruction set tables and Hexadecimal Values of instructions are shown in Table 10 and Table 11.



(NOTES) * If the associated mask bit is set when the interrupt is requested, this cycle will be an instruction fetch from address location PC + 1. However, if the interrupt is accepted ($\overline{\text{NMI}}$ or an unmasked $\overline{\text{FIRQ}}$ or $\overline{\text{IRQ}}$) interrupt processing continues with this cycle as (m) on Figure 9 and 10 (Interrupt Timing).

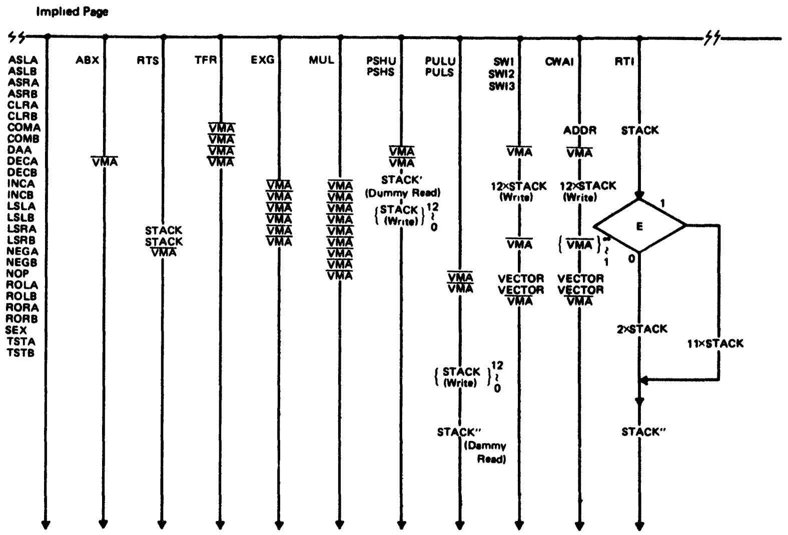
** If mask bits are clear, $\overline{\text{IRQ}}$ and $\overline{\text{FIRQ}}$ must be held "Low" for three cycles to guarantee that interrupt will be taken, although only one cycle is necessary to bring the processor out of SYNC.

Figure 19 Sync Timing



(NOTE) Write operation during store instruction.

Figure 20 Address Bus Cycle-by-Cycle Performance



(NOTE) STACK' : Address stored in stack pointer before execution.
STACK'': Address set to stack pointer as the result of the execution.

Figure 20 Address Bus Cycle-by-Cycle Performance (Continued)

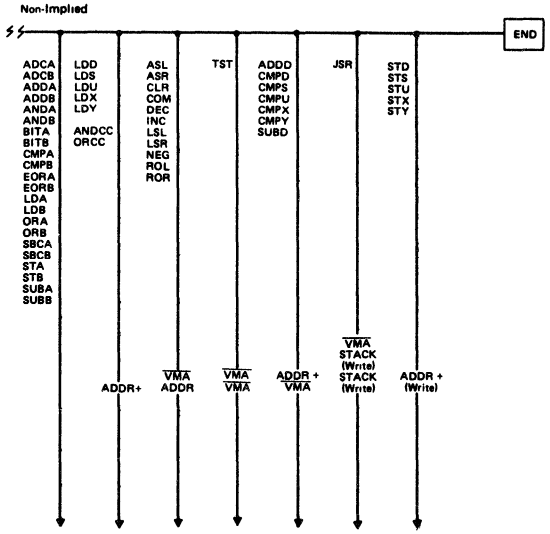


Figure 20 Address Bus Cycle-by-Cycle Performance (Continued)

Table 5 8-Bit Accumulator and Memory Instructions

Mnemonic(s)	Operation
ADCA, ADCB	Add memory to accumulator with carry
ADDA, ADDB	Add memory to accumulator
ANDA, ANDB	And memory with accumulator
ASL, ASLA, ASLB	Arithmetic shift of accumulator or memory left
ASR, ASRA, ASRB	Arithmetic shift of accumulator or memory right
BITA, BITB	Bit test memory with accumulator
CLR, CLRA, CLRB	Clear accumulator or memory location
CMPA, CMPB	Compare memory from accumulator
COM, COMA, COMB	Complement accumultor or memory location
DAA	Decimal adjust A accumulator
DEC, DECA, DECB	Decrement accumulator or memory location
EORA, EORB	Exclusive or memory with accumulator
EXG R1, R2	Exchange R1 with R2 (R1, R2 = A, B, CC, DP)
INC, INCA, INCB	Increment accumulator or memory location
LDA, LDB	Load accumulator from memory
LSL, LSLA, LSLB	Logical shift left accumulator or memory location
LSR, LSRA, LSRB	Logical shift right accumulator or memory location
MUL	Unsigned multiply (A × B → D)
NEG, NEGA, NEGB	Negate accumulator or memory
ORA, ORB	Or memory with accumulator
ROL, ROLA, ROLB	Rotate accumulator or memory left
ROR, RORA, RORB	Rotate accumulator or memory right
SBCA, SBCB	Subtract memory from accumulator with borrow
STA, STB	Store accumulator to memory
SUBA, SUBB	Subtract memory from accumulator
TST, TSTA, TSTB	Test accumulator or memory location
TFR R1, R2	Transfer R1 to R2 (R1, R2 = A, B, CC, DP)

(NOTE) A, B, CC or DP may be pushed to (pulled from) either stack with PSHS, PSHU (PULS, PULU) instructions.

Table 6 16-Bit Accumulator and Memory Instructions

Mnemonic(s)	Operation
ADDD	Add memory to D accumulator
CMPD	Compare memory from D accumulator
EXG D, R	Exchange D with X, Y, S, U or PC
LDD	Load D accumulator from memory
SEX	Sign Extend B accumulator into A accumulator
STD	Store D accumulator to memory
SUBD	Subtract memory from D accumulator
TFR D, R	Transfer D to X, Y, S, U or PC
TFR R, D	Transfer X, Y, S, U or PC to D

(NOTE) D may be pushed (pulled) to either stack with PSHS, PSHU (PULS, PULU) instructions.

Table 7 Index Register/Stack Pointer Instructions

Mnemonic(s)	Operation
CMPS, CMPU	Compare memory from stack pointer
CMPX, CMPY	Compare memory from index register
EXG R1, R2	Exchange D, X, Y, S, U or PC with D, X, Y, S, U or PC
LEAS, LEAU	Load effective address into stack pointer
LEAX, LEAY	Load effective address into index register
LDS, LDU	Load stack pointer from memory
LDX, LDY	Load index register from memory
PSHS	Push A, B, CC, DP, D, X, Y, U, or PC onto hardware stack
PSHU	Push A, B, CC, DP, D, X, Y, S, or PC onto user stack
PULS	Pull A, B, CC, DP, D, X, Y, U or PC from hardware stack
PULU	Pull A, B, CC, DP, D, X, Y, S or PC from user stack
STS, STU	Store stack pointer to memory
STX, STY	Store index register to memory
TFR R1, R2	Transfer D, X, Y, S, U or PC to D, X, Y, S, U or PC
ABX	Add B accumulator to X (unsigned)

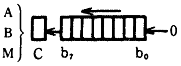
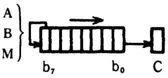
Table 8 Branch Instructions

Mnemonic(s)	Operation
	SIMPLE BRANCHES
BEQ, LBEQ	Branch if equal
BNE, LBNE	Branch if not equal
BMI, LBMI	Branch if minus
BPL, LBPL	Branch if plus
BCS, LBCS	Branch if carry set
BCC, LBCC	Branch if carry clear
BVS, LBVS	Branch if overflow set
BVC, LBVC	Branch if overflow clear
	SIGNED BRANCHES
BGT, LBGT	Branch if greater (signed)
BGE, LBGE	Branch if greater than or equal (signed)
BEQ, LBEQ	Branch if equal
BLE, LBLE	Branch if less than or equal (signed)
BLT, LBLT	Branch if less than (signed)
	UNSIGNED BRANCHES
BHI, LBHI	Branch if higher (unsigned)
BHS, LBHS	Branch if higher or same (unsigned)
BEQ, LBEQ	Branch if equal
BLS, LBLS	Branch if lower or same (unsigned)
BLO, LBLO	Branch if lower (unsigned)
	OTHER BRANCHES
BSR, LBSR	Branch to subroutine
BRA, LBRA	Branch always
BRN, LBRN	Branch never

Table 9 Miscellaneous Instructions

Mnemonic(s)	Operation
ANDCC	AND condition code register
CWAI	AND condition code register, then wait for interrupt
NOP	No operation
ORCC	OR condition code register
JMP	Jump
JSR	Jump to subroutine
RTI	Return from interrupt
RTS	Return from subroutine
SWI, SWI2, SWI3	Software interrupt (absolute indirect)
SYNC	Synchronize with interrupt line

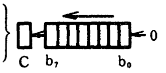
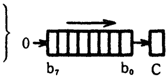
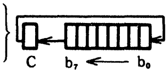
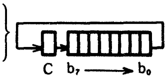
Table 10. HD6809 Instruction Set Table

INSTRUCTION/ FORMS		IMP ACCM REG			DIRECT			EXTND			IMMED			INDEX①			RELATIVE			DESCRIPTION	7	6	5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~⑤	#		E	F	H	I	N	Z	V	C
ABX		3A	3	1																B + X→X	●	●	●	●	●	●	●	●
																				(UNSIGNED)								
ADC	ADCA				99	4	2	B9	5	3	89	2	2	A9	4 +	2 +				A + M + C→A	●	●	↕	●	↕	↕	↕	↕
	ADCB				D9	4	2	F9	5	3	C9	2	2	E9	4 +	2 +				B + M + C→B	●	●	↕	●	↕	↕	↕	↕
ADD	ADDA				9B	4	2	BB	5	3	8B	2	2	AB	4 +	2 +				A + M→A	●	●	↕	●	↕	↕	↕	↕
	ADDB				DB	4	2	FB	5	3	CB	2	2	EB	4 +	2 +				B + M→B	●	●	↕	●	↕	↕	↕	↕
	ADDD				D3	6	2	F3	7	3	C3	4	3	E3	6 +	2 +				D + M M + 1→D	●	●	●	●	↕	↕	↕	↕
AND	ANDA				94	4	2	B4	5	3	84	2	2	A4	4 +	2 +				A∧M→A	●	●	●	●	↕	↕	R	●
	ANDB				D4	4	2	F4	5	3	C4	2	2	E4	4 +	2 +				B∧M→B	●	●	●	●	↕	↕	R	●
	ANDCC										1C	3	2							CC∧IMM→CC	(			⑦				)
ASL	ASLA	48	2	1																	●	●	⑧	●	↕	↕	↕	↕
	ASLB	58	2	1																	●	●	⑧	●	↕	↕	↕	↕
	ASL				08	6	2	78	7	3				68	6 +	2 +					●	●	⑧	●	↕	↕	↕	↕
ASR	ASRA	47	2	1																	●	●	⑧	●	↕	↕	●	↕
	ASRB	57	2	1																	●	●	⑧	●	↕	↕	●	↕
	ASR				07	6	2	77	7	3				67	6 +	2 +					●	●	⑧	●	↕	↕	●	↕
BCC	BCC																24	3	2	Branch C = 0	●	●	●	●	●	●	●	●
	LBCC																10	5(6)	4	Long Branch	●	●	●	●	●	●	●	●
																	24			C = 0								
BCS	BCS																25	3	2	Branch C = 1	●	●	●	●	●	●	●	●
	LBCS																10	5(6)	4	Long Branch	●	●	●	●	●	●	●	●
																	25			C = 1								
BEQ	BEQ																27	3	2	Branch Z = 1	●	●	●	●	●	●	●	●
	LBEQ																10	5(6)	4	Long Branch	●	●	●	●	●	●	●	●
																	27			Z = 1								
BGE	BGE																2C	3	2	Branch N⊕V = 0	●	●	●	●	●	●	●	●
	LBGE																10	5(6)	4	Long Branch	●	●	●	●	●	●	●	●
																	2C			N⊕V = 0								
BGT	BGT																2E	3	2	Branch Z∨(N⊕V) = 0	●	●	●	●	●	●	●	●
	LBGT																10	5(6)	4	Long Branch	●	●	●	●	●	●	●	●
																	2E			Z∨(N⊕V) = 0								
BHI	BHI																22	3	2	Branch C∨Z = 0	●	●	●	●	●	●	●	●
	LBHI																10	5(6)	4	Long Branch	●	●	●	●	●	●	●	●
																	22			C∨Z = 0								
BHS	BHS																24	3	2	Branch C = 0	●	●	●	●	●	●	●	●
	LBHS																10	5(6)	4	Long Branch	●	●	●	●	●	●	●	●
																	24			C = 0								
BIT	BITA				95	4	2	B5	5	3	85	2	2	A5	4 +	2 +				Bit Test A (M∧A)	●	●	●	●	↕	↕	R	●
	BITB				D5	4	2	F5	5	3	C5	2	2	E5	4 +	2 +				Bit Test B (M∧B)	●	●	●	●	↕	↕	R	●
BLE	BLE																2F	3	2	Branch Z∨(N⊕V) = 1	●	●	●	●	●	●	●	●
	LBLE																10	5(6)	4	Long Branch	●	●	●	●	●	●	●	●
																	2F			Z∨(N⊕V) = 1								
BLO	BLO																25	3	2	Branch C = 1	●	●	●	●	●	●	●	●
	LBLO																10	5(6)	4	Long Branch	●	●	●	●	●	●	●	●
																	25			C = 1								
BLS	BLS																23	3	2	Branch C∨Z = 1	●	●	●	●	●	●	●	●
	LBLS																10	5(6)	4	Long Branch	●	●	●	●	●	●	●	●
																	23			C∨Z = 1								
BLT	BLT																2D	3	2	Branch N⊕V = 1	●	●	●	●	●	●	●	●
	LBLT																10	5(6)	4	Long Branch	●	●	●	●	●	●	●	●
																	2D			N⊕V = 1								
BMI	BMI																2B	3	2	Branch N = 1	●	●	●	●	●	●	●	●
	LBMI																10	5(6)	4	Long Branch	●	●	●	●	●	●	●	●
																	2B			N = 1								

(Continued)

INSTRUCTION/ FORMS		IMP ACCM REG.			DIRECT			EXTND			IMMED			INDEX①			RELATIVE			DESCRIPTION	7	6	5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~⑤	#		E	F	H	I	N	Z	V	C
BNE	BNE																26	3	2	Branch Z = 0	●	●	●	●	●	●	●	●
	LBNE																10	5(6)	4	Long Branch	●	●	●	●	●	●	●	●
																	26			Z = 0								
BPL	BPL																2A	3	2	Branch N = 0	●	●	●	●	●	●	●	●
	LBPL																10	5(6)	4	Long Branch	●	●	●	●	●	●	●	●
																	2A			N = 0								
BRA	BRA																20	3	2	Branch Always	●	●	●	●	●	●	●	●
	LBRA																16	5	3	Long Branch Always	●	●	●	●	●	●	●	●
BRN	BRN																21	3	2	Branch Never	●	●	●	●	●	●	●	●
	LBRN																10	5	4	Long Branch Never	●	●	●	●	●	●	●	●
																	21											
BSR	BSR																8D	7	2	Branch to Subroutine	●	●	●	●	●	●	●	●
	LBSR																17	9	3	Long Branch to Subroutine	●	●	●	●	●	●	●	●
BVC	BVC																28	3	2	Branch V = 0	●	●	●	●	●	●	●	●
	LBVC																10	5(6)	4	Long Branch	●	●	●	●	●	●	●	●
																	28			V = 0								
BVS	BVS																29	3	2	Branch V = 1	●	●	●	●	●	●	●	●
	LBVS																10	5(6)	4	Long Branch	●	●	●	●	●	●	●	●
																	29			V = 1								
CLR	CLRA	4F	2	1																0→A	●	●	●	●	R	S	R	R
	CLRB	5F	2	1																0→B	●	●	●	●	R	S	R	R
	CLR				0F	6	2	7F	7	3				6F	6+	2+				0→M	●	●	●	●	R	S	R	R
CMP	CMPA				91	4	2	B1	5	3	81	2	2	A1	4+	2+				Compare M from A	●	●	⑧	●	↕	↕	↕	↕
	CMPB				D1	4	2	F1	5	3	C1	2	2	E1	4+	2+				Compare M from B	●	●	⑧	●	↕	↕	↕	↕
	CMPD				10	7	3	10	8	4	10	5	4	10	7+	3+				Compare M M+1	●	●	●	●	↕	↕	↕	↕
					93			B3			83			A3						from D								
	CMPS				11	7	3	11	8	4	11	5	4	11	7+	3+				Compare M M+1	●	●	●	●	↕	↕	↕	↕
					9C			BC			8C			AC						from S								
	CMPU				11	7	3	11	8	4	11	5	4	11	7+	3+				Compare M M+1	●	●	●	●	↕	↕	↕	↕
					93			B3			83			A3						from U								
	CMPX				9C	6	2	BC	7	3	8C	4	3	AC	6+	2+				Compare M M+1	●	●	●	●	↕	↕	↕	↕
																				from X								
	CMPY				10	7	3	10	8	4	10	5	4	10	7+	3+				Compare M M+1	●	●	●	●	↕	↕	↕	↕
					9C			BC			8C			AC						from Y								
COM	COMA	43	2	1																$\overline{A}$→A	●	●	●	●	↕	↕	R	S
	COMB	53	2	1																$\overline{B}$→B	●	●	●	●	↕	↕	R	S
	COM				03	6	2	73	7	3				63	6+	2+				$\overline{M}$→M	●	●	●	●	↕	↕	R	S
CWAI		3C	20	2																CC ∧ IMM→CC Wait for Interrupt	S	(			⑦			)
DAA		19	2	1																Decimal Adjust A	●	●	●	●	↕	↕	⑧	↕
DEC	DECA	4A	2	1																A − 1→A	●	●	●	●	↕	↕	↕	●
	DECB	5A	2	1																B − 1→B	●	●	●	●	↕	↕	↕	●
	DEC				0A	6	2	7A	7	3				6A	6+	2+				M − 1→M	●	●	●	●	↕	↕	↕	●
EOR	EORA				98	4	2	B8	5	3	88	2	2	A8	4+	2+				A⊕M→A	●	●	●	●	↕	↕	R	●
	EORB				D8	4	2	F8	5	3	C8	2	2	E8	4+	2+				B⊕M→B	●	●	●	●	↕	↕	R	●
EXG	R1, R2	1E	8	2																R1→R2②	(			⑩				)
INC	INCA	4C	2	1																A + 1→A	●	●	●	●	↕	↕	↕	●
	INCB	5C	2	1																B + 1→B	●	●	●	●	↕	↕	↕	●
	INC				0C	6	2	7C	7	3				6C	6+	2+				M + 1→M	●	●	●	●	↕	↕	↕	●
JMP					0E	3	2	7E	4	3				6E	3+	2+				EA③→PC	●	●	●	●	●	●	●	●
JSR					9D	7	2	BD	8	3				AD	7+	2+				Jump to Subroutine	●	●	●	●	●	●	●	●

(Continued)

INSTRUCTION/ FORMS		IMP ACCM REG.			DIRECT			EXTND			IMMED			INDEX①			RELATIVE			DESCRIPTION	7	6	5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~⑤	#		E	F	H	I	N	Z	V	C
LD	LDA				96	4	2	B6	5	3	86	2	2	A6	4+	2+				M→A	●	●	●	●	↕	↕	R	●
	LDB				D6	4	2	F6	5	3	C6	2	2	E6	4+	2+				M→B	●	●	●	●	↕	↕	R	●
	LDD				DC	5	2	FC	6	3	CC	3	3	EC	5+	2+				MM+1→D	●	●	●	●	↕	↕	R	●
	LDS				10	6	3	10	7	4	10	4	4	10	6+	3+				MM+1→S	●	●	●	●	↕	↕	R	●
					DE			FE			CE			EE														
	LDU				DE	5	2	FE	6	3	CE	3	3	EE	5+	2+				MM+1→U	●	●	●	●	↕	↕	R	●
	LDX				9E	5	2	BE	6	3	8E	3	3	AE	5+	2+				MM+1→X	●	●	●	●	↕	↕	R	●
	LDY				10	6	3	10	7	4	10	4	4	10	6+	3+				MM+1→Y	●	●	●	●	↕	↕	R	●
					9E			BE			8E			AE														
LEA	LEAS													32	4+	2+				EA③→S	●	●	●	●	●	●	●	●
	LEAU													33	4+	2+				EA③→U	●	●	●	●	●	●	●	●
	LEAX													30	4+	2+				EA③→X	●	●	●	●	●	↕	●	●
	LEAY													31	4+	2+				EA③→Y	●	●	●	●	●	↕	●	●
LSL	LSLA	48	2	1																A 	●	●	●	●	↕	↕	↕	↕
	LSLB	58	2	1																B	●	●	●	●	↕	↕	↕	↕
	LSL				08	6	2	78	7	3				68	6+	2+				M	●	●	●	●	↕	↕	↕	↕
LSR	LSRA	44	2	1																A 	●	●	●	●	R	↕	●	↕
	LSRB	54	2	1																B	●	●	●	●	R	↕	●	↕
	LSR				04	6	2	74	7	3				64	6+	2+				M	●	●	●	●	R	↕	●	↕
MUL		3D	11	1																A×B→D (Unsigned)	●	●	●	●	●	↕	●	⑨
NEG	NEGA	40	2	1																$\overline{A}$+1→A	●	●	⑧	●	↕	↕	↕	↕
	NEGB	50	2	1																$\overline{B}$+1→B	●	●	⑧	●	↕	↕	↕	↕
	NEG				00	6	2	70	7	3				60	6+	2+				$\overline{M}$+1→M	●	●	⑧	●	↕	↕	↕	↕
NOP		12	2	1																No Operation	●	●	●	●	●	●	●	●
OR	ORA				9A	4	2	BA	5	3	8A	2	2	AA	4+	2+				A∨M→A	●	●	●	●	↕	↕	R	●
	ORB				DA	4	2	FA	5	3	CA	2	2	EA	4+	2+				B∨M→B	●	●	●	●	↕	↕	R	●
	ORCC										1A	3	2							CC∨ IMM→CC	(	—	—	⑦	—	—	—	)
PSH	PSHS	34	5+④	2																Push Registers on S Stack	●	●	●	●	●	●	●	●
	PSHU	36	5+④	2																Push Registers on U Stack	●	●	●	●	●	●	●	●
PUL	PULS	35	5+④	2																Pull Registers from S Stack	(	—	—	⑩	—	—	—	)
	PULU	37	5+④	2																Pull Registers from U Stack	(	—	—	⑩	—	—	—	)
ROL	ROLA	49	2	1																A 	●	●	●	●	↕	↕	↕	↕
	ROLB	59	2	1																B	●	●	●	●	↕	↕	↕	↕
	ROL				09	6	2	79	7	3				69	6+	2+				M	●	●	●	●	↕	↕	↕	↕
ROR	RORA	46	2	1																A 	●	●	●	●	↕	↕	●	↕
	RORB	56	2	1																B	●	●	●	●	↕	↕	●	↕
	ROR				06	6	2	76	7	3				66	6+	2+				M	●	●	●	●	↕	↕	●	↕
RTI		3B	6 15	1																Return from Interrupt	(	—	—	⑦	—	—	—	)
RTS		39	5	1																Return from Subroutine	●	●	●	●	●	●	●	●
SBC	SBCA				92	4	2	B2	5	3	82	2	2	A2	4+	2+				A−M−C→A	●	●	⑧	●	↕	↕	↕	↕
	SBCB				D2	4	2	F2	5	3	C2	2	2	E2	4+	2+				B−M−C→B	●	●	⑧	●	↕	↕	↕	↕
SEX		1D	2	1																Sign Extend B into A {Bのビット7＝1 FF→A {Bのビット7＝0 0→A	●	●	●	●	↕	↕	●	●

(Continued)

INSTRUCTIONS/ FORMS		IMP ACCM REG			DIRECT			EXTND			IMMED			INDEX①			RELATIVE			DESCRIPTION	7	6	5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~⑤	#		E	F	H	I	N	Z	V	C
ST	STA				97	4	2	B7	5	3				A7	4+	2+				A→M	●	●	●	●	↕	↕	R	●
	STB				D7	4	2	F7	5	3				E7	4+	2+				B→M	●	●	●	●	↕	↕	R	●
	STD				DD	5	2	FD	6	3				ED	5+	2+				D→M M+1	●	●	●	●	↕	↕	R	●
	STS				10	6	3	10	7	4				10	6+	3+				S→M M+1	●	●	●	●	↕	↕	R	●
					DF			FF						EF														
	STU				DF	5	2	FF	6	3				EF	5+	2+				U→M M+1	●	●	●	●	↕	↕	R	●
	STX				9F	5	2	BF	6	3				AF	5+	2+				X→M M+1	●	●	●	●	↕	↕	R	●
	STY				10	6	3	10	7	4				10	6+	3+				Y→M M+1	●	●	●	●	↕	↕	R	●
					9F			BF						AF														
SUB	SUBA				90	4	2	B0	5	3	80	2	2	A0	4+	2+				A−M→A	●	●	⑧	●	↕	↕	↕	↕
	SUBB				D0	4	2	F0	5	3	C0	2	2	E0	4+	2+				B−M→B	●	●	⑧	●	↕	↕	↕	↕
	SUBD				93	6	2	B3	7	3	83	4	3	A3	6+	2+				D−M M+1→D	●	●	●	●	↕	↕	↕	↕
SWI	SWI⑥	3F	19	1																Software interrupt 1	S	S	●	S	●	●	●	●
	SWI2⑥	10	20	2																Software interrupt 2	S	●	●	●	●	●	●	●
		3F																										
	SWI3⑥	11	20	2																Software interrupt 3	S	●	●	●	●	●	●	●
		3F																										
SYNC		13	≥2	1																Synchronize to interrupt	●	●	●	●	●	●	●	●
TFR	R1, R2	1F	6	2																R1 → R2②	(			⑩				)
TST	TSTA	4D	2	1																Test A	●	●	●	●	↕	↕	R	●
	TSTB	5D	2	1																Test B	●	●	●	●	↕	↕	R	●
	TST				0D	6	2	7D	7	3				6D	6+	2+				Test M	●	●	●	●	↕	↕	R	●

(NOTES)

① This column gives a base cycle and byte count. To obtain total count, and the values obtained from the INDEXED ADDRESSING MODES table
② R1 and R2 may be any pair of 8 bit or any pair of 16 bit registers
The 8 bit registers are: A, B, CC, DP
The 16 bit registers are: X, Y, U, S, D, PC
③ EA is the effective address
④ The PSH and PUL instructions require 5 cycle plus 1 cycle for each byte pushed or pulled
⑤ 5(6) means· 5 cycles if branch not taken, 6 cycles if taken.
⑥ SWI sets 1 and F bits. SW12 and SW13 do not affect I and F.
⑦ Conditions Codes set as a direct result of the instruction
⑧ Value of half-carry flag is undefined.
⑨ Special Case—Carry set if b7 is SET.
⑩ Condition Codes set as a direct result of the instruction if CC is specified, and not affected otherwise

LEGEND:

OP	Operation Code (Hexadecimal)	Z	Zero (byte)
~	Number of MPU Cycles	V	Overflow, 2's complement
#	Number of Program Bytes	C	Carry from bit 7
+	Arithmetic Plus	↕	Test and set if true, cleared otherwise
−	Arithmetic Minus	●	Not Affected
×	Multiply	CC	Condition Code Register
$\overline{M}$	Complement of M		Concatenation
→	Transfer Into	∨	Logical or
H	Half-carry (from bit 3)	∧	Logical and
N	Negative (sign bit)	⊕	Logical Exclusive or

Table 11 Hexadecimal Values of Machine Codes

OP	Mnem	Mode	~	#	OP	Mnem	Mode	~	#	OP	Mnem	Mode	~	#
00	NEG	Direct	6	2	30	LEAX	Indexed	4+	2+	60	NEG	Indexed	6+	2+
01	*				31	LEAY		4+	2+	61	*			
02	*				32	LEAS		4+	2+	62	*			
03	COM		6	2	33	LEAU	Indexed	4+	2+	63	COM		6+	2+
04	LSR		6	2	34	PSHS	Implied	5+	2	64	LSR		6+	2+
05	*				35	PULS		5+	2	65	*			
06	ROR		6	2	36	PSHU		5+	2	66	ROR		6+	2+
07	ASR		6	2	37	PULU		5+	2	67	ASR		6+	2+
08	ASL, LSL		6	2	38	*				68	ASL, LSL		6+	2+
09	ROL		6	2	39	RTS		5	1	69	ROL		6+	2+
0A	DEC		6	2	3A	ABX		3	1	6A	DEC		6+	2+
0B	*				3B	RTI	Implied	6, 15	1	6B	*			
0C	INC		6	2	3C	CWAI	Immed	≧20	2	6C	INC		6+	2+
0D	TST		6	2	3D	MUL	Implied	11	1	6D	TST		6+	2+
0E	JMP		3	2	3E	*				6E	JMP		3+	2+
0F	CLR	Direct	6	2	3F	SWI	Implied	19	1	6F	CLR	Indexed	6+	2+
10	} See	–	–	–	40	NEGA	Implied	2	1	70	NEG	Extended	7	3
11	Next Page	–	–	–	41	*				71	*			
12	NOP	Implied	2	1	42	*				72	*			
13	SYNC	Implied	≧4	1	43	COMA		2	1	73	COM		7	3
14	*				44	LSRA		2	1	74	LSR		7	3
15	*				45	*				75	*			
16	LBRA	Relative	5	3	46	RORA		2	1	76	ROR		7	3
17	LBSR	Relative	9	3	47	ASRA		2	1	77	ASR		7	3
18	*				48	ASLA, LSLA		2	1	78	ASL, LSL		7	3
19	DAA	Implied	2	1	49	ROLA		2	1	79	ROL		7	3
1A	ORCC	Immed	3	2	4A	DECA		2	1	7A	DEC		7	3
1B	*				4B	*				7B	*			
1C	ANDCC	Immed	3	2	4C	INCA		2	1	7C	INC		7	3
1D	SEX	Implied	2	1	4D	TSTA		2	1	7D	TST		7	3
1E	EXG		8	2	4E	*				7E	JMP		4	3
1F	TFR	Implied	6	2	4F	CLRA	Implied	2	1	7F	CLR	Extended	7	3
20	BRA	Relative	3	2	50	NEGB	Implied	2	1	80	SUBA	Immed	2	2
21	BRN		3	2	51	*				81	CMPA		2	2
22	BHI		3	2	52	*				82	SBCA		2	2
23	BLS		3	2	53	COMB		2	1	83	SUBD		4	3
24	BHS, BCC		3	2	54	LSRB		2	1	84	ANDA		2	2
25	BLO, BCS		3	2	55	*				85	BITA		2	2
26	BNE		3	2	56	RORB		2	1	86	LDA		2	2
27	BEQ		3	2	57	ASRB		2	1	87	*			
28	BVC		3	2	58	ASLB, LSLB		2	1	88	EORA		2	2
29	BVS		3	2	59	ROLB		2	1	89	ADCA		2	2
2A	BPL		3	2	5A	DECB		2	1	8A	ORA		2	2
2B	BMI		3	2	5B	*				8B	ADDA		2	2
2C	BGE		3	2	5C	INCB		2	1	8C	CMPX	Immed	4	3
2D	BLT		3	2	5D	TSTB		2	1	8D	BSR	Relative	7	2
2E	BGT		3	2	5E	*				8E	LDX	Immed	3	3
2F	BLE	Relative	3	2	5F	CLRB	Implied	2	1	8F	*			

LEGEND:
~ Number of MPU cycles (less possible push pull or indexed-mode cycles)
# Number of program bytes
* Denotes unused opcode

(to be continued)

OP	Mnem	Mode	~	#	OP	Mnem	Mode	~	#	OP	Mnem	Mode	~	#
90	SUBA	Direct	4	2	C6	LDB	Immed	2	2	FC	LDD	Extended	6	3
91	CMPA		4	2	C7	*				FD	STD		6	3
92	SBCA		4	2	C8	EORB		2	2	FE	LDU		6	3
93	SUBD		6	2	C9	ADCB		2	2	FF	STU	Extended	6	3
94	ANDA		4	2	CA	ORB		2	2					
95	BITA		4	2	CB	ADDB		2	2					
96	LDA		4	2	CC	LDD		3	3		2 Bytes Opcode			
97	STA		4	2	CD	*								
98	EORA		4	2	CE	LDU	Immed	3	3	1021	LBRN	Relative	5	4
99	ADCA		4	2	CF	*				1022	LBHI		5(6)	4
9A	ORA		4	2						1023	LBLS		5(6)	4
9B	ADDA		4	2	D0	SUBB	Direct	4	2	1024	LBHS, LBCC		5(6)	4
9C	CMPX		6	2	D1	CMPB		4	2	1025	LBCS, LBLO		5(6)	4
9D	JSR		7	2	D2	SBCB		4	2	1026	LBNE		5(6)	4
9E	LDX		5	2	D3	ADDD		6	2	1027	LBEQ		5(6)	4
9F	STX	Direct	5	2	D4	ANDB		4	2	1028	LBVC		5(6)	4
					D5	BITB		4	2	1029	LBVS		5(6)	4
A0	SUBA	Indexed	4+	2+	D6	LDB		4	2	102A	LBPL		5(6)	4
A1	CMPA		4+	2+	D7	STB		4	2	102B	LBMI		5(6)	4
A2	SBCA		4+	2+	D8	EORB		4	2	102C	LBGE		5(6)	4
A3	SUBD		6+	2+	D9	ADCB		4	2	102D	LBLT		5(6)	4
A4	ANDA		4+	2+	DA	ORB		4	2	102E	LBGT		5(6)	4
A5	BITA		4+	2+	DB	ADDB		4	2	102F	LBLE	Relative	5(6)	4
A6	LDA		4+	2+	DC	LDD		5	2	103F	SWI2	Implied	20	2
A7	STA		4+	2+	DD	STD		5	2	1083	CMPD	Immed	5	4
A8	EORA		4+	2+	DE	LDU		5	2	108C	CMPY		5	4
A9	ADCA		4+	2+	DF	STU	Direct	5	2	108E	LDY	Immed	4	4
AA	ORA		4+	2+						1093	CMPD	Direct	7	3
AB	ADDA		4+	2+	E0	SUBB	Indexed	4+	2+	109C	CMPY		7	3
AC	CMPX		6+	2+	E1	CMPB		4+	2+	109E	LDY		6	3
AD	JSR		7+	2+	E2	SBCB		4+	2+	109F	STY	Direct	6	3
AE	LDX		5+	2+	E3	ADDD		6+	2+	10A3	CMPD	Indexed	7+	3+
AF	STX	Indexed	5+	2+	E4	ANDB		4+	2+	10AC	CMPY		7+	3+
					E5	BITB		4+	2+	10AE	LDY		6+	3+
B0	SUBA	Extended	5	3	E6	LDB		4+	2+	10AF	STY	Indexed	6+	3+
B1	CMPA		5	3	E7	STB		4+	2+	10B3	CMPD	Extended	8	4
B2	SBCA		5	3	E8	EORB		4+	2+	10BC	CMPY		8	4
B3	SUBD		7	3	E9	ADCB		4+	2+	10BE	LDY		7	4
B4	ANDA		5	3	EA	ORB		4+	2+	10BF	STY	Extended	7	4
B5	BITA		5	3	EB	ADDB		4+	2+	10CE	LDS	Immed	4	4
B6	LDA		5	3	EC	LDD		5+	2+	10DE	LDS	Direct	6	3
B7	STA		5	3	ED	STD		5+	2+	10DF	STS	Direct	6	3
B8	EORA		5	3	EE	LDU		5+	2+	10EE	LDS	Indexed	6+	3+
B9	ADCA		5	3	EF	STU	Indexed	5+	2+	10EF	STS	Indexed	6+	3+
BA	ORA		5	3						10FE	LDS	Extended	7	4
BB	ADDA		5	3	F0	SUBB	Extended	5	3	10FF	STS	Extended	7	4
BC	CMPX		7	3	F1	CMPB		5	3	113F	SWI3	Implied	20	2
BD	JSR		8	3	F2	SBCB		5	3	1183	CMPU	Immed	5	4
BE	LDX		6	3	F3	ADDD		7	3	118C	CMPS	Immed	5	4
BF	STX	Extended	6	3	F4	ANDB		5	3	1193	CMPU	Direct	7	3
					F5	BITB		5	3	119C	CMPS	Direct	7	3
C0	SUBB	Immed	2	2	F6	LDB		5	3	11A3	CMPU	Indexed	7+	3+
C1	CMPB		2	2	F7	STB		5	3	11AC	CMPS	Indexed	7+	3+
C2	SBCB		2	2	F8	EORB		5	3	11B3	CMPU	Extended	8	4
C3	ADDD		4	3	F9	ADCB		5	3	11BC	CMPS	Extended	8	4
C4	ANDB		2	2	FA	ORB		5	3					
C5	BITB	Immed	2	2	FB	ADDB	Extended	5	3					

(NOTE): All unused opcodes are both undefined and illegal

■ NOTE FOR USE

[1] Exceptional Operation of HD6809

(a) Exceptional Operations of $\overline{\text{DMA/BREQ}}$, BA signals (#1)

HD6809 acknowledges the input signal level of $\overline{\text{DMA/BREQ}}$ at the end of each cycle, then determines whether the next sequence is MPU or DMA. When "Low" level is detected, HD6809 executes DMA sequence by setting BA, BS to "High" level. However, in the conditions shown below the assertion of BA, BS delays one clock cycle.

< Conditions for the exception >

(1) $\overline{\text{DMA/BREQ}}$: "Low" for 6~13 cycles
(2) $\overline{\text{DMA/BREQ}}$: "High" for 3 cycles

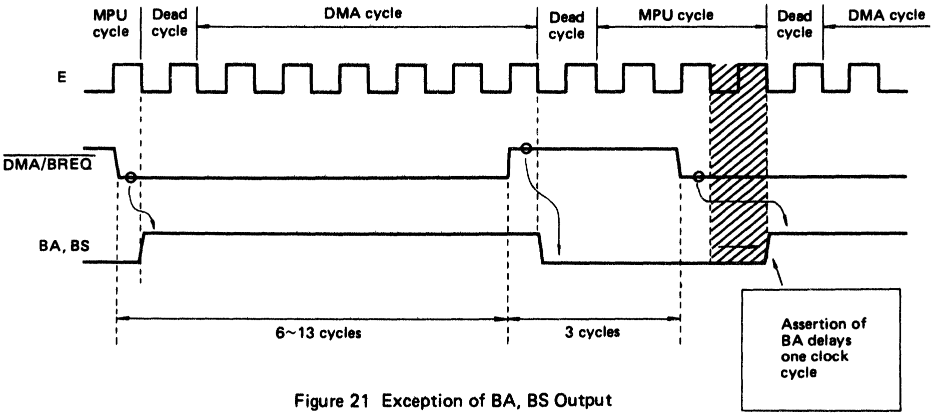


Figure 21 Exception of BA, BS Output

(b) Exceptional Operations of $\overline{\text{DMA/BREQ}}$, BA signals (#2)

HD6809 includes a self refresh counter for the reverce cycle steal. And it is only cleared if $\overline{\text{DMA/BREQ}}$ is inactive ("High") for 3 or more MPU cycles. So 1 or 2 inactive cycle(s) doesn't affect the self refresh counter.

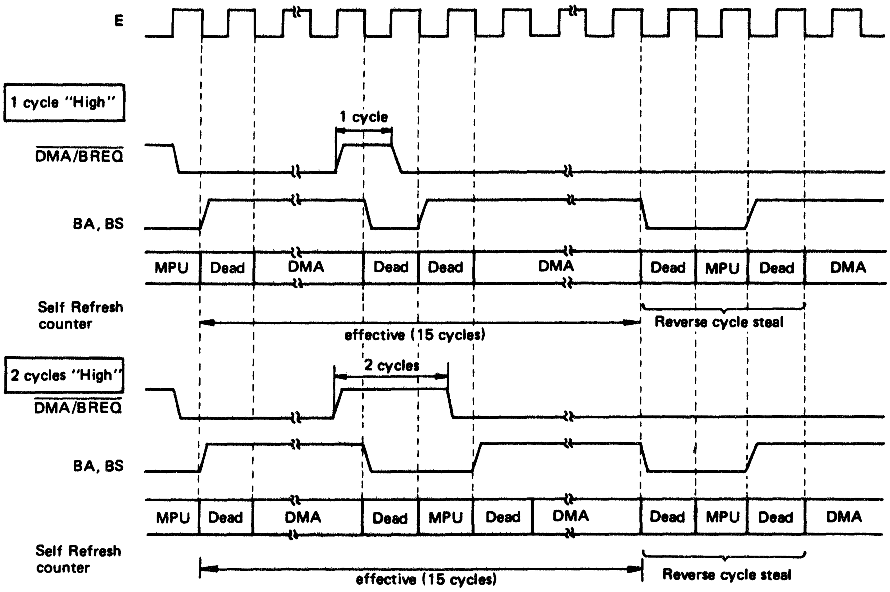


Figure 22 Exception of $\overline{\text{DMA/BREQ}}$

(c) **How to avoid these exceptional operations**

It is necessary to provide 4 or more cycles for inactive DMA/BREQ level as shown in Fig. 23.

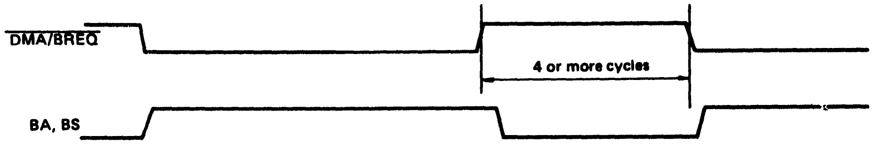


Figure 23 How to Avoid Exceptional Operations

[2] **Restriction for DMA Transfer**

There is a restriction for the DMA transfer in the HD6809 (MPU), HD6844 (DMAC) system. Please take care of following.

(a) **An Example of the System Configuration**

This restriction is applied to the following system.

(1) DMA/BREQ is used for DMA request.

(2) "Halt Burst Mode" is used for DMA transfer

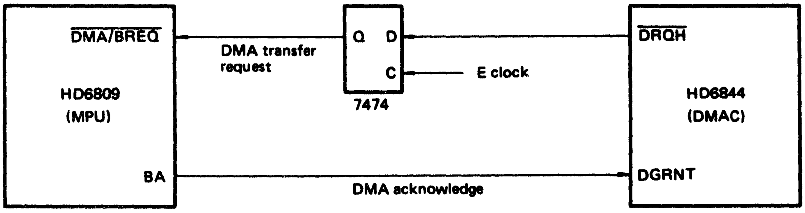


Figure 24 An Example of HD6809, HD6844 System

(The restriction is also applied to the system which doesn't use 7474 Flip-Flop. Fig. 24, Fig. 25 shows an example which uses 7474 for synchronizing DMA request with E.)

(b) **Restriction**

"The number of transfer Byte per one DMA Burst transfer must be less than or equal to 14."

Halt burst DMA transfer should be less than or equal to 14 cycles. In another word, the number stored into DMA Byte count register should be 0~14.

★ Please than care of the section [1](b) if 2 or more DMA channels are used for the DMA transfer.

(c) **Incorrect operation of HD6809, HD6844 system**

"Incorrect Operation" will occur if the number of DMA transfer Byte is more than 14 bytes. If DMA/BREQ is kept in "Low" level HD6809 performs reverse cycle steals once in 14 DMA cycles by taking back the bus control. In this case, however, the action taken by MPU is a little bit different from the DMAC.

As shown in Fig. 25, DMA controller can't stop DMA transfer (Ⓐ) by BA falling edge and excutes an extra DMA cycle during HD6809 dead cycle. So MPU cycle is excuted right after DMA cycle, the Bus confliction occurs at the beginning of MPU cycle.

(d) **How to impliment Halt Bust DMA transfer (> 14 cycles)**

Please use HALT input of HD6809 for the DMA request instead of DMA/BREQ.

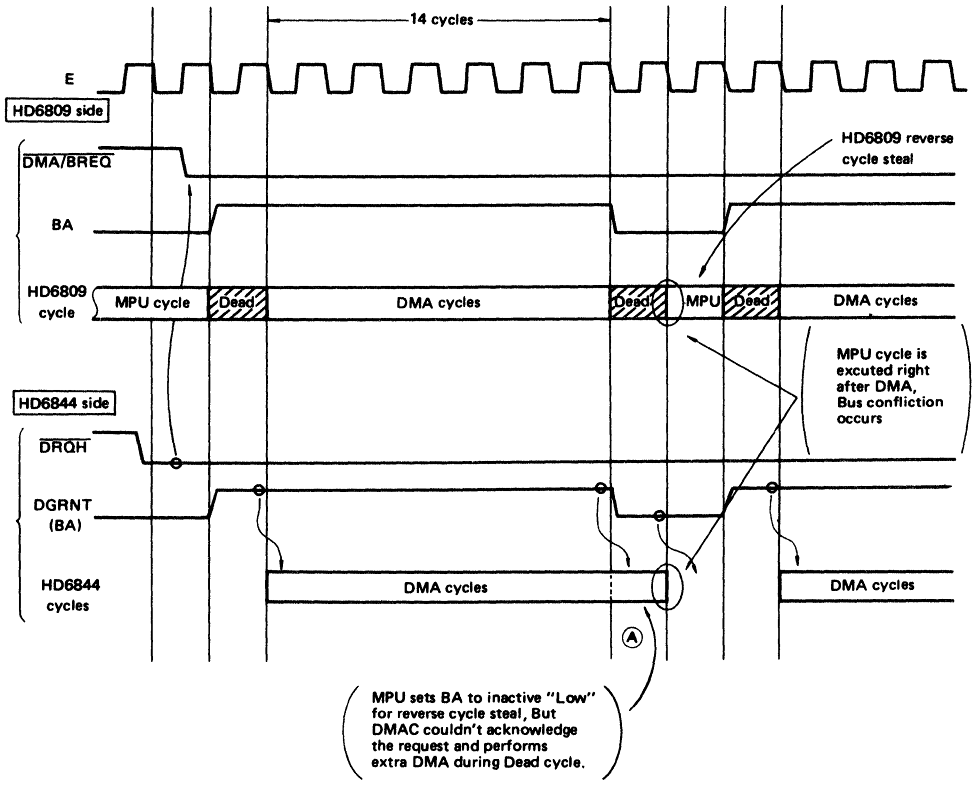

14 cycles
E
HD6809 side
$\overline{\text{DMA/BREQ}}$
BA
HD6809 cycle
MPU cycle
Dead
DMA cycles
Dead
MPU
Dead
DMA cycles
HD6809 reverse cycle steal
MPU cycle is excuted right after DMA, Bus confliction occurs
HD6844 side
$\overline{\text{DRQH}}$
DGRNT (BA)
HD6844 cycles
DMA cycles
DMA cycles
A
MPU sets BA to inactive "Low" for reverse cycle steal, But DMAC couldn't acknowledge the request and performs extra DMA during Dead cycle.

Figure 25 Comparison of HD6809, HD6844 DMA cycles

[3] Note for CLR Instruction

Cycle-by-cycle flow of CLR instruction (Direct, Extended, Indexed Addressing Mode) is shown below. In this sequence the content of the memory location specified by the operand is read before writing "00" into it. Note that status Flags, such as IRQ Flag, will be cleared by this extra data read operation when accessing the control/status register (sharing the same address between read and write) of peripheral devices.

Example: CLR (Extended)

```
$8000    CLR    $A000
$A000    FCB    $80
```

Cycle #	Address	Data	R/$\overline{\text{W}}$	Description
1	8000	7F	1	Opcode Fetch
2	8001	A0	1	Operand Address, High Byte
3	8002	00	1	Operand Address, Low Byte
4	FFFF	*	1	$\overline{\text{VMA}}$ Cycle
5	A000	80	1	Read the Data
6	FFFF	*	1	$\overline{\text{VMA}}$ Cycle
7	A000	00	0	Store Fixed "00" into Specified Location

* The data bus has the data at that particular address.

[4] **Note for MRDY**

HD6809 require synchronization of the MRDY input with the 4f clock. The synchronization necessitates an external oscillator as shown in Figure 26. The negative transition of the MRDY signal, normally derived from the chip select decoding, must meet the t_{PCS} timing. MRDY's positive transition must occur with the rising edge of 4f.

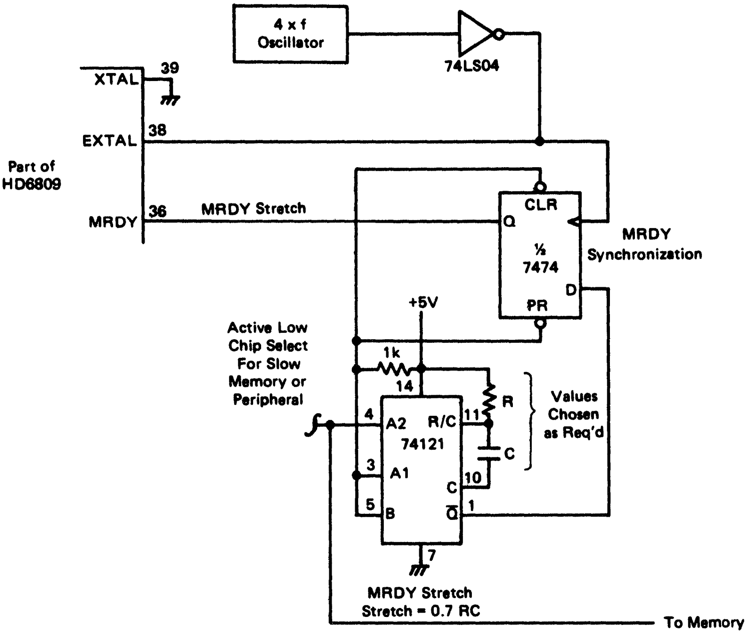


Figure 26 MRDY Synchronization

HD6809E, HD68A09E, HD68B09E
MPU(Micro Processing Unit)

The HD6809E is a revolutionary high performance 8-bit microprocessor which supports modern programming techniques such as position independence, reentrancy, and modular programming.

This third-generation addition to the HMCS6800 family has major architectural improvements which include additional registers, instructions and addressing modes.

The basic instructions of any computer are greatly enhanced by the presence of powerful addressing modes. The HD6809E has the most complete set of addressing modes available on any 8-bit microprocessor today.

The HD6809E has hardware and software features which make it an ideal processor for higher level language execution or standard controller applications. External clock inputs are provided to allow synchronization with peripherals, systems or other MPUs.

HD6800 COMPATIBLE

- Hardware – Interfaces with All HMCS6800 Peripherals
- Software – Upward Source Code Compatible Instruction Set and Addressing Modes

■ ARCHITECTURAL FEATURES

- Two 16-bit Index Registers
- Two 16-bit Indexable Stack Pointers
- Two 8-bit Accumulators can be Concatenated to Form One 16-Bit Accumulator
- Direct Page Register Allows Direct Addressing Throughout Memory

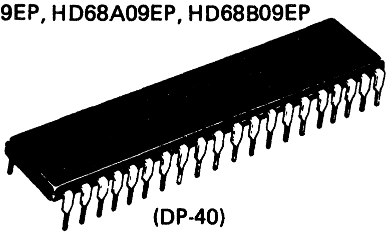
■ HARDWARE FEATURES

- External Clock Inputs, E and Q, Allow Synchronization
- TSC Input Controls Internal Bus Buffers
- LIC Indicates Opcode Fetch
- AVMA Allows Efficient Use of Common Resources in A Multiprocessor System
- BUSY is a Status Line for Multiprocessing
- Fast Interrupt Request Input Stacks Only Condition Code Register and Program Counter
- Interrupt Acknowledge Output Allows Vectoring By Devices
- SYNC Acknowledge Output Allows for Synchronization to External Event
- Single Bus-Cycle RESET
- Single 5-Volt Supply Operation
- NMI Blocked After RESET Until After First Load of Stack Pointer
- Early Address Valid Allows Use With Slower Memories
- Early Write-Data for Dynamic Memories

■ SOFTWARE FEATURES

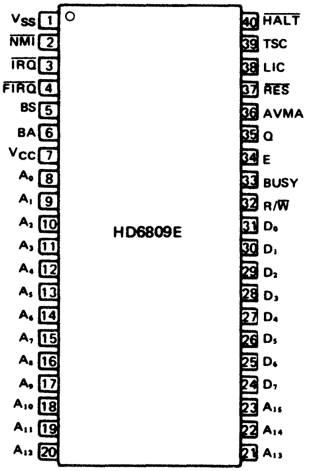
- 10 Addressing Modes
  - HMCS6800 Upward Compatible Addressing Modes
  - Direct Addressing Anywhere in Memory Map
  - Long Relative Branches
  - Program Counter Relative
  - True Indirect Addressing
  - Expanded Indexed Addressing:
    0, 5, 8, or 16-bit Constant Offsets
    8, or 16-bit Accumulator Offsets
    Auto-Increment/Decrement by 1 or 2
- Improved Stack Manipulation
- 1464 Instruction with Unique Addressing Modes
- 8 x 8 Unsigned Multiply
- 16-bit Arithmetic
- Transfer/Exchange All Registers
- Push/Pull Any Registers or Any Set of Registers
- Load Effective Address

HD6809EP, HD68A09EP, HD68B09EP



(DP-40)

■ PIN ARRANGEMENT



(Top View)

■ ABSOLUTE MAXIMUM RATINGS

Item	Symbol	Value	Unit
Supply Voltage	Vcc*	–0.3 ~ +7.0	V
Input Voltage	Vin*	–0.3 ~ +7.0	V
Operating Temperature Range	Topr	–20 ~ +75	°C
Storage Temperature Range	Tstg	–55 ~ +150	°C

* With respect to Vss (SYSTEM GND)

(NOTE) Permanent LSI damage may occur if maximum ratings are exceeded. Normal operation should be under recommended operating conditions. If these conditions are exceeded, it could affect reliability of LSI.

■ RECOMMENDED OPERATING CONDITIONS

Item		Symbol	min	typ	max	unit
Supply Voltage		Vcc*	4.75	5.0	5.25	V
Input Voltage	Logic, Q, $\overline{\text{RES}}$	V_{IL}*	-0.2	–	0.8	V
	E	V_{ILC}*	-0.3	–	0.4	V
	Logic	V_{IH}*	2.2	–	Vcc*	V
	$\overline{\text{RES}}$		4.0	–	Vcc*	V
	E	V_{IHC}*	Vcc* –0.75	–	Vcc* +0.3	V
Operating Temperature		Topr	–20	25	75	°C

* With respect to Vss (SYSTEM GND)

■ ELECTRICAL CHARACTERISTICS

● DC CHARACTERISTICS (V_{CC} = 5.0V ±5%, V_{SS} = 0V, Ta = -20 ~ +75°C, unless otherwise noted.)

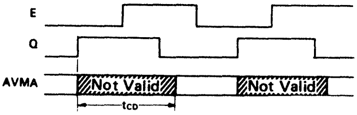
Item		Symbol	Test Condition	HD6809E			HD68A09E			HD68B09E			Unit
				min	typ*	max	min	typ*	max	min	typ*	max	
Input "High" Voltage	Logic, Q	V_{IH}		2.2	–	Vcc	2.2	–	Vcc	2.2	–	Vcc	V
	$\overline{\text{RES}}$	V_{IHR}		4.0	–	Vcc	4.0	–	Vcc	4.0	–	Vcc	V
	E	V_{IHC}		Vcc –0.75	–	Vcc +0.3	Vcc –0.75	–	Vcc +0.3	Vcc –0.75	–	Vcc +0.3	V
Input "Low" Voltage	Logic, Q, $\overline{\text{RES}}$	V_{IL}		–0.2	–	0.8	–0.2	–	0.8	–0.2	–	0.8	V
	E	V_{ILC}		–0.3	–	0.4	–0.3	–	0.4	–0.3	–	0.4	V
Input Leakage Current	Logic, Q, $\overline{\text{RES}}$	I_{in}	Vin = 0 ~ 5.25V, Vcc = max	–2.5	–	2.5	–2.5	–	2.5	–2.5	–	2.5	μA
	E			–100	–	100	–100	–	100	–100	–	100	μA
Output "High" Voltage	D_0 ~ D_7	V_{OH}	I_{Load} = –205μA, Vcc = min	2.4	–	–	2.4	–	–	2.4	–	–	V
	A_0 ~ A_{15}, $R/\overline{W}$		I_{Load} = –145μA, Vcc = min	2.4	–	–	2.4	–	–	2.4	–	–	V
	BA, BS, LIC, AVMA, BUSY		I_{Load} = –100μA, Vcc = min	2.4	–	–	2.4	–	–	2.4	–	–	V
Output "Low" Voltage		V_{OL}	I_{Load} = 2mA, Vcc = min	–	–	0.5	–	–	0.5	–	–	0.5	V
Power Dissipation		P_D		–	–	1.0	–	–	1.0	–	–	1.0	W
Input Capacitance	D_0 ~ D_7, Logic Input, Q, $\overline{\text{RES}}$	C_{in}	Vin = 0V, Ta = 25°C, f = 1MHz	–	10	15	–	10	15	–	10	15	pF
	E			–	30	50	–	30	50	–	30	50	pF
Output Capacitance	A_0 ~ A_{15}, $R/\overline{W}$, BA, BS, LIC, AVMA, BUSY	Cout	Vin = 0V, Ta = 25°C, f = 1MHz	–	10	15	–	10	15	–	10	15	pF
Frequency of Operation	E, Q	f		0.1	–	1.0	0.1	–	1.5	0.1	–	2.0	MHz
Three-State (Off State) Input Current	D_0 ~ D_7	I_{TSI}	Vin = 0.4 ~ 2.4V, Vcc = max	–10	–	10	–10	–	10	–10	–	10	μA
	A_0 ~ A_{15}, $R/\overline{W}$			–100	–	100	–100	–	100	–100	–	100	μA

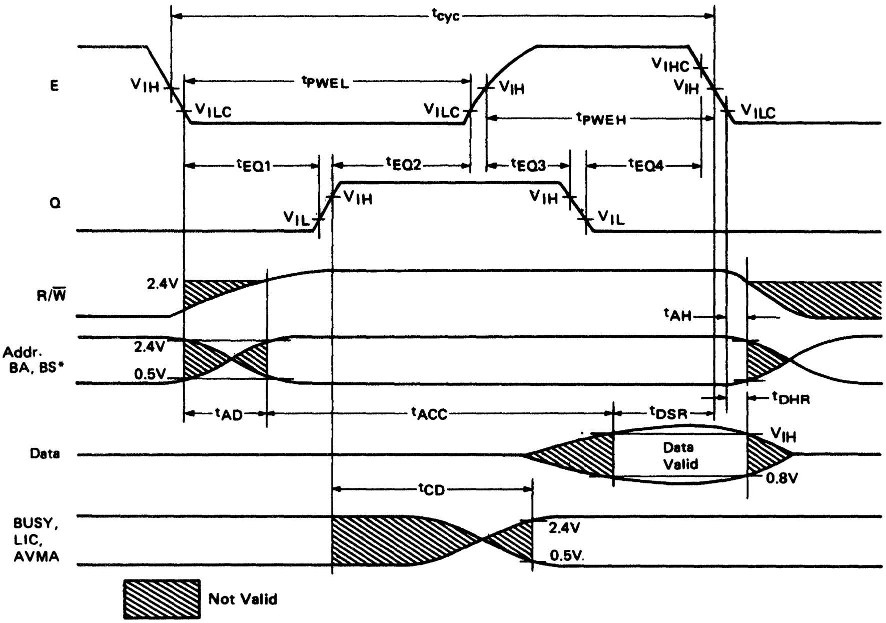
* Ta = 25°C, V_{CC} = 5V

- **AC CHARACTERISTICS (V_{CC} = 5.0V ±5%, V_{SS} = 0V, Ta = -20 ~ +75°C, unless otherwise noted.) READ/WRITE TIMING**

Item		Symbol	Test Condition	HD6809E			HD68A09E			HD68B09E			Unit
				min	typ	max	min	typ	max	min	typ	max	
Cycle Time		t_{cyc}	Fig. 1, 2, 7 ~ 10, 14 and 17	1000	–	10000	667	–	10000	500	–	10000	ns
Peripheral Read Access Times $t_{cyc} - t_{Ef} - t_{AD} - t_{DSR} = t_{ACC}$		t_{ACC}		695	–	–	440	–	–	330	–	–	ns
Data Setup Time (Read)		t_{DSR}		80	–	–	60	–	–	40	–	–	ns
Input Data Hold Time		t_{DHR}		10	–	–	10	–	–	10	–	–	ns
Output Data Hold Time	Ta = 0 ~ +75°C	t_{DHW}		30	–	–	30	–	–	30	–	–	ns
	Ta = -20 ~ 0°C			20	–	–	20	–	–	20	–	–	ns
Address Hold Time (Address, R/$\overline{W}$)	Ta = 0 ~ +75°C	t_{AH}		20	–	–	20	–	–	20	–	–	ns
	Ta = -20 ~ 0°C			10	–	–	10	–	–	10	–	–	ns
Address Delay		t_{AD}		–	–	200	–	–	140	–	–	120	ns
Data Delay Time (Write)		t_{DDW}		–	–	200	–	–	140	–	–	110	ns
E Clock "Low"		t_{PWEL}		450	–	9500	295	–	9500	210	–	9500	ns
E Clock "High" (Measured at V_{IH})		t_{PWEH}		450	–	9500	280	–	9500	220	–	9500	ns
E Rise and Fall Time		t_{Er}, t_{Ef}		–	–	25	–	–	25	–	–	20	ns
Q Clock "High"		t_{PWQH}		450	–	9500	280	–	9500	220	–	9500	ns
Q Rise and Fall Time		t_{Qr}, t_{Qf}		–	–	25	–	–	25	–	–	20	ns
E "Low" to Q Rising		t_{EQ1}		200	–	–	130	–	–	100	–	–	ns
Q "High" to E Rising		t_{EQ2}		200	–	–	130	–	–	100	–	–	ns
E "High" to Q Falling		t_{EQ3}		200	–	–	130	–	–	100	–	–	ns
Q "Low" to E Falling		t_{EQ4}		200	–	–	130	–	–	100	–	–	ns
Interrupts $\overline{HALT}$, $\overline{RES}$ and TSC Setup Time		t_{PCS}		200	–	–	140	–	–	110	–	–	ns
TSC Drive to Valid Logic Levels		t_{TSA}		–	–	210	–	–	150	–	–	120	ns
TSC Release MOS Buffers to High Impedance		t_{TSR}		–	–	200	–	–	140	–	–	110	ns
TSC Three-State Delay		t_{TSD}		–	–	120	–	–	85	–	–	80	ns
Control Delay (BUSY, LIC)		t_{CD}		–	–	300	–	–	250	–	–	200	ns
Control Delay (AVMA*)		t_{CD}		–	–	300	–	–	270	–	–	240	ns
Processor Control Rise/Fall		t_{PCr}, t_{PCf}		–	–	100	–	–	100	–	–	100	ns
TSC Input Delay		t_{PCT}		10	–	–	10	–	–	10	–	–	ns

* AVMA drives a not-valid data before providing correct output, so spec t_{CD} max = 270 nsec (HD68A09E) and 240 nsec (HD68B09E) are applied to this signal. When this delay time causes a problem in user's application, please use D-type latch to get stable output.

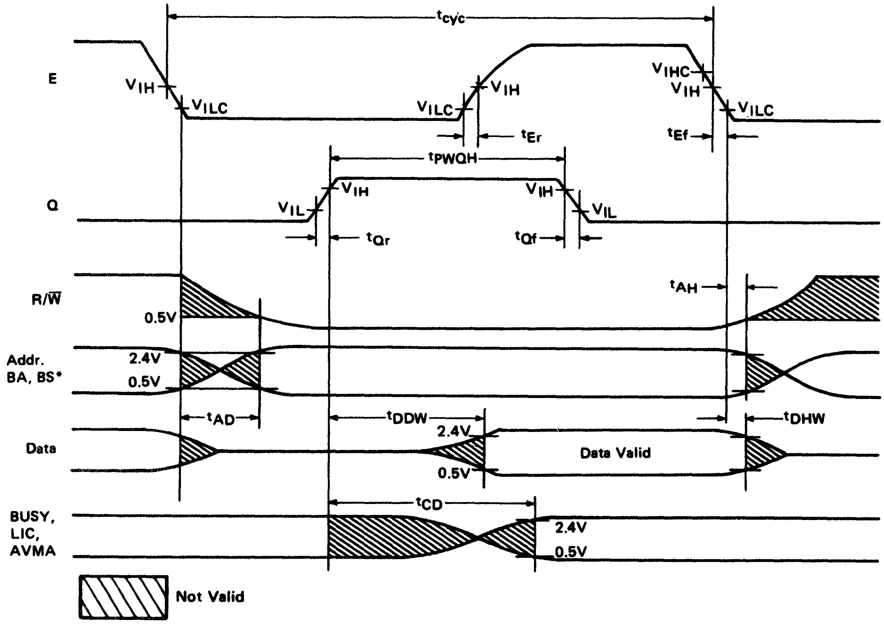




* Hold time for BA, BS not specified

(NOTE) Waveform measurements for all inputs and outputs are specified at logic "High" = V_{IHmin} and logic "Low" = V_{ILmax} unless otherwise specified.

Figure 1 Read Data from Memory or Peripherals



* Hold time for BA, BS not specified

(NOTE) Waveform measurements for all inputs and outputs are specified at logic "High" = V_{IHmin} and logic "Low" = V_{ILmax} unless otherwise specified.

Figure 2 Write Data to Memory or Peripherals

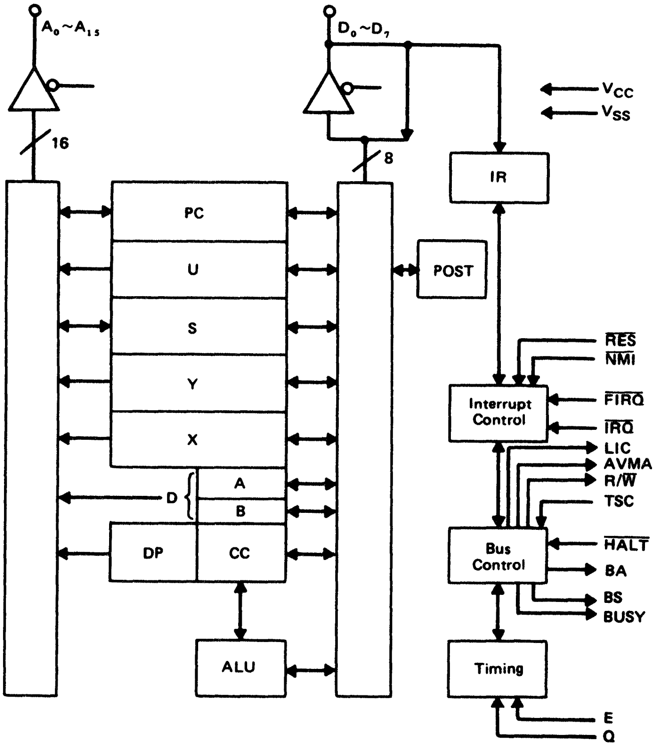
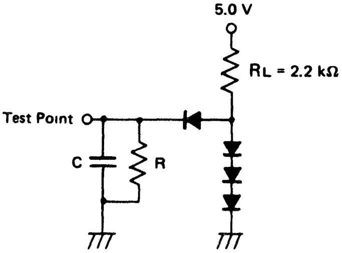


Figure 3 HD6809E Expanded Block Diagram



C = 30 pF for BA, BS, LIC, AVMA, BUSY
130 pF for $D_0 \sim D_7$
90 pF for $A_0 \sim A_{15}$, R/W̄

R = 11.7 kΩ for $D_0 \sim D_7$
16.5 kΩ for $A_0 \sim A_{15}$, R/W̄
24 kΩ for BA, BS, LIC, AVMA, BUSY

All diodes are 1S2074Ⓗ or equivalent.
C includes stray capacitance.

Figure 4 Bus Timing Test Load

■ PROGRAMMING MODEL

As shown in Figure 5, the HD6809E adds three registers to the set available in the HD6800. The added registers include a Direct Page Register, the User Stack pointer and a second Index Register.

● Accumulators (A, B, D)

The A and B registers are general purpose accumulators which are used for arithmetic calculations and manipulation of data.

Certain instructions concatenate the A and B registers to form a single 16-bit accumulator. This is referred to as the D Register, and is formed with the A Register as the most significant byte.

● Direct Page Register (DP)

The Direct Page Register of the HD6809E serves to enhance the Direct Addressing Mode. The content of this register appears at the higher address outputs ($A_8 \sim A_{15}$) during direct addressing instruction execution. This allows the direct mode to be used at any place in memory, under program control. To ensure HD6800 compatibility, all bits of this register are cleared during Processor Reset.

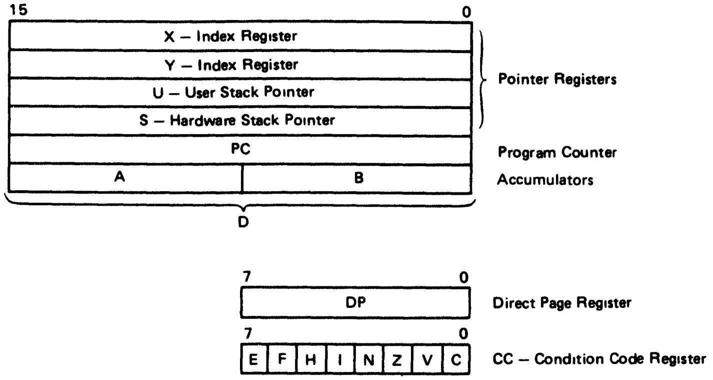


Figure 5 Programming Model of The Microprocessing Unit

- **Index Registers (X, Y)**

The Index Registers are used in indexed mode of addressing. The 16-bit address in this register takes part in the calculation of effective addresses. This address may be used to point to data directly or may be modified by an optional constant or register offset. During some indexed modes, the contents of the index register are incremented or decremented to point to the next item of tabular type data. All four pointer registers (X, Y, U, S) may be used as index registers.

- **Stack Pointer (U, S)**

The Hardware Stack Pointer (S) is used automatically by the processor during subroutine calls and interrupts. The User Stack Pointer (U) is controlled exclusively by the programmer thus allowing arguments to be passed to and from subroutines with ease. The U-register is frequently used as a stack marker. Both Stack Pointers have the same indexed mode addressing capabilities as the X and Y registers, but also support **Push** and **Pull** instructions. This allows the HD6809E to be used efficiently as a stack processor, greatly enhancing its ability to support higher level languages and modular programming.

(NOTE) The stack pointers of the HD6809E point to the top of the stack, in contrast to the HD6800 stack pointer, which pointed to the next free location on stack.

- **Program Counter (PC)**

The Program Counter is used by the processor to point to the address of the next instruction to be executed by the processor. Relative Addressing is provided allowing the Program Counter to be used like an index register in some situations.

- **Condition Code Register (CC)**

The Condition Code Register defines the state of the processor at any given time. See Figure 6.

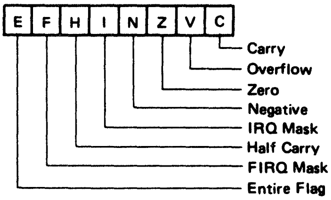


Figure 6 Condition Code Register Format

▪ CONDITION CODE REGISTER DESCRIPTION

- **Bit 0 (C)**

Bit 0 is the carry flag, and is usually the carry from the binary ALU. C is also used to represent a 'borrow' from subtract like instructions (CMP, NEG, SUB, SBC) and is the complement of the carry from the binary ALU.

- **Bit 1 (V)**

Bit 1 is the overflow flag, and is set to a one by an operation which causes a signed two's complement arithmetic overflow. This overflow is detected in an operation in which the carry from the MSB in the ALU does not match the carry from the MSB-1.

- **Bit 2 (Z)**

Bit 2 is the zero flag, and is set to a one if the result of the previous operation was identically zero.

- **Bit 3 (N)**

Bit 3 is the negative flag, which contains exactly the value of the MSB of the result of the preceding operation. Thus, a negative two's-complement result will leave N set to a one.

- **Bit 4 (I)**

Bit 4 is the $\overline{\text{IRQ}}$ mask bit. The processor will not recognize interrupts from the $\overline{\text{IRQ}}$ line if this bit is set to a one. $\overline{\text{NMI}}$, $\overline{\text{FIRQ}}$, $\overline{\text{IRQ}}$, $\overline{\text{RES}}$ and SWI all set I to a one; SWI2 and SWI3 do not affect I.

- **Bit 5 (H)**

Bit 5 is the half-carry bit, and is used to indicate a carry from bit 3 in the ALU as a result of an 8-bit addition only (ADC or ADD). This bit is used by the DAA instruction to perform a BCD decimal add adjust operation. The state of this flag is undefined in all subtract-like instructions.

- **Bit 6 (F)**

Bit 6 is the $\overline{\text{FIRQ}}$ mask bit. The processor will not recognize interrupts from the $\overline{\text{FIRQ}}$ line if this bit is a one. $\overline{\text{NMI}}$, $\overline{\text{FIRQ}}$, SWI, and $\overline{\text{RES}}$ all set F to a one. $\overline{\text{IRQ}}$, SWI2 and SWI3 do not affect F.

- **Bit 7 (E)**

Bit 7 is the entire flag, and when set to a one indicates that the complete machine state (all the registers) was stacked, as opposed to the subset state (PC and CC). The E bit of the stacked CC is used on a return from interrupt (RTI) to determine the extent of the unstacking. Therefore, the current E left in the Condition Code Register represents past action.

■ HD6809E MPU SIGNAL DESCRIPTION

- **Power (Vss, Vcc)**

Two pins are used to supply power to the part: Vss is ground or 0 volts, while Vcc is +5.0 V ±5%.

- **Address Bus ($A_0 \sim A_{15}$)**

Sixteen pins are used to output address information from the MPU onto the Address Bus. When the processor does not require the bus for a data transfer, it will output address FFFF_{16}, R/$\overline{\text{W}}$ = "High", and BS = "Low"; this is a "dummy access" or $\overline{\text{VMA}}$ cycle. All address bus drivers are made high-impedance when output Bus Available (BA) is "High" or when TSC is asserted. Each pin will drive one Schottky TTL load or four LS TTL loads, and 90 pF. Refer to Figures 1 and 2.

- **Data Bus ($D_0 \sim D_7$)**

These eight pins provide communication with the system bi-directional data bus. Each pin will drive one Schottky TTL load or four LS TTL loads, and 130 pF.

- **Read/Write (R/$\overline{\text{W}}$)**

This signal indicates the direction of data transfer on the data bus. A "Low" indicates that the MPU is writing data onto the data bus. R/$\overline{\text{W}}$ is made high impedance when BA is "High" or when TSC is asserted. Refer to Figures 1 and 2.

- **$\overline{\text{RES}}$**

A "Low" level on this Schmitt-trigger input for greater than one bus cycle will reset the MPU, as shown in Figure 7. The Reset vectors are fetched from locations FFFE_{16} and FFFF_{16} (Table 1) when Interrupt Acknowledge is true, ($\overline{\text{BA}}$ · BS = 1). During initial power-on, the Reset line should be held "Low" until the clock input signals are fully operational.

Because the HD6809E Reset pin has a Schmitt-trigger input with a threshold voltage higher than that of standard peripherals, a simple R/C network may be used to reset the entire system. This higher threshold voltage ensures that all peripherals are out of the reset state before the Processor.

Table 1 Memory Map for Interrupt Vectors

Memory Map for Vector Locations		Interrupt Vector Description
MS	LS	
FFFE	FFFF	$\overline{\text{RES}}$
FFFC	FFFD	$\overline{\text{NMI}}$
FFFA	FFFB	SWI
FFF8	FFF9	$\overline{\text{IRQ}}$
FFF6	FFF7	$\overline{\text{FIRQ}}$
FFF4	FFF5	SWI2
FFF2	FFF3	SWI3
FFF0	FFF1	Reserved

- **$\overline{\text{HALT}}$**

A "Low" level on this input pin will cause the MPU to stop running at the end of the present instruction and remain halted indefinitely without loss of data. When halted, the BA output is driven "High" indicating the buses are high impedance. BS is also "High" which indicates the processor is in the Halt state. While halted, the MPU will not respond to external real-time requests ($\overline{\text{FIRQ}}$, $\overline{\text{IRQ}}$) although $\overline{\text{NMI}}$ or $\overline{\text{RES}}$ will be latched for later response. During the Halt state Q and E should continue to run normally. A halted state (BA · BS = 1) can be achieved by pulling $\overline{\text{HALT}}$ "Low" while $\overline{\text{RES}}$ is still "Low". See Figure 8.

- **Bus Available, Bus Status (BA, BS)**

The Bus Available output is an indication of an internal control signal which makes the MOS buses of the MPU high impedance. When BA goes "Low", a dead cycle will elapse before the MPU acquires the bus. BA will not be asserted when TSC is active, thus allowing dead cycle consistency.

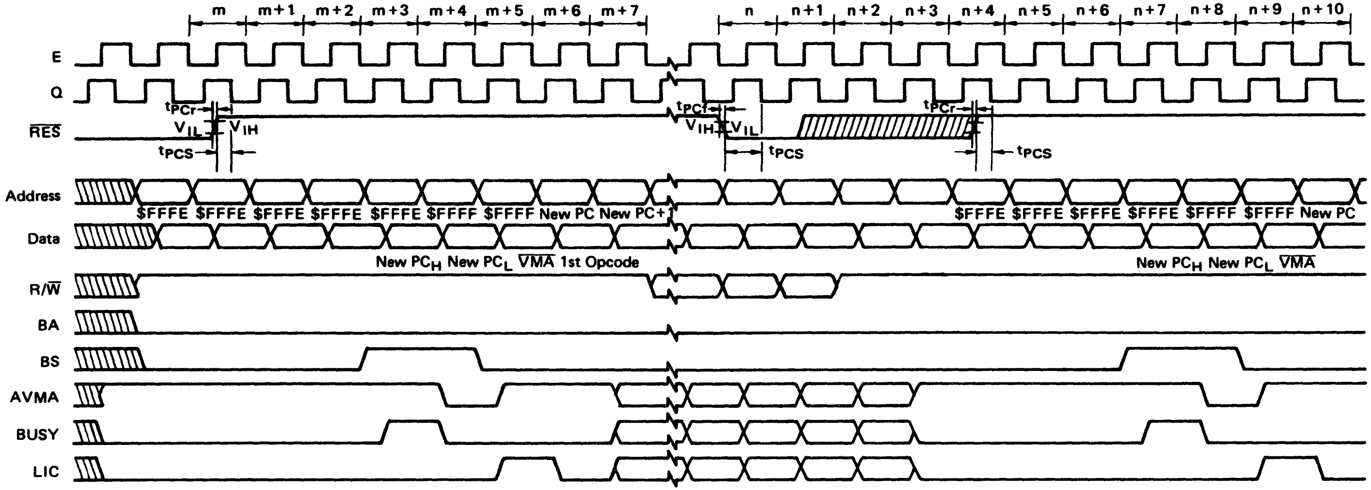
The Bus Status output signal, when decoded with BA, represents the MPU state (valid with leading edge of Q).

MPU State		MPU State Definition
BA	BS	
0	0	Normal (Running)
0	1	Interrupt or RESET Acknowledge
1	0	SYNC Acknowledge
1	1	HALT Acknowledge

Interrupt Acknowledge is indicated during both cycles of a hardware-vector-fetch ($\overline{\text{RES}}$, $\overline{\text{NMI}}$, $\overline{\text{FIRQ}}$, $\overline{\text{IRQ}}$, SWI, SWI2, SWI3). This signal, plus decoding of the lower four address lines, can provide the user with an indication of which interrupt level is being serviced and allow vectoring by device. See Table 1.

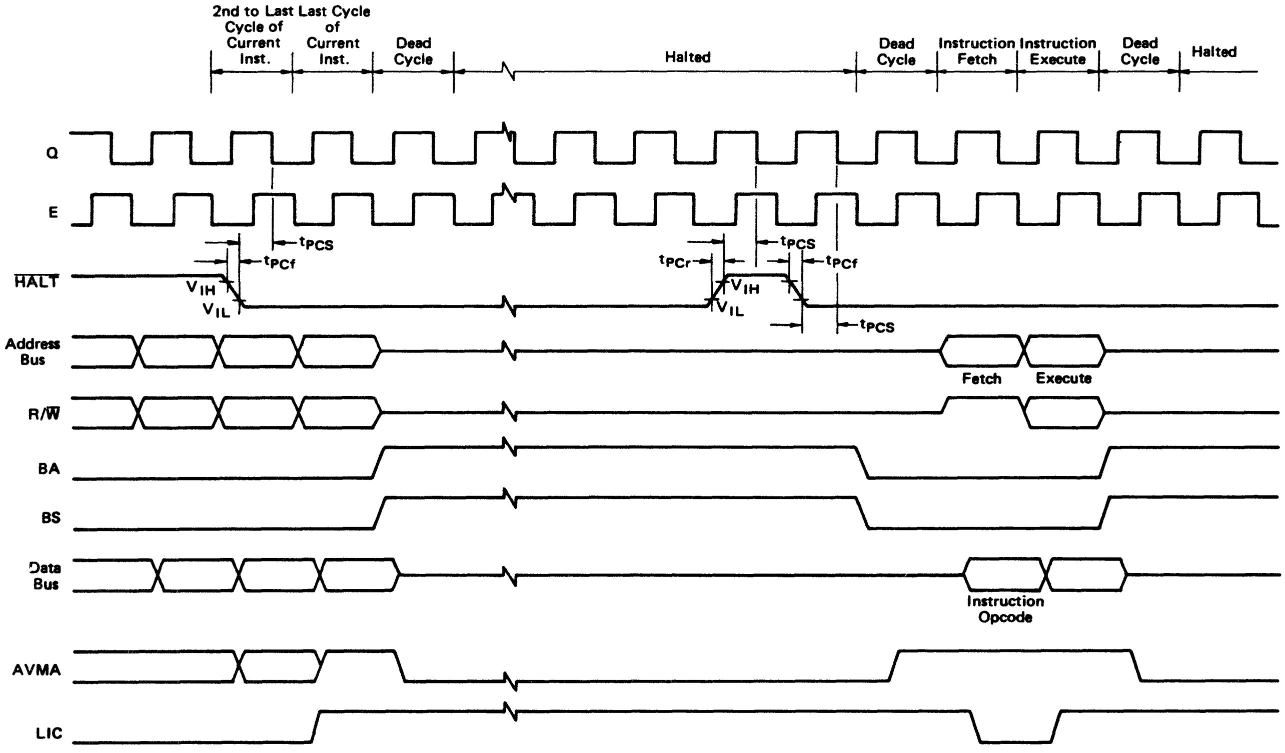
Sync Acknowledge is indicated while the MPU is waiting for external synchronization on an interrupt line.

Halt Acknowledge is indicated when the HD6809E is in a Halt condition.


m
m+1
m+2
m+3
m+4
m+5
m+6
m+7
n
n+1
n+2
n+3
n+4
n+5
n+6
n+7
n+8
n+9
n+10
E
Q
RES
Address
Data
R/W
BA
BS
AVMA
BUSY
LIC
tPCr
VIL
VIH
tPCS
tPCf
$FFFE $FFFE $FFFE $FFFE $FFFE $FFFF $FFFF New PC New PC+1
New PCH New PCL VMA 1st Opcode
$FFFE $FFFE $FFFE $FFFE $FFFF $FFFF New PC
New PCH New PCL VMA

(NOTE) Waveform measurements for all inputs and outputs are specified at logic "High" = V_{IHmin} and logic "Low" = V_{ILmax} unless otherwise specified.

Figure 7 $\overline{\text{RES}}$ Timing



(NOTE) Waveform measurements for all inputs and outputs are specified at logic "High" = V_{IHmin} and logic "Low" = V_{ILmax} unless otherwise specified.

Figure 8 $\overline{\text{HALT}}$ and Single Instruction Execution for System Debug

- **Non Maskable Interrupt ($\overline{\text{NMI}}$)***

A negative transition on this input requests that a non-maskable interrupt sequence be generated. A non-maskable interrupt cannot be inhibited by the program, and also has a higher priority than $\overline{\text{FIRQ}}$, $\overline{\text{IRQ}}$ or software interrupts. During recognition of an $\overline{\text{NMI}}$, the entire machine state is saved on the hardware stack. After reset, an $\overline{\text{NMI}}$ will not be recognized until the first program load of the Hardware Stack Pointer (S). The pulse width of $\overline{\text{NMI}}$ low must be at least one E cycle. If the $\overline{\text{NMI}}$ input does not meet the minimum set up with respect to Q, the interrupt will not be recognized until the next cycle. See Figure 9.

- **Fast-Interrupt Request ($\overline{\text{FIRQ}}$)***

A "Low" level on this input pin will initiate a fast interrupt sequence, provided its mask bit (F) in the CC is clear. This sequence has priority over the standard Interrupt Request ($\overline{\text{IRQ}}$), and is fast in the sense that it stacks only the contents of the condition code register and the program counter. The interrupt service routine should clear the source of the interrupt before doing an RTI. See Figure 10.

- **Interrupt Request ($\overline{\text{IRQ}}$)***

A "Low" level input on this pin will initiate an Interrupt Request sequence provided the mask bit (I) in the CC is clear. Since $\overline{\text{IRQ}}$ stacks the entire machine state it provides a slower response to interrupts than $\overline{\text{FIRQ}}$. $\overline{\text{IRQ}}$ also has a lower priority than $\overline{\text{FIRQ}}$. Again, the interrupt service routine should clear the source of the interrupt before doing an RTI. See Figure 9.

* $\overline{\text{NMI}}$, $\overline{\text{FIRQ}}$, and $\overline{\text{IRQ}}$ requests are sampled on the falling edge of Q. One cycle is required for synchronization before these interrupts are recognized. The pending interrupt(s) will not be serviced until completion of the current instruction unless a SYNC or CWAI condition is present. If $\overline{\text{IRQ}}$ and $\overline{\text{FIRQ}}$ do not remain "Low" until completion of the current instruction they may not be recognized. However, $\overline{\text{NMI}}$ is latched and need only remain "Low" for one cycle.

- **Clock Inputs E, Q**

E and Q are the clock signals required by the HD6809E. Q must lead E; that is, a transition on Q must be followed by a similar transition on E after a minimum delay. Addresses will be valid from the MPU, t_{AD} after the falling edge of E, and data will be latched from the bus by the falling edge of E. While the Q input is fully TTL compatible, the E input directly drives internal MOS circuitry and, thus, requires levels above normal TTL levels. This approach minimizes clock skew inherent with an internal buffer. Timing and waveforms for E and Q are shown in Figures 1 and 2 while Figure 11 shows a simple clock generator for the HD6809E.

- **BUSY**

Busy will be "High" for the read and modify cycles of a read-modify-write instruction and during the access of the first byte of a double-byte operation (e.g., LDX, STD, ADDD). Busy is also "High" during the first byte of any indirect or other vector fetch (e.g., jump extended, SWI indirect etc.).

In a multi-processor system, busy indicates the need to defer the rearbitration of the next bus cycle to insure the integrity of the above operations. This difference provides the indivisible memory access required for a "test-and-set" primitive, using any one of several read-modify-write instructions.

Busy does not become active during PSH or PUL operations. A typical read-modify-write instruction (ASL) is shown in Figure 12. Timing information is given in Figure 13. Busy is valid t_{CD} after the rising edge of Q.

- **AVMA**

AVMA is the Advanced VMA signal and indicates that the MPU will use the bus in the following bus cycle. The predictive nature of the AVMA signal allows efficient shared-bus multiprocessor systems. AVMA is "Low" when the MPU is in either a HALT or SYNC state. AVMA is valid t_{CD} after the rising edge of Q.

- **LIC**

LIC (Last Instruction Cycle) is "High" during the last cycle of every instruction, and its transition from "High" to "Low" will indicate that the first byte of an opcode will be latched at the end of the present bus cycle. LIC will be "High" when the MPU is Halted at the end of an instruction, (i.e., not in CWAI or RESET) in SYNC state or while stacking during interrupts. LIC is valid t_{CD} after the rising edge of Q.

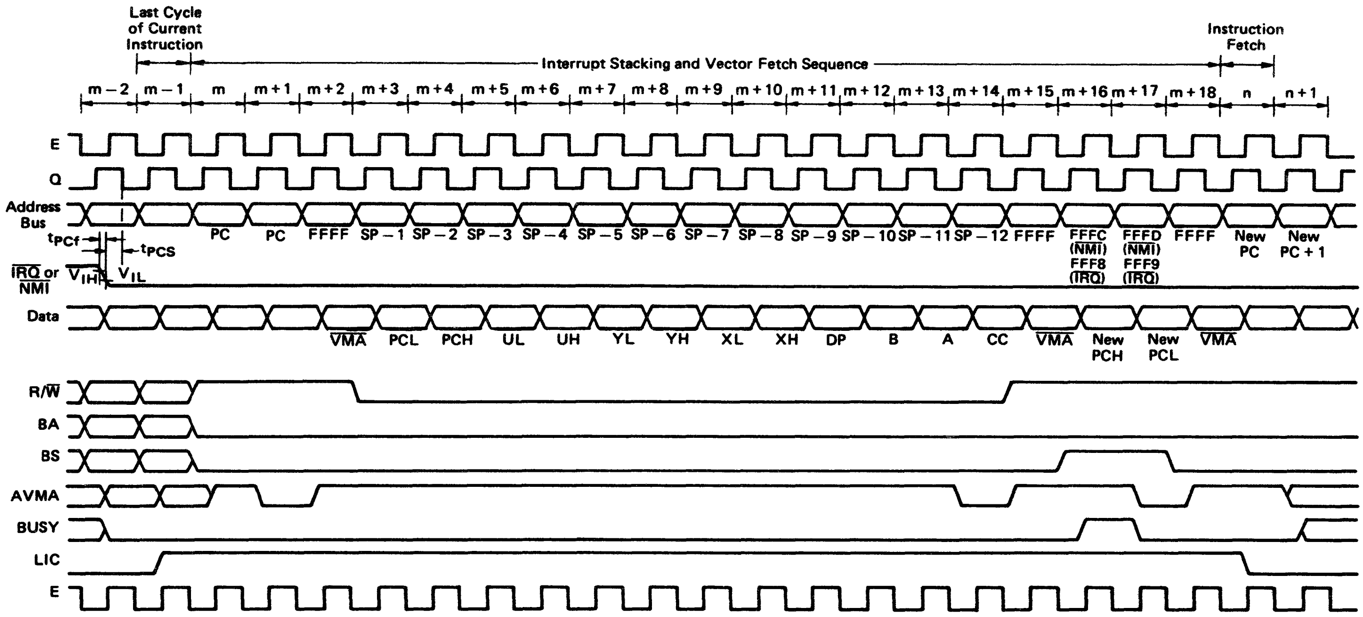
- **TSC**

TSC (Three-State Control) will cause MOS address, data, and R/$\overline{\text{W}}$ buffers to assume a high-impedance state. The control signals (BA, BS, BUSY, AVMA and LIC) will not go to the high-impedance state. TSC is intended to allow a single bus to be shared with other bus masters (processors or DMA controllers).

While E is "Low", TSC controls the address buffers and R/$\overline{\text{W}}$ directly. The data bus buffers during a write operation are in a high-impedance state until Q rises at which time, if TSC is true, they will remain in a high-impedance state. If TSC is held beyond the rising edge of E, then it will be internally latched, keeping the bus drivers in a high-impedance state for the remainder of the bus cycle. See Figure 14.

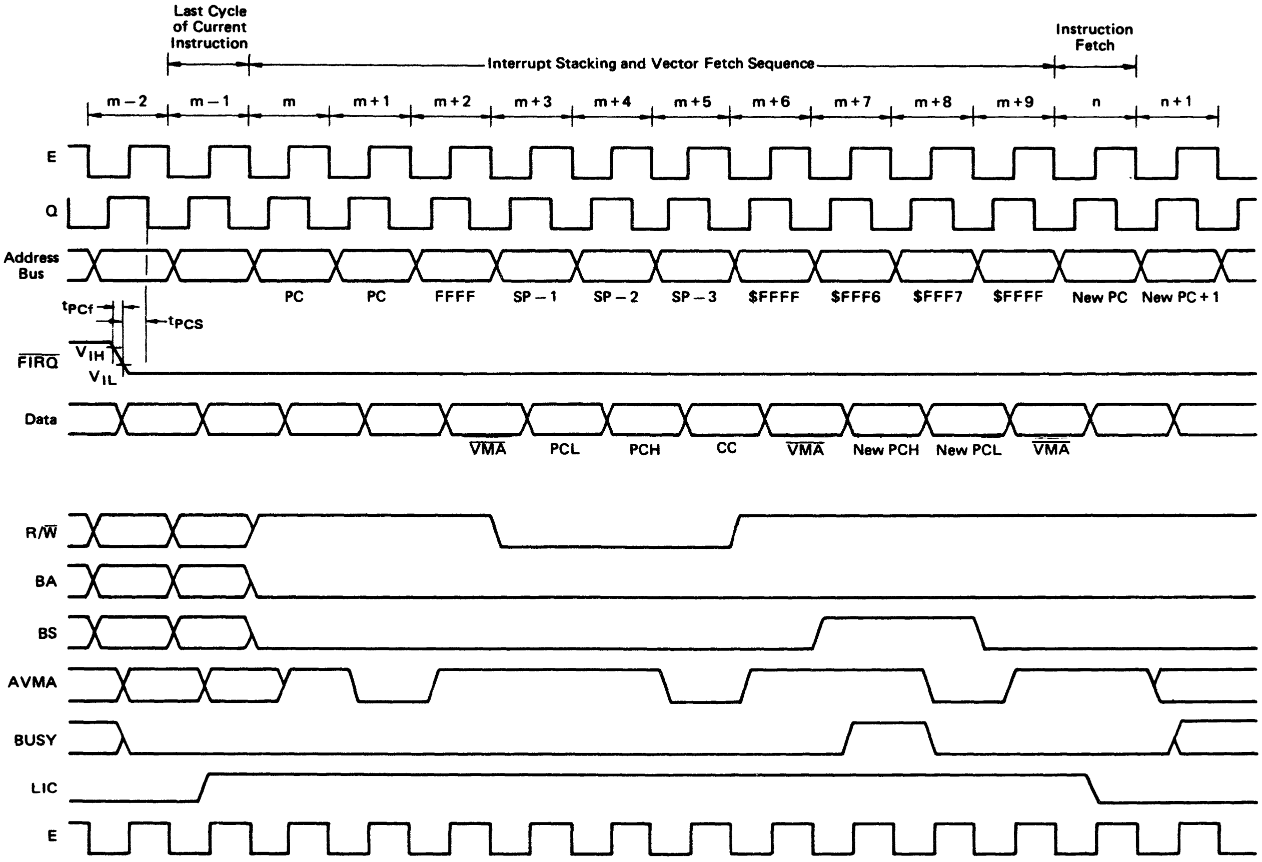
- **MPU Operation**

During normal operation, the MPU fetches an instruction from memory and then executes the requested function. This sequence begins after $\overline{\text{RES}}$ and is repeated indefinitely unless altered by a special instruction or hardware occurrence. Software instructions that alter normal MPU operation are: SWI, SWI2, SWI3, CWAI, RTI and SYNC. An interrupt or $\overline{\text{HALT}}$ input can also alter the normal execution of instructions. Figure 15 illustrates the flow chart for the HD6809E.



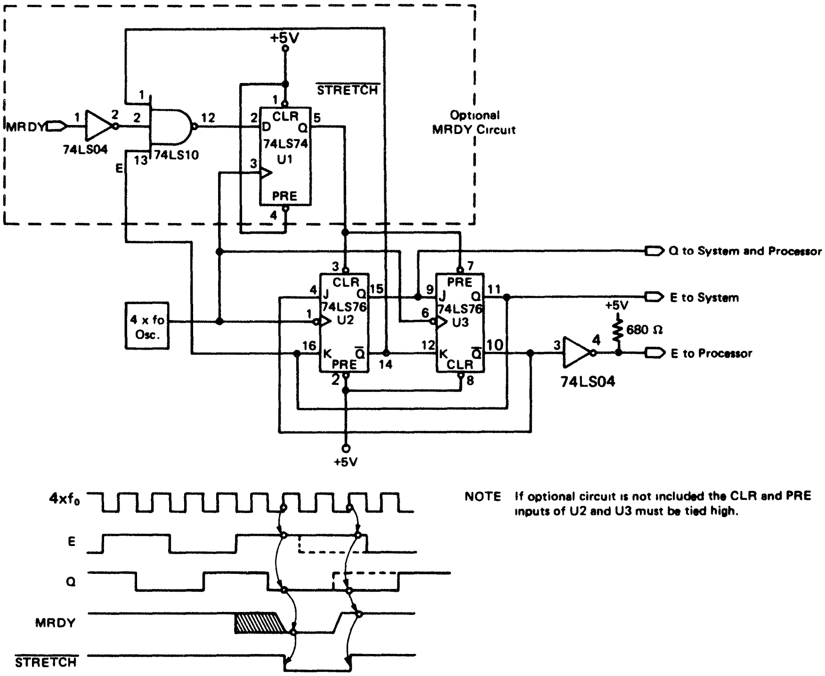
(NOTE) Waveform measurements for all inputs and outputs are specified at logic "High" = V_{IHmin} and logic "Low" = V_{ILmax} unless otherwise specified. E clock shown for reference only.

Figure 9 $\overline{\text{IRQ}}$ and $\overline{\text{NMI}}$ Interrupt Timing



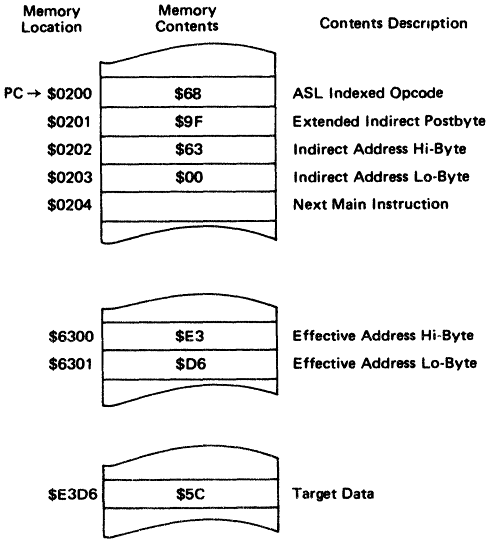
(NOTE) Waveform measurements for all inputs and outputs are specified at logic "High" = V_{IHmin} and logic "Low" = V_{ILmax} unless otherwise specified. E clock shown for reference only.

Figure 10 $\overline{\text{FIRQ}}$ Interrupt Timing



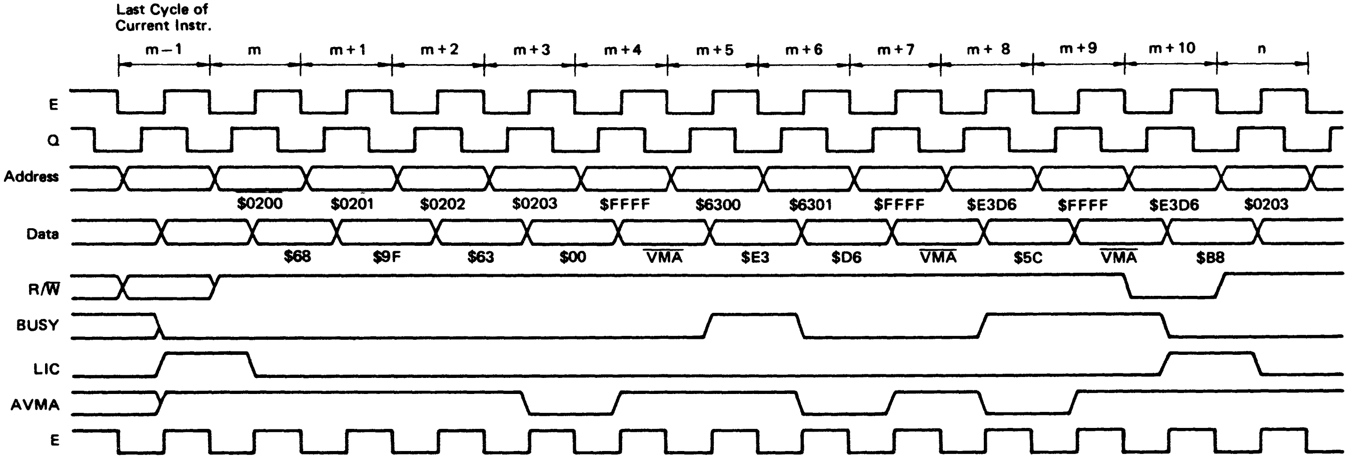
NOTE If optional circuit is not included the CLR and PRE inputs of U2 and U3 must be tied high.

Figure 11 HD6809E Clock Generator



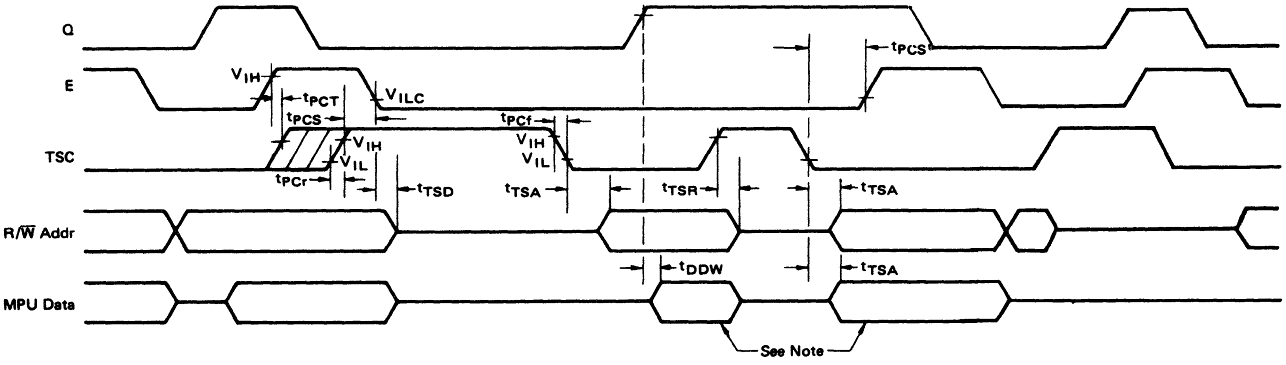
Memory Location	Memory Contents	Contents Description
PC → $0200	$68	ASL Indexed Opcode
$0201	$9F	Extended Indirect Postbyte
$0202	$63	Indirect Address Hi-Byte
$0203	$00	Indirect Address Lo-Byte
$0204		Next Main Instruction
$6300	$E3	Effective Address Hi-Byte
$6301	$D6	Effective Address Lo-Byte
$E3D6	$5C	Target Data

Figure 12 Read Modify Write Instruction Example (ASL Extended Indirect)



(NOTE) Waveform measurements for all inputs and outputs are specified at logic "High" = V_{IHmin} and logic "Low" = V_{ILmax} unless otherwise specified.

Figure 13 BUSY Timing (ASL Extended Indirect Instruction)



(NOTES) Data will be asserted by the MPU only during the interval while R/$\overline{W}$ is "Low" and E or Q is "High".
Waveform measurements for all inputs and outputs are specified at logic "High" = V_{IHmin} and logic "Low" = V_{ILmax} unless otherwise specified.

Figure 14 TSC Timing

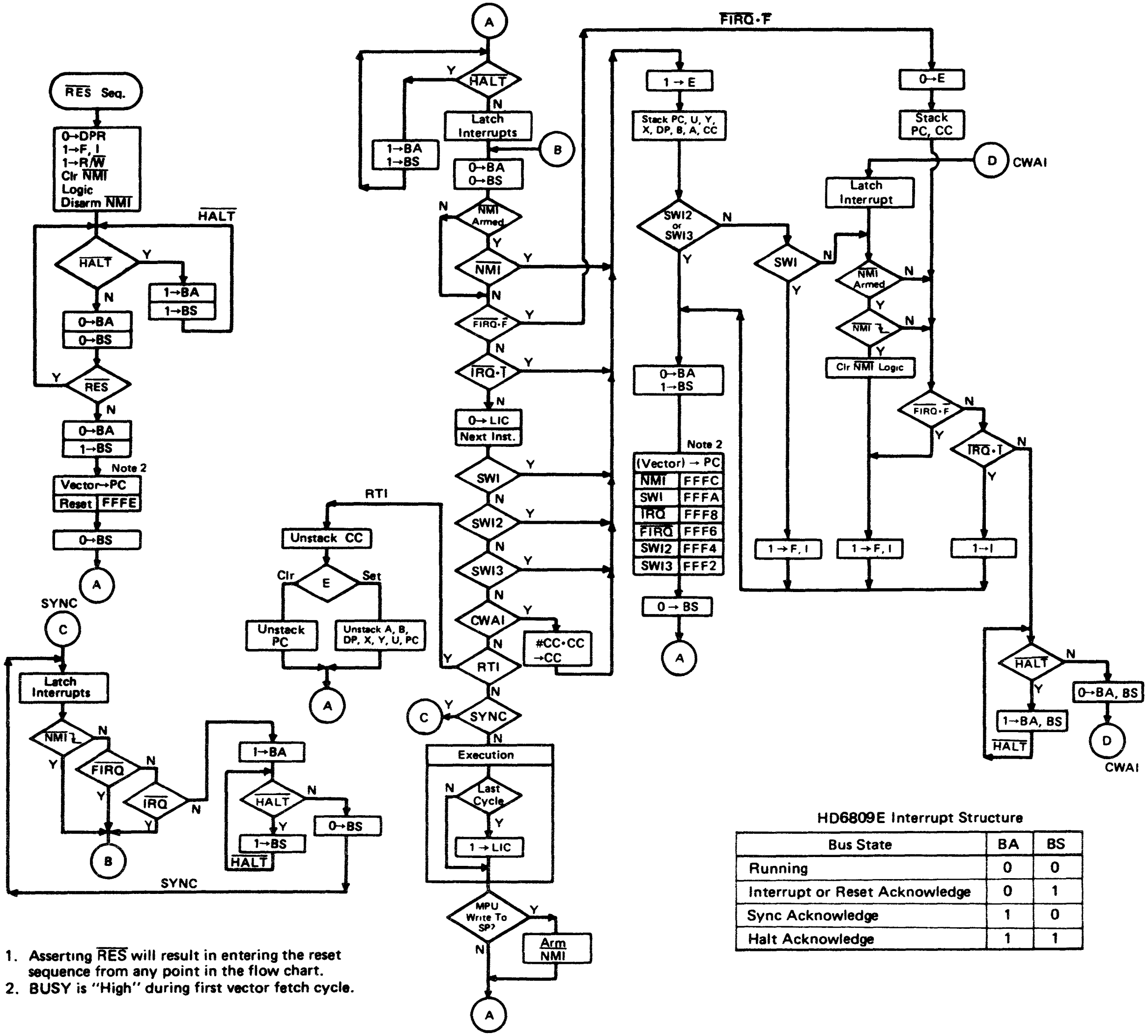


Figure 15 Flowchart for HD6809E Instruction

HD6809E Interrupt Structure

Bus State	BA	BS
Running	0	0
Interrupt or Reset Acknowledge	0	1
Sync Acknowledge	1	0
Halt Acknowledge	1	1

(NOTES) 1. Asserting $\overline{\text{RES}}$ will result in entering the reset sequence from any point in the flow chart.

2. BUSY is "High" during first vector fetch cycle.

■ ADDRESSING MODES

The basic instructions of any computer are greatly enhanced by the presence of powerful addressing modes. The HD6809E has the most complete set of addressing modes available on any microcomputer today. For example, the HD6809E has 59 basic instructions; however, it recognizes 1464 different variations of instructions and addressing modes. The addressing modes support modern programming techniques. The following addressing modes are available on the HD6809E:

(1) Implied (Includes Accumulator)
(2) Immediate
(3) Extended
(4) Extended Indirect
(5) Direct
(6) Register
(7) Indexed
  Zero-Offset
  Constant Offset
  Accumulator Offset
  Auto Increment/Decrement
(8) Indexed Indirect
(9) Relative
(10) Program Counter Relative

● Implied (Includes Accumulator)

In this addressing mode, the opcode of the instruction contains all the address information necessary. Examples of Implied Addressing are: ABX, DAA, SWI, ASRA, and CLRB.

● Immediate Addressing

In Immediate Addressing, the effective address of the data is the location immediately following the opcode (i.e., the data to be used in the instruction immediately follows the opcode of the instruction). The HD6809E uses both 8 and 16-bit immediate values depending on the size of argument specified by the opcode. Examples of instructions with immediate Addressing are:

```
LDA  #$20
LDX  #$F000
LDY  #CAT
```

(NOTE) # signifies immediate addressing, $ signifies hexadecimal value.

● Extended Addressing

In Extended Addressing, the contents of the two bytes immediately following the opcode fully specify the 16-bit effective address used by the instruction. Note that the address generated by an extended instruction defines an absolute address and is not position independent. Examples of Extended Addressing include:

```
LDA  CAT
STX  MOUSE
LDD  $2000
```

● Extended Indirect

As a special case of indexed addressing (discussed below), one level of indirection may be added to Extended Addressing. In Extended Indirect, the two bytes following the postbyte of an Indexed instruction contain the address of the data.

```
LDA  [CAT]
LDX  [$FFFE]
STU  [DOG]
```

● Direct Addressing

Direct addressing is similar to extended addressing except that only one byte of address follows the opcode. This byte specifies the lower 8 bits of the address to be used. The upper 8 bits of the addres.. are supplied by the direct page register. Since only one byte of address is required in direct addressing, this mode requires less memory and executes faster than extended addressing. Of course, only 256 locations (one page) can be accessed without redefining the contents of the DP register. Since the DP register is set to $00 on Reset, direct addressing on the HD6809E is compatible with direct addressing on the HD6800. Indirection is not allowed in direct addressing. Some examples of direct addressing are:

```
LDA    $30
SETDP  $10     (Assembler directive)
LDB    $1030
LDD    <CAT
```

(NOTE) < is an assembler directive which forces direct addressing.

● Register Addressing

Some opcodes are followed by a byte that defines a register or set of registers to be used by the instruction. This is called a postbyte. Some examples of register addressing are:

```
TFR   X, Y        Transfer X into Y
EXG   A, B        Exchanges A with B
PSHS  A, B, X, Y  Push Y, X, B and A onto S
PULU  X, Y, D     Pull D, X, and Y from U
```

● Indexed Addressing

In all indexed addressing, one of the pointer registers (X, Y, U, S, and sometimes PC) is used in a calculation of the effective address of the operand to be used by the instruction. Five basic types of indexing are available and are discussed below. The postbyte of an indexed instruction specifies the basic type and variation of the addressing mode as well as the pointer register to be used. Figure 16 lists the legal formats for the postbyte. Table 2 gives the assembler form and the number of cycles and bytes added to the basic values for indexed addressing for each variation.

Post-Byte Register Bit								Indexed Addressing Mode
7	6	5	4	3	2	1	0	
0	R	R	d	d	d	d	d	EA = ,R + 5 Bit Offset
1	R	R	0	0	0	0	0	,R +
1	R	R	i	0	0	0	1	,R + +
1	R	R	0	0	0	1	0	, -R
1	R	R	i	0	0	1	1	, - - R
1	R	R	i	0	1	0	0	EA = ,R + 0 Offset
1	R	R	i	0	1	0	1	EA = ,R + ACCB Offset
1	R	R	i	0	1	1	0	EA = ,R + ACCA Offset
1	R	R	i	1	0	0	0	EA = , R + 8 Bit Offset
1	R	R	i	1	0	0	1	EA = ,R + 16 Bit Offset
1	R	R	i	1	0	1	1	EA = ,R + D Offset
1	x	x	i	1	1	0	0	EA = ,PC + 8 Bit Offset
1	x	x	i	1	1	0	1	EA = ,PC + 16 Bit Offset
1	R	R	i	1	1	1	1	EA = [,Address]

Addressing Mode Field

Indirect Field
(Sigh bit when b7 = 0)

Register Field RR
00 = X
01 = Y
10 = U
11 = S

x = Don't Care
d = Offset Bit
i = { 0 = Non Indirect
  1 = Indirect

Figure 16 Index Addressing Postbyte Register Bit Assignments

Table 2 Indexed Addressing Mode

Type	Forms	Non Indirect				Indirect			
		Assembler Form	Postbyte OP Code	$\overset{+}{\sim}$	$\overset{+}{\#}$	Assembler Form	Postbyte OP Code	$\overset{+}{\sim}$	$\overset{+}{\#}$
Constant Offset From R (2's Complement Offsets)	No Offset	,R	1RR00100	0	0	[,R]	1RR10100	3	0
	5 Bit Offset	n, R	0RRnnnnn	1	0	defaults to 8-bit			
	8 Bit Offset	n, R	1RR01000	1	1	[n, R]	1RR11000	4	1
	16 Bit Offset	n, R	1RR01001	4	2	[n, R]	1RR11001	7	2
Accumulator Offset From R (2's Complement Offsets)	A Register Offset	A, R	1RR00110	1	0	[A, R]	1RR10110	4	0
	B Register Offset	B, R	1RR00101	1	0	[B, R]	1RR10101	4	0
	D Register Offset	D, R	1RR01011	4	0	[D, R]	1RR11011	7	0
Auto Increment/Decrement R	Increment By 1	,R +	1RR00000	2	0	not allowed			
	Increment By 2	,R + +	1RR00001	3	0	[,R + +]	1RR10001	6	0
	Decrement By 1	, - R	1RR00010	2	0	not allowed			
	Decrement By 2	, - - R	1RR00011	3	0	[, - - R]	1RR10011	6	0
Constant Offset From PC (2's Complement Offsets)	8 Bit Offset	n, PCR	1xx01100	1	1	[n, PCR]	1xx11100	4	1
	16 Bit Offset	n, PCR	1xx01101	5	2	[n, PCR]	1xx11101	8	2
Extended Indirect	16 Bit Address	—	—	—	—	[n]	10011111	5	2

R = X, Y, U or S
x = Don't Care

RR:
00 = X
01 = Y
10 = U
11 = S

$\overset{+}{\sim}$ and $\overset{+}{\#}$ indicate the number of additional cycles and bytes for the particular variation.

Zero-Offset Indexed

In this mode, the selected pointer register contains the effective address of the data to be used by the instruction. This is the fastest indexing mode.

Examples are:

```
LDD   0, X
LDA   S
```

Constant Offset Indexed

In this mode, a two's-complement offset and the contents of one of the pointer registers are added to form the effective address of the operand. The pointer register's initial content is unchanged by the addition.

Three sizes of offsets are available:
5-bit (−16 to +15)
8-bit (−128 to +127)
16-bit (−32768 to +32767)

The two's complement 5-bit offset is included in the postbyte and, therefore, is most efficient in use of bytes and cycles. The two's complement 8-bit offset is contained in a single byte following the postbyte. The two's complement 16-bit offset is in the two bytes following the postbyte. In most cases the programmer need not be concerned with the size of this offset since the assembler will select the optimal size automatically.

Examples of constant-offset indexing are:

```
LDA   23, X
LDX   −2, S
LDY   300, X
LDU   CAT, Y
```

Accumulator-Offset Indexed

This mode is similar to constant offset indexed except that the two's-complement value in one of the accumulators (A, B or D) and the contents of one of the pointer registers are added to form the effective address of the operand. The contents of both the accumulator and the pointer register are unchanged by the addition. The postbyte specifies which accumulator to use as an offset and no additional bytes are required. The advantage of an accumulator offset is that the value of the offset can be calculated by a program at run-time.

Some examples are:

```
LDA    B, Y
LDX    D, Y
LEAX   B, X
```

Auto Increment/Decrement Indexed

In the auto increment addressing mode, the pointer register contains the address of the operand. Then, after the pointer register is used it is incremented by one or two. This addressing mode is useful in stepping through tables, moving data, or for the creation of software stacks. In auto decrement, the pointer register is decremented prior to use as the address of the data. The use of auto decrement is similar to that of auto increment; but the tables, etc., are scanned from the high to low addresses. The size of the increment/decrement can be either one or two to allow for tables of either 8- or 16-bit data to be accessed and is selectable by the programmer. The predecrement, post-increment nature of these modes allow them to be used to create additional software stacks that behave identically to the U and S stacks.

Some examples of the auto increment/decrement addressing modes are:

```
LDA    ,X+
STD    ,Y++
LDB    ,-Y
LDX    ,--S
```

Care should be taken in performing operations on 16-bit pointer registers (X, Y, U, S) where the same register is used to calculate the effective address.

Consider the following instruction:

STX 0, X + + (X initialized to 0)

The desired result is to store a 0 in locations $0000 and $0001 then increment X to point to $0002. In reality, the following occurs:

0 → temp	calculate the EA; temp is a holding register
X + 2 → X	perform autoincrement
X → (temp)	do store operation

- **Indexed Indirect**

All of the indexing modes with the exception of auto increment/decrement by one, or a ±4-bit offset may have an additional level of indirection specified. In indirect addressing, the effective address is contained at the location specified by the contents of the Index Register plus any offset. In the example below, the A accumulator is loaded indirectly using an effective address calculated from the Index Register and an offset.

```
        Before Execution
        A = XX (don't care)
        X = $F000
$0100   LDA [$10, X]        EA is now $F010
$F010   $F1                 $F150 is now the
$F011   $50                 new EA
$F150   $AA
        After Execution
        A = $AA (Actual Data Loaded)
        X = $F000
```

All modes of indexed indirect are included except those which are meaningless (e.g., auto increment/decrement by 1 indirect). Some examples of indexed indirect are:

```
LDA    [,X]
LDD    [10,S]
LDA    [B,Y]
LDD    [,X++]
```

- **Relative Addressing**

The byte(s) following the branch opcode is (are) treated as a signed offset which may be added to the program counter. If the branch condition is true then the calculated address (PC + signed offset) is loaded into the program counter. Program execution continues at the new location as indicated by the PC; short (1 byte offset) and long (2 bytes offset) relative addressing modes are available. All of memory can be reached in long relative addressing as an effective address is interpreted modulo 2^{16}. Some examples of relative addressing are:

```
        BEQ      CAT        (short)
        BGT      DOG        (short)
CAT     LBEQ     RAT        (long)
DOG     LBGT     RABBIT     (long)
          •
          •
          •
RAT     NOP
RABBIT  NOP
```

- **Program Counter Relative**

The PC can be used as the pointer register with 8 or 16-bit signed offsets. As in relative addressing, the offset is added to the current PC to create the effective address. The effective address is then used as the address of the operand or data. Program Counter Relative Addressing is used for writing position independent programs. Tables related to a particular routine will maintain the same relationship after the routine is moved, if referenced relative to the Program Counter. Examples are:

```
LDA    CAT, PCR
LEAX   TABLE, PCR
```

Since program counter relative is a type of indexing, an additional level of indirection is available.

```
LDA    [CAT, PCR]
LDU    [DOG, PCR]
```

■ HD6809E INSTRUCTION SET

The instruction set of the HD6809E is similar to that of the HD6800 and is upward compatible at the source code level. The number of opcodes has been reduced from 72 to 59, but because of the expanded architecture and additional addressing modes, the number of available opcodes (with different addressing modes) has risen from 197 to 1464.

Some of the new instructions are described in detail below:

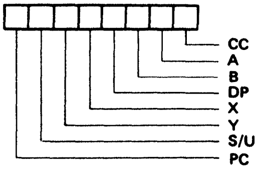
- **PSHU/PSHS**

The push instructions have the capability of pushing onto either the hardware stack (S) or user stack (U) any single register, or set of registers with a single instruction.

- **PULU/PULS**

The pull instructions have the same capability of the push instruction, in reverse order. The byte immediately following the push or pull opcode determines which register or registers are to be pushed or pulled. The actual PUSH/PULL sequence is fixed; each bit defines a unique register to push or pull, as shown in below.

PUSH/PULL POST BYTE



	← Pull Order					Push Order →	
PC	U	Y	X	DP	B	A	CC
FFFF	← increasing memory address						0000
PC	S	Y	X	DP	B	A	CC

- **TFR/EXG**

Within the HD6809E, any register may be transferred to or exchanged with another of like-size; i.e., 8-bit to 8-bit or 16-bit to 16-bit. Bits 4~7 of postbyte define the source register, while bits 0~3 represent the destination register. These are denoted as follows:

0000 – D	0101 – PC
0001 – X	1000 – A
0010 – Y	1001 – B
0011 – U	1010 – CC
0100 – S	1011 – DP

(NOTE) All other combinations are undefined and INVALID.

TRANSFER/EXCHANGE POST BYTE

SOURCE	DESTINATION

- **LEAX/LEAY/LEAU/LEAS**

The LEA (Load Effective Address) works by calculating the effective address used in an indexed instruction and stores that address value, rather than the data at that address, in a pointer register. This makes all the features of the internal addressing hardware available to the programmer. Some of the implications of this instruction are illustrated in Table 3.

The LEA instruction also allows the user to access data in a position independent manner. For example:

```
        LEAX    MSG1, PCR
        LBSR    PDATA (Print message routine)
        •
        •
MSG1    FCC     'MESSAGE'
```

This sample program prints: 'MESSAGE'. By writing MSG1, PCR, the assembler computes the distance between the present address and MSG1. This result is placed as a constant into the LEAX instruction which will be indexed from the PC value at the time of execution. No matter where the code is located, when it is executed, the computed offset from the PC will put the absolute address of MSG1 into the X pointer register. This code is totally position independent.

The LEA instructions are very powerful and use an internal holding register (temp). Care must be exercised when using the LEA instructions with the autoincrement and autodecrement addressing modes due to the sequence of internal operations. The LEA internal sequence is outlined as follows:

```
LEAa, b+      (any of the 16-bit pointer registers X, Y, U
              or S may be substituted for a and b.)
1. b → temp   (calculate the EA)
2. b + 1 → b  (modify b, postincrement)
3. temp → a   (load a)

LEAa, – b
1. b – 1 → temp  (calculate EA with predecrement)
2. b – 1 → b     (modify b, predecrement)
3. temp → a      (load a)
```

Autoincrement-by-two and autodecrement-by-two instructions work similarly. Note that LEAX, X+ does not change X, however LEAX, –X does decrement X. LEAX 1, X should be used to increment X by one.

Table 3 LEA Examples

Instruction	Operation	Comment
LEAX 10, X	X + 10 → X	Adds 5-bit constant 10 to X
LEAX 500, X	X + 500 → X	Adds 16-bit constant 500 to X
LEAY A, Y	Y + A → Y	Adds 8-bit A accumulator to Y
LEAY D, Y	Y + D → Y	Adds 16-bit D accumulator to Y
LEAU –10, U	U – 10 → U	Subtracts 10 from U
LEAS –10, S	S – 10 → S	Used to reserve area on stack
LEAS 10, S	S + 10 → S	Used to 'clean up' stack
LEAX 5, S	S + 5 → X	Transfers as well as adds

- **MUL**

Multiplies the unsigned binary numbers in the A and B accumulator and places the unsigned result into the 16-bit D accumulator. This unsigned multiply also allows multiple-precision multiplications.

Long and Short Relative Branches

The HD6809E has the capability of program counter relative branching throughout the entire memory map. In this mode, if the branch is to be taken, the 8 or 16-bit signed offset is added to the value of the program counter to be used as the effective address. This allows the program to branch anywhere in the 64k memory map. Position independent code can be easily generated through the use of relative branching. Both short (8-bit) and long (16-bit) branches are available.

- **SYNC**

After encountering a Sync instruction, the MPU enters a Sync state, stops processing instructions and waits for an interrupt. If the pending interrupt is non-maskable ($\overline{\text{NMI}}$) or maskable ($\overline{\text{FIRQ}}$, $\overline{\text{IRQ}}$) with its mask bit (F or I) clear, the processor will clear the Sync state and perform the normal interrupt stacking and service routine. Since $\overline{\text{FIRQ}}$ and $\overline{\text{IRQ}}$ are not edge-triggered, a low level with a minimum duration of three bus cycles is required to assure that the interrupt will be taken. If the pending interrupt is maskable ($\overline{\text{FIRQ}}$, $\overline{\text{IRQ}}$) with its mask bit (F or I) set, the processor will clear the Sync state and continue processing by executing the next inline instruction. Figure 17 depicts Sync timing.

Software Interrupts

A Software Interrupt is an instruction which will cause an interrupt, and its associated vector fetch. These Software Interrupts are useful in operating system calls, software debugging, trace operations, memory mapping, and software development systems. Three levels of SWI are available on this HD6809E, and are prioritized in the following order: SWI, SWI2, SWI3.

16-Bit Operation

The HD6809E has the capability of processing 16-bit data. These instructions include loads, stores, compares, adds, subtracts, transfers, exchanges, pushes and pulls.

- **CYCLE-BY-CYCLE OPERATION**

The address bus cycle-by-cycle performance chart illustrates the memory-access sequence corresponding to each possible instruction and addressing mode in the HD6809E. Each instruction begins with an opcode fetch. While that opcode is being internally decoded, the next program byte is always fetched. (Most instructions will use the next byte, so this technique considerably speeds throughput.) Next, the operation of each opcode will follow the flow chart. $\overline{\text{VMA}}$ is an indication of FFFF_{16} on the address bus, R/$\overline{\text{W}}$ = "High" and BS = "Low". The following examples illustrate the use of the chart; see Figure 18.

Example 1: LBSR (Branch Taken)
Before Execution SP = F000

```
                         •
                         •
                         •
$8000                    LBSR       CAT
                         •
                         •
                         •
$A000     CAT            •
```

CYCLE-BY-CYCLE FLOW

Cycle #	Address	Data	R/$\overline{\text{W}}$	Description
1	8000	17	1	Opcode Fetch
2	8001	1F	1	Offset High Byte
3	8002	FD	1	Offset Low Byte
4	FFFF	*	1	$\overline{\text{VMA}}$ Cycle
5	FFFF	*	1	$\overline{\text{VMA}}$ Cycle
6	FFFF	*	1	$\overline{\text{VMA}}$ Cycle
7	FFFF	*	1	$\overline{\text{VMA}}$ Cycle
8	EFFF	03	0	Stack Low Order Byte of Return Address
9	EFFE	80	0	Stack High Order Byte of Return Address

Example 2: DEC (Extended)

```
$8000          DEC          $A000
$A000          FCB          $80
```

CYCLE-BY-CYCLE FLOW

Cycle #	Address	Data	R/$\overline{\text{W}}$	Description
1	8000	7A	1	Opcode Fetch
2	8001	A0	1	Operand Address, High Byte
3	8002	00	1	Operand Address, Low Byte
4	FFFF	*	1	$\overline{\text{VMA}}$ Cycle
5	A000	80	1	Read the Data
6	FFFF	*	1	$\overline{\text{VMA}}$ Cycle
7	A000	7F	0	Store the Decremented Data

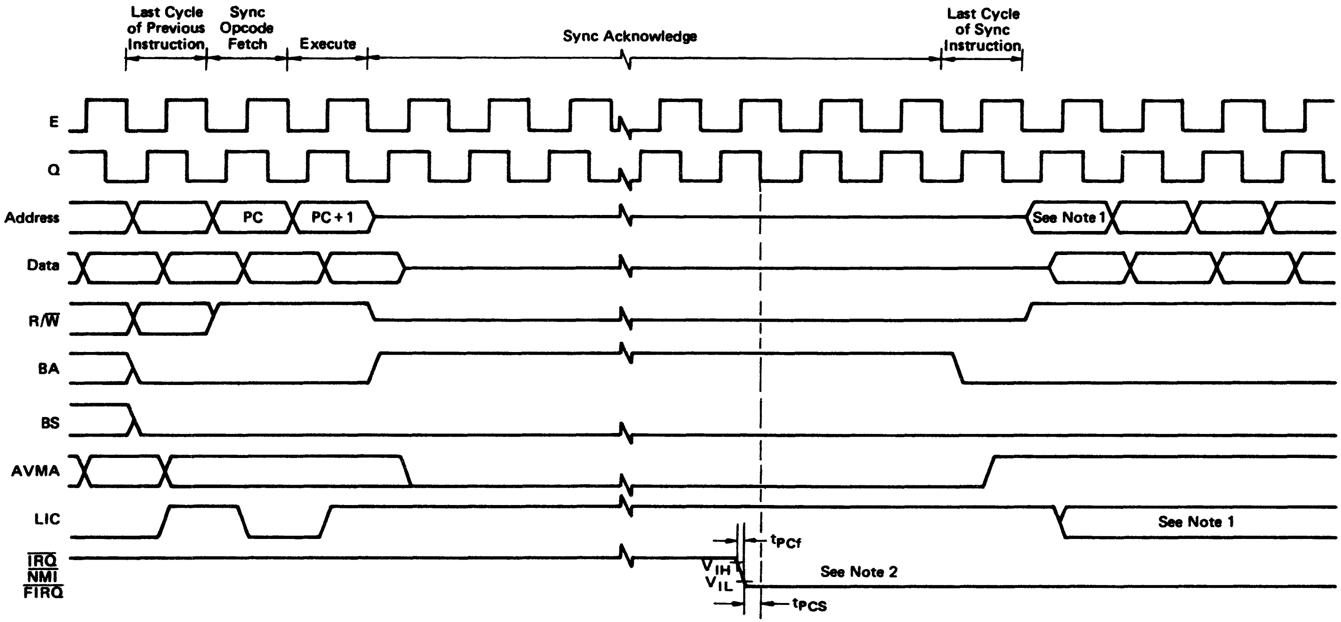
* The data bus has the data at that particular address.

■ HD6809E INSTRUCTION SET TABLES

The instructions of the HD6809E have been broken down into five different categories. They are as follows:

8-Bit operation (Table 4)
16-Bit operation (Table 5)
Index register/stack pointer instructions (Table 6)
Relative branches (long or short) (Table 7)
Miscellaneous instructions (Table 8)

HD6809E instruction set tables and Hexadecimal Values of instructions are shown in Table 9 and Table 10.

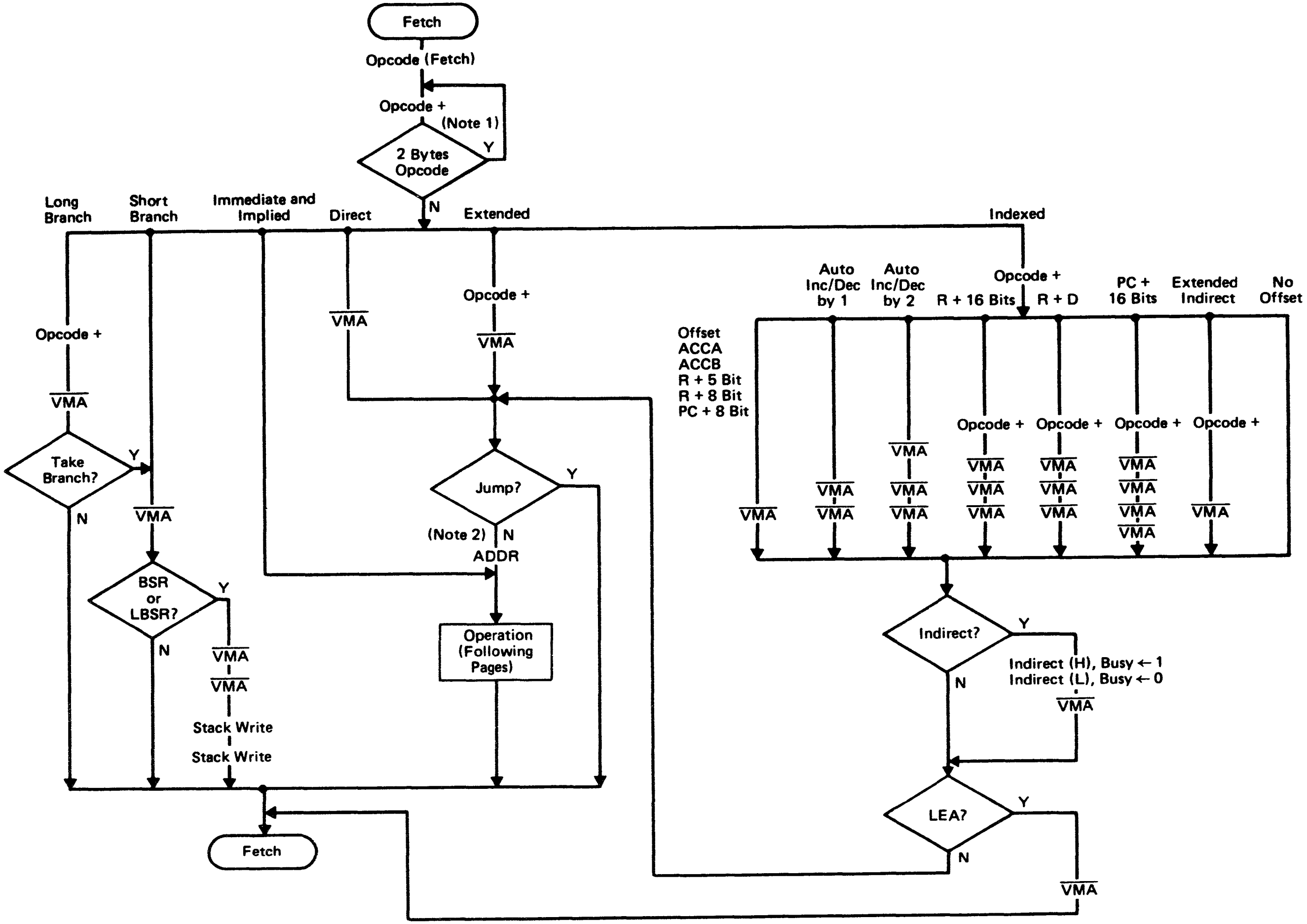


(NOTES) 1. If the associated mask bit is set when the interrupt is requested, LIC will go "Low" and this cycle will be an instruction fetch from address location PC + 1. However, if the interrupt is accepted ($\overline{NMI}$ or an unmasked $\overline{FIRQ}$ or $\overline{IRQ}$) LIC will remain "High" and interrupt processing will start with this cycle as (m) on Figure 9 and 10 (Interrupt Timing).

2. If mask bits are clear, $\overline{IRQ}$ and $\overline{FIRQ}$ must be held "Low" for three cycles to guarantee that interrupt will be taken, although only one cycle is necessary to bring the processor out of SYNC.

3. Waveform measurements for all inputs and outputs are specified at logic "High" = V_{IHmin} and logic "Low" = V_{ILmax} unless otherwise specified.

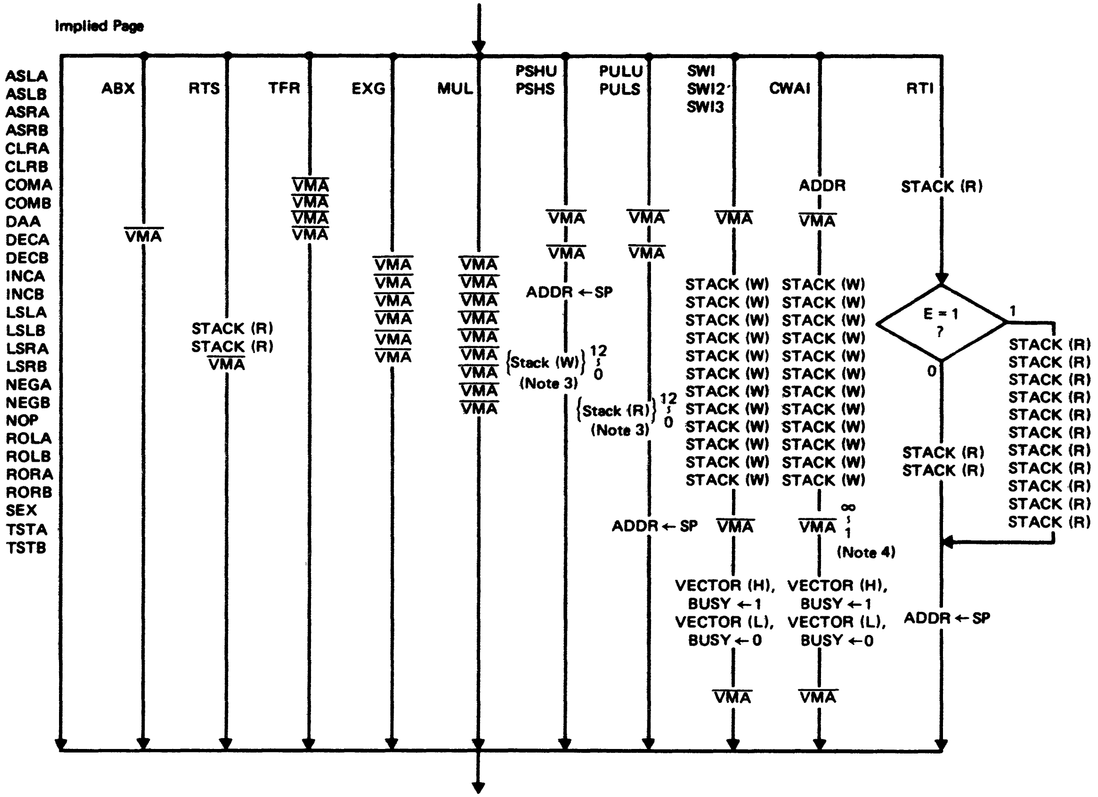
Figure 17 SYNC Timing



(NOTE)

1. Busy = "High" during access of first byte of double byte immediate load.
2. Write operation during store instruction. Busy = "High" during first two cycles of a double-byte access and the first cycle of read-modify-write access.
3 AVMA is asserted on the cycle before a $\overline{\text{VMA}}$ cycle.

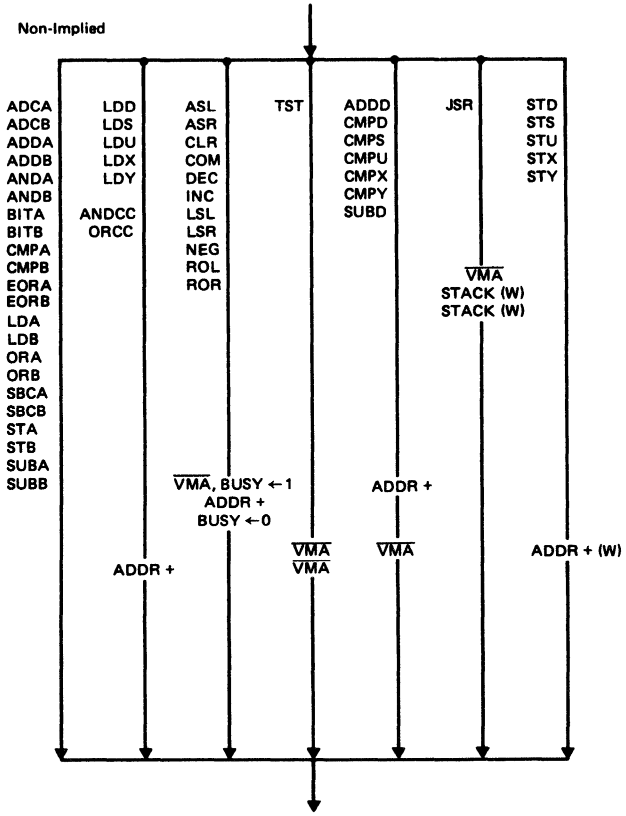
Figure 18 Address Bus Cycle-by-Cycle Performance



(NOTES)

1. Stack (W) refers to the following sequence: SP ← SP − 1, then ADDR ← SP with R/$\overline{W}$ = "Low"
   Stack (R) refers to the following sequence: ADDR ← SP with R/$\overline{W}$ = "High", then SP ← SP + 1.
   PSHU, PULU instructions use the user stack pointer (i.e., SP = U) and PSHS, PULS use the hardware stack pointer (i.e., SP = S).
2. Vector refers to the address of an interrupt or reset vector (see Table 1).
3. The number of stack accesses will vary according to the number of bytes saved.
4. $\overline{VMA}$ cycles will occur until an interrupt occurs.

Figure 18 Address Bus Cycle-by-Cycle Performance (Continued)



(NOTES)

1 Stack (W) refers to the following sequence SP ← SP — 1, then ADDR ← SP with R/$\overline{W}$ = "Low"
Stack (R) refers to the following sequence ADDR ← with R/$\overline{W}$ = "High", then SP ← SP + 1
PSHU, PULU instructions use the user stack pointer (i.e., SP = U) and PSHS, PULS use the hardware stack pointer (i.e., SP = S)

2. Vector refers to the address of an interrupt or reset vector (see Table 1)

3 The number of stack accesses will vary according to the number of bytes saved.

4 $\overline{VMA}$ cycles will occur until an interrupt occurs.

Figure 18 Address Bus Cycle-by-Cycle Performance (Continued)

Table 4 8-Bit Accumulator and Memory Instructions

Mnemonic(s)	Operation
ADCA, ADCB	Add memory to accumulator with carry
ADDA, ADDB	Add memory to accumulator
ANDA, ANDB	And memory with accumulator
ASL, ASLA, ASLB	Arithmetic shift of accumulator or memory left
ASR, ASRA, ASRB	Arithmetic shift of accumulator or memory right
BITA, BITB	Bit test memory with accumulator
CLR, CLRA, CLRB	Clear accumulator or memory location
CMPA, CMPB	Compare memory from accumulator
COM, COMA, COMB	Complement accumultor or memory location
DAA	Decimal adjust A accumulator
DEC, DECA, DECB	Decrement accumulator or memory location
EORA, EORB	Exclusive or memory with accumulator
EXG R1, R2	Exchange R1 with R2 (R1, R2 = A, B, CC, DP)
INC, INCA, INCB	Increment accumulator or memory location
LDA, LDB	Load accumulator from memory
LSL, LSLA, LSLB	Logical shift left accumulator or memory location

(Continued)

Table 4 8-Bit Accumulator and Memory Instructions (Continued)

Mnemonic(s)	Operation
LSR, LSRA, LSRB	Logical shift right accumulator or memory location
MUL	Unsigned multiply (A × B → D)
NEG, NEGA, NEGB	Negate accumulator or memory
ORA, ORB	Or memory with accumulator
ROL, ROLA, ROLB	Rotate accumulator or memory left
ROR, RORA, RORB	Rotate accumulator or memory right
SBCA, SBCB	Subtract memory from accumulator with borrow
STA, STB	Store accumulator to memory
SUBA, SUBB	Subtract memory from accumulator
TST, TSTA, TSTB	Test accumulator or memory location
TFR R1, R2	Transfer R1 to R2 (R1, R2 = A, B, CC, DP)

(NOTE) A, B, CC or DP may be pushed to (pulled from) either stack with PSHS, PSHU (PULS, PULU) instructions.

Table 5 16-Bit Accumulator and Memory Instructions

Mnemonic(s)	Operation
ADDD	Add memory to D accumulator
CMPD	Compare memory from D accumulator
EXG D, R	Exchange D with X, Y, S, U or PC
LDD	Load D accumulator from memory
SEX	Sign Extend B accumulator into A accumulator
STD	Store D accumulator to memory
SUBD	Subtract memory from D accumulator
TFR D, R	Transfer D to X, Y, S, U or PC
TFR R, D	Transfer X, Y, S, U or PC to D

(NOTE) D may be pushed (pulled) to either stack with PSHS, PSHU (PULS, PULU) instructions.

Table 6 Index Register Stack Pointer Instructions

Mnemonic(s)	Operation
CMPS, CMPU	Compare memory from stack pointer
CMPX, CMPY	Compare memory from index register
EXG R1, R2	Exchange D, X, Y, S, U or PC with D, X, Y, S, U or PC
LEAS, LEAU	Load effective address into stack pointer
LEAX, LEAY	Load effective address into index register
LDS, LDU	Load stack pointer from memory
LDX, LDY	Load index register from memory
PSHS	Push A, B, CC, DP, D, X, Y, U, or PC onto hardware stack
PSHU	Push A, B, CC, DP, D, X, Y, S, or PC onto user stack
PULS	Pull A, B, CC, DP, D, X, Y, U or PC from hardware stack
PULU	Pull A, B, CC, DP, D, X, Y, S or PC from user stack
STS, STU	Store stack pointer to memory
STX, STY	Store index register to memory
TFR R1, R2	Transfer D, X, Y, S, U or PC to D, X, Y, S, U or PC
ABX	Add B accumulator to X (unsigned)

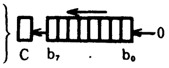
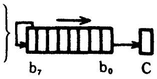
Table 7 Branch Instructions

Mnemonic(s)	Operation
	SIMPLE BRANCHES
BEQ, LBEQ	Branch if equal
BNE, LBNE	Branch if not equal
BMI, LBMI	Branch if minus
BPL, LBPL	Branch if plus
BCS, LBCS	Branch if carry set
BCC, LBCC	Branch if carry clear
BVS, LBVS	Branch if overflow set
BVC, LBVC	Branch if overflow clear
	SIGNED BRANCHES
BGT, LBGT	Branch if greater (signed)
BGE, LBGE	Branch if greater than or equal (signed)
BEQ, LBEQ	Branch if equal
BLE, LBLE	Branch if less than or equal (signed)
BLT, LBLT	Branch if less than (signed)
	UNSIGNED BRANCHES
BHI, LBHI	Branch if higher (unsigned)
BHS, LBHS	Branch if higher or same (unsigned)
BEQ, LBEQ	Branch if equal
BLS, LBLS	Branch if lower or same (unsigned)
BLO, LBLO	Branch if lower (unsigned)
	OTHER BRANCHES
BSR, LBSR	Branch to subroutine
BRA, LBRA	Branch always
BRN, LBRN	Branch never

Table 8 Miscellaneous Instructions

Mnemonic(s)	Operation
ANDCC	AND condition code register
CWAI	AND condition code register, then wait for interrupt
NOP	No operation
ORCC	OR condition code register
JMP	Jump
JSR	Jump to subroutine
RTI	Return from interrupt
RTS	Return from subroutine
SWI, SWI2, SWI3	Software interrupt (absolute indirect)
SYNC	Synchronize with interrupt line

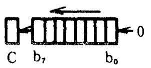
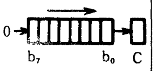
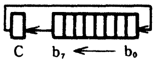
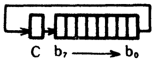
Table 9. HD6309E Instruction Set Table

INSTRUCTIONS/ FORMS		IMP ACCM REG			DIRECT			EXTND			IMMED			INDEX①			RELATIVE			DESCRIPTION	7	6	5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~⑤	#		E	F	H	I	N	Z	V	C
ABX		3A	3	1																B + X→X	●	●	●	●	●	●	●	●
																				(UNSIGNED)								
ADC	ADCA				99	4	2	B9	5	3	89	2	2	A9	4+	2+				A + M + C→A	●	●	↕	●	↕	↕	↕	↕
	ADCB				D9	4	2	F9	5	3	C9	2	2	E9	4+	2+				B + M + C→B	●	●	↕	●	↕	↕	↕	↕
ADD	ADDA				9B	4	2	BB	5	3	8B	2	2	AB	4+	2+				A + M→A	●	●	↕	●	↕	↕	↕	↕
	ADDB				DB	4	2	FB	5	3	CB	2	2	EB	4+	2+				B + M→B	●	●	↕	●	↕	↕	↕	↕
	ADDD				D3	6	2	F3	7	3	C3	4	3	E3	6+	2+				D + M M + 1→D	●	●	●	●	↕	↕	↕	↕
AND	ANDA				94	4	2	B4	5	3	84	2	2	A4	4+	2+				A∧M→A	●	●	●	●	↕	↕	R	●
	ANDB				D4	4	2	F4	5	3	C4	2	2	E4	4+	2+				B∧M→B	●	●	●	●	↕	↕	R	●
	ANDCC										1C	3	2							C C∧IMM→C C	(			⑦				)
ASL	ASLA	48	2	1																A 	●	●	⑧	●	↕	↕	↕	↕
	ASLB	58	2	1																B	●	●	⑧	●	↕	↕	↕	↕
	ASL				08	6	2	78	7	3				68	6+	2+				M	●	●	⑧	●	↕	↕	↕	↕
ASR	ASRA	47	2	1																A 	●	●	⑧	●	↕	↕	●	↕
	ASRB	57	2	1																B	●	●	⑧	●	↕	↕	●	↕
	ASR				07	6	2	77	7	3				67	6+	2+				M	●	●	⑧	●	↕	↕	●	↕
BCC	BCC																24	3	2	Branch C = 0	●	●	●	●	●	●	●	●
	LBCC																10	5(6)	4	Long Branch	●	●	●	●	●	●	●	●
																	24			C = 0								
BCS	BCS																25	3	2	Branch C = 1	●	●	●	●	●	●	●	●
	LBCS																10	5(6)	4	Long Branch	●	●	●	●	●	●	●	●
																	25			C = 1								
BEQ	BEQ																27	3	2	Branch Z = 1	●	●	●	●	●	●	●	●
	LBEQ																10	5(6)	4	Long Branch	●	●	●	●	●	●	●	●
																	27			Z = 1								
BGE	BGE																2C	3	2	Branch N⊕V = 0	●	●	●	●	●	●	●	●
	LBGE																10	5(6)	4	Long Branch	●	●	●	●	●	●	●	●
																	2C			N⊕V = 0								
BGT	BGT																2E	3	2	Branch Z∨(N⊕V) = 0	●	●	●	●	●	●	●	●
	LBGT																10	5(6)	4	Long Branch	●	●	●	●	●	●	●	●
																	2E			Z∨(N⊕V) = 0								
BHI	BHI																22	3	2	Branch C∨Z = 0	●	●	●	●	●	●	●	●
	LBHI																10	5(6)	4	Long Branch	●	●	●	●	●	●	●	●
																	22			C∨Z = 0								
BHS	BHS																24	3	2	Branch C = 0	●	●	●	●	●	●	●	●
	LBHS																10	5(6)	4	Long Branch	●	●	●	●	●	●	●	●
																	24			C = 0								
BIT	BITA				95	4	2	B5	5	3	85	2	2	A5	4+	2+				Bit Test A (M∧A)	●	●	●	●	↕	↕	R	●
	BITB				D5	4	2	F5	5	3	C5	2	2	E5	4+	2+				Bit Test B (M∧B)	●	●	●	●	↕	↕	R	●
BLE	BLE																2F	3	2	Branch Z∨(N⊕V) = 1	●	●	●	●	●	●	●	●
	LBLE																10	5(6)	4	Long Branch	●	●	●	●	●	●	●	●
																	2F			Z∨(N⊕V) = 1								
BLO	BLO																25	3	2	Branch C = 1	●	●	●	●	●	●	●	●
	LBLO																10	5(6)	4	Long Branch	●	●	●	●	●	●	●	●
																	25			C = 1								
BLS	BLS																23	3	2	Branch C∨Z = 1	●	●	●	●	●	●	●	●
	LBLS																10	5(6)	4	Long Branch	●	●	●	●	●	●	●	●
																	23			C∨Z = 1								
BLT	BLT																2D	3	2	Branch N⊕V = 1	●	●	●	●	●	●	●	●
	LBLT																10	5(6)	4	Long Branch	●	●	●	●	●	●	●	●
																	2D			N⊕V = 1								
BMI	BMI																2B	3	2	Branch N = 1	●	●	●	●	●	●	●	●
	LBMI																10	5(6)	4	Long Branch	●	●	●	●	●	●	●	●
																	2B			N = 1								

(Continued)

INSTRUCTIONS/ FORMS		IMP ACCM REG			DIRECT			EXTND			IMMED			INDEX①			RELATIVE			DESCRIPTION	7	6	5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~⑤	#		E	F	H	I	N	Z	V	C
BNE	BNE																26	3	2	Branch Z = 0	●	●	●	●	●	●	●	●
	LBNE																10	5(6)	4	Long Branch	●	●	●	●	●	●	●	●
																	26			Z = 0								
BPL	BPL																2A	3	2	Branch N = 0	●	●	●	●	●	●	●	●
	LBPL																10	5(6)	4	Long Branch	●	●	●	●	●	●	●	●
																	2A			N = 0								
BRA	BRA																20	3	2	Branch Always	●	●	●	●	●	●	●	●
	LBRA																16	5	3	Long Branch Always	●	●	●	●	●	●	●	●
BRN	BRN																21	3	2	Branch Never	●	●	●	●	●	●	●	●
	LBRN																10	5	4	Long Branch Never	●	●	●	●	●	●	●	●
																	21											
BSR	BSR																8D	7	2	Branch to	●	●	●	●	●	●	●	●
																				Subroutine								
	LBSR																17	9	3	Long Branch to	●	●	●	●	●	●	●	●
																				Subroutine								
BVC	BVC																28	3	2	Branch V = 0	●	●	●	●	●	●	●	●
	LBVC																10	5(6)	4	Long Branch	●	●	●	●	●	●	●	●
																	28			V = 0								
BVS	BVS																29	3	2	Branch V = 1	●	●	●	●	●	●	●	●
	LBVS																10	5(6)	4	Long Branch	●	●	●	●	●	●	●	●
																	29			V = 1								
CLR	CLRA	4F	2	1																0→A	●	●	●	●	R	S	R	R
	CLRB	5F	2	1																0→B	●	●	●	●	R	S	R	R
	CLR				0F	6	2	7F	7	3				6F	6+	2+				0→M	●	●	●	●	R	S	R	R
CMP	CMPA				91	4	2	B1	5	3	81	2	2	A1	4+	2+				Compare M from A	●	●	⑧	●	↕	↕	↕	↕
	CMPB				D1	4	2	F1	5	3	C1	2	2	E1	4+	2+				Compare M from B	●	●	⑧	●	↕	↕	↕	↕
	CMPD				10	7	3	10	8	4	10	5	4	10	7+	3+				Compare M M+1	●	●	●	●	↕	↕	↕	↕
					93			B3			83			A3						from D								
	CMPS				11	7	3	11	8	4	11	5	4	11	7+	3+				Compare M M+1	●	●	●	●	↕	↕	↕	↕
					9C			BC			8C			AC						from S								
	CMPU				11	7	3	11	8	4	11	5	4	11	7+	3+				Compare M M+1	●	●	●	●	↕	↕	↕	↕
					93			B3			83			A3						from U								
	CMPX				9C	6	2	BC	7	3	8C	4	3	AC	6+	2+				Compare M M+1	●	●	●	●	↕	↕	↕	↕
																				from X								
	CMPY				10	7	3	10	8	4	10	5	4	10	7+	3+				Compare M M+1	●	●	●	●	↕	↕	↕	↕
					9C			BC			8C			AC						from Y								
COM	COMA	43	2	1																$\overline{A}$→A	●	●	●	●	↕	↕	R	S
	COMB	53	2	1																$\overline{B}$→B	●	●	●	●	↕	↕	R	S
	COM				03	6	2	73	7	3				63	6+	2+				$\overline{M}$→M	●	●	●	●	↕	↕	R	S
CWAI											3C	≥20	2							CC ∧ IMM→CC	S	(			⑦			)
																				Wait for Interrupt								
DAA		19	2	1																Decimal Adjust A	●	●	●	●	↕	↕	⑧	↕
DEC	DECA	4A	2	1																A − 1→A	●	●	●	●	↕	↕	↕	●
	DECB	5A	2	1																B − 1→B	●	●	●	●	↕	↕	↕	●
	DEC				0A	6	2	7A	7	3				6A	6+	2+				M − 1→M	●	●	●	●	↕	↕	↕	●
EOR	EORA				98	4	2	B8	5	3	88	2	2	A8	4+	2+				A⊕M→A	●	●	●	●	↕	↕	R	●
	EORB				D8	4	2	F8	5	3	C8	2	2	E8	4+	2+				B⊕M→B	●	●	●	●	↕	↕	R	●
EXG	R1, R2	1E	8	2																R1↔R2②	(			⑩				)
INC	INCA	4C	2	1																A + 1→A	●	●	●	●	↕	↕	↕	●
	INCB	5C	2	1																B + 1→B	●	●	●	●	↕	↕	↕	●
	INC				0C	6	2	7C	7	3				6C	6+	2+				M + 1→M	●	●	●	●	↕	↕	↕	●
JMP					0E	3	2	7E	4	3				6E	3+	2+				EA③→PC	●	●	●	●	●	●	●	●
JSR					9D	7	2	BD	8	3				AD	7+	2+				Jump to Subroutine	●	●	●	●	●	●	●	●

(Continued)

INSTRUCTIONS/ FORMS		IMP ACCM REG			DIRECT			EXTND			IMMED			INDEX①			RELATIVE			DESCRIPTION	7	6	5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~⑤	#		E	F	H	I	N	Z	V	C
LD	LDA				96	4	2	B6	5	3	86	2	2	A6	4+	2+				M→A	●	●	●	●	↕	↕	R	●
	LDB				D6	4	2	F6	5	3	C6	2	2	E6	4+	2+				M→B	●	●	●	●	↕	↕	R	●
	LDD				DC	5	2	FC	6	3	CC	3	3	EC	5+	2+				MM+1→D	●	●	●	●	↕	↕	R	●
	LDS				10	6	3	10	7	4	10	4	4	10	6+	3+				MM+1→S	●	●	●	●	↕	↕	R	●
					DE			FE			CE			EE														
	LDU				DE	5	2	FE	6	3	CE	3	3	EE	5+	2+				MM+1→U	●	●	●	●	↕	↕	R	●
	LDX				9E	5	2	BE	6	3	8E	3	3	AE	5+	2+				MM+1→X	●	●	●	●	↕	↕	R	●
	LDY				10	6	3	10	7	4	10	4	4	10	6+	3+				MM+1→Y	●	●	●	●	↕	↕	R	●
					9E			BE			8E			AE														
LEA	LEAS													32	4+	2+				EA③→S	●	●	●	●	●	●	●	●
	LEAU													33	4+	2+				EA③→U	●	●	●	●	●	●	●	●
	LEAX													30	4+	2+				EA③→X	●	●	●	●	●	↕	●	●
	LEAY													31	4+	2+				EA③→Y	●	●	●	●	●	↕	●	●
LSĹ	LSLA	48	2	1																A	●	●	●	●	↕	↕	↕	↕
	LSLB	58	2	1																B 	●	●	●	●	↕	↕	↕	↕
	LSL				08	6	2	78	7	3				68	6+	2+				M	●	●	●	●	↕	↕	↕	↕
LSR	LSRA	44	2	1																A	●	●	●	●	R	↕	●	↕
	LSRB	54	2	1																B 	●	●	●	●	R	↕	●	↕
	LSR				04	6	2	74	7	3				64	6+	2+				M	●	●	●	●	R	↕	●	↕
MUL		3D	11	1																A × B→D (Unsigned)	●	●	●	●	●	↕	●	⑨
NEG	NEGA	40	2	1																$\overline{A}$ + 1→A	●	●	⑧	●	↕	↕	↕	↕
	NEGB	50	2	1																$\overline{B}$ + 1→B	●	●	⑧	●	↕	↕	↕	↕
	NEG				00	6	2	70	7	3				60	6+	2+				$\overline{M}$ + 1→M	●	●	⑧	●	↕	↕	↕	↕
NOP		12	2	1																No Operation	●	●	●	●	●	●	●	●
OR	ORA				9A	4	2	BA	5	3	8A	2	2	AA	4+	2+				A∨M→A	●	●	●	●	↕	↕	R	●
	ORB				DA	4	2	FA	5	3	CA	2	2	EA	4+	2+				B∨M→B	●	●	●	●	↕	↕	R	●
	ORCC										1A	3	2							CC∨ IMM→CC	(			⑦				)
PSH	PSHS	34	5+④	2																Push Registers on S Stack	●	●	●	●	●	●	●	●
	PSHU	36	5+④	2																Push Registers on U Stack	●	●	●	●	●	●	●	●
PUL	PULS	35	5+④	2																Pull Registers from S Stack	(			⑩				)
	PULU	37	5+④	2																Pull Registers from U Stack	(			⑩				)
ROL	ROLA	49	2	1																A	●	●	●	●	↕	↕	↕	↕
	ROLB	59	2	1																B 	●	●	●	●	↕	↕	↕	↕
	ROL				09	6	2	79	7	3				69	6+	2+				M	●	●	●	●	↕	↕	↕	↕
ROR	RORA	46	2	1																A	●	●	●	●	↕	↕	●	↕
	RORB	56	2	1																B 	●	●	●	●	↕	↕	●	↕
	ROR				06	6	2	76	7	3				66	6+	2+				M	●	●	●	●	↕	↕	●	↕
RTI		3B	6/15	1																Return from Interrupt	(			⑦				)
RTS		39	5	1																Return from Subroutine	●	●	●	●	●	●	●	●
SBC	SBCA				92	4	2	B2	5	3	82	2	2	A2	4+	2+				A − M − C→A	●	●	⑧	●	↕	↕	↕	↕
	SBCB				D2	4	2	F2	5	3	C2	2	2	E2	4+	2+				B − M − C→B	●	●	⑧	●	↕	↕	↕	↕
SEX		1D	2	1																Sign Extend B into A { Bのビット7 = 1 FF→A { Bのビット7 = 0 0→A	●	●	●	●	↕	↕	●	●

(Continued)

INSTRUCTIONS/ FORMS		ACCM IMP REG			DIRECT			EXTND			IMMED			INDEX①			RELATIVE			DESCRIPTION	7	6	5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~⑤	#		E	F	H	I	N	Z	V	C
ST	STA				97	4	2	B7	5	3				A7	4+	2+				A→M	●	●	●	●	↕	↕	R	●
	STB				D7	4	2	F7	5	3				E7	4+	2+				B→M	●	●	●	●	↕	↕	R	●
	STD				DD	5	2	FD	6	3				ED	5+	2+				D→M M+1	●	●	●	●	↕	↕	R	●
	STS				10 DF	6	3	10 FF	7	4				10 EF	6+	3+				S→M M+1	●	●	●	●	↕	↕	R	●
	STU				DF	5	2	FF	6	3				EF	5+	2+				U→M M+1	●	●	●	●	↕	↕	R	●
	STX				9F	5	2	BF	6	3				AF	5+	2+				X→M M+1	●	●	●	●	↕	↕	R	●
	STY				10 9F	6	3	10 BF	7	4				10 AF	6+	3+				Y→M M+1	●	●	●	●	↕	↕	R	●
SUB	SUBA				90	4	2	B0	5	3	80	2	2	A0	4+	2+				A -M→A	●	●	⑧	●	↕	↕	↕	↕
	SUBB				D0	4	2	F0	5	3	C0	2	2	E0	4+	2+				B - M→B	●	●	⑧	●	↕	↕	↕	↕
	SUBD				93	6	2	B3	7	3	83	4	3	A3	6+	2+				D - M M+1→D	●	●	●	●	↕	↕	↕	↕
SWI	SWI⑥	3F	19	1																Software interrupt 1	S	S	●	S	●	●	●	●
	SWI2⑥	10 3F	20	2																Software interrupt 2	S	●	●	●	●	●	●	●
	SWI3⑥	11 3F	20	2																Software interrupt 3	S	●	●	●	●	●	●	●
SYNC		13	≥4	1																Synchronize to interrupt	●	●	●	●	●	●	●	●
TFR	R1, R2	1F	6	2																R1 → R2②	(⑩)							
TST	TSTA	4D	2	1																Test A	●	●	●	●	↕	↕	R	●
	TSTB	5D	2	1																Test B	●	●	●	●	↕	↕	R	●
	TST				0D	6	2	7D	7	3				6D	6+	2+				Test M	●	●	●	●	↕	↕	R	●

(NOTES)

① This column gives a base cycle and byte count. To obtain total count, and the values obtained from the INDEXED ADDRESSING MODES table.

② R1 and R2 may be any pair of 8 bit or any pair of 16 bit registers.
The 8 bit registers are: A, B, CC, DP
The 16 bit registers are: X, Y, U, S, D, PC

③ EA is the effective address.

④ The PSH and PUL instructions require 5 cycle plus 1 cycle for each byte pushed or pulled.

⑤ 5(6) means: 5 cycles if branch not taken, 6 cycles if taken.

⑥ SWI sets 1 and F bits. SWI2 and SWI3 do not affect I and F.

⑦ Conditions Codes set as a direct result of the instruction.

⑧ Value of half-carry flag is undefined.

⑨ Special Case — Carry set if b7 is SET.

⑩ Condition Codes set as a direct result of the instruction if CC is specified, and not affected otherwise.

LEGEND:

OP	Operation Code (Hexadecimal)	Z	Zero (byte)
~	Number of MPU Cycles	V	Overflow, 2's complement
#	Number of Program Bytes	C	Carry from bit 7
+	Arithmetic Plus	↕	Test and set if true, cleared otherwise
–	Arithmetic Minus	●	Not Affected
×	Multiply	CC	Condition Code Register
$\overline{M}$	Complement of M	:	Concatenation
→	Transfer Into	∨	Logical or
H	Half-carry (from bit 3)	∧	Logical and
N	Negative (sign bit)	⊕	Logical Exclusive or

Table 10 Hexadecimal Values of Machine Codes

OP	Mnem	Mode	~	#	OP	Mnem	Mode	~	#	OP	Mnem	Mode	~	#
00	NEG	Direct	6	2	30	LEAX	Indexed	4+	2+	60	NEG	Indexed	6+	2+
01	*	↑			31	LEAY	↑	4+	2+	61	*	↑		
02	*				32	LEAS	↓	4+	2+	62	*			
03	COM		6	2	33	LEAU	Indexed	4+	2+	63	COM		6+	2+
04	LSR		6	2	34	PSHS	Implied	5+	2	64	LSR		6+	2+
05	*				35	PULS	↑	5+	2	65	*			
06	ROR		6	2	36	PSHU		5+	2	66	ROR		6+	2+
07	ASR		6	2	37	PULU		5+	2	67	ASR		6+	2+
08	ASL, LSL		6	2	38	*				68	ASL, LSL		6+	2+
09	ROL		6	2	39	RTS		5	1	69	ROL		6+	2+
0A	DEC		6	2	3A	ABX	↓	3	1	6A	DEC		6+	2+
0B	*				3B	RTI	Implied	6, 15	1	6B	*			
0C	INC		6	2	3C	CWAI	Immed	≧20	2	6C	INC		6+	2+
0D	TST		6	2	3D	MUL	Implied	11	1	6D	TST		6+	2+
0E	JMP	↓	3	2	3E	*				6E	JMP	↓	3+	2+
0F	CLR	Direct	6	2	3F	SWI	Implied	19	1	6F	CLR	Indexed	6+	2+
10	} See	—	—	—	40	NEGA	Implied	2	1	70	NEG	Extended	7	3
11	} Next Page	—	—	—	41	*	↑			71	*	↑		
12	NOP	Implied	2	1	42	*				72	*			
13	SYNC	Implied	≧4	1	43	COMA		2	1	73	COM		7	3
14	*				44	LSRA		2	1	74	LSR		7	3
15	*				45	*				75	*			
16	LBRA	Relative	5	3	46	RORA		2	1	76	ROR		7	3
17	LBSR	Relative	9	3	47	ASRA		2	1	77	ASR		7	3
18	*				48	ASLA, LSLA		2	1	78	ASL, LSL		7	3
19	DAA	Implied	2	1	49	ROLA		2	1	79	ROL		7	3
1A	ORCC	Immed	3	2	4A	DECA		2	1	7A	DEC		7	3
1B	*	—			4B	*				7B	*			
1C	ANDCC	Immed	3	2	4C	INCA		2	1	7C	INC		7	3
1D	SEX	Implied	2	1	4D	TSTA		2	1	7D	TST		7	3
1E	EXG	↕	8	2	4E	*	↓			7E	JMP	↓	4	3
1F	TFR	Implied	6	2	4F	CLRA	Implied	2	1	7F	CLR	Extended	7	3
20	BRA	Relative	3	2	50	NEGB	Implied	2	1	80	SUBA	Immed	2	2
21	BRN	↑	3	2	51	*	↑			81	CMPA	↑	2	2
22	BHI		3	2	52	*				82	SBCA		2	2
23	BLS		3	2	53	COMB		2	1	83	SUBD		4	3
24	BHS, BCC		3	2	54	LSRB		2	1	84	ANDA		2	2
25	BLO, BCS		3	2	55	*				85	BITA		2	2
26	BNE		3	2	56	RORB		2	1	86	LDA		2	2
27	BEQ		3	2	57	ASRB		2	1	87	*			
28	BVC		3	2	58	ASLB, LSLB		2	1	88	EORA		2	2
29	BVS		3	2	59	ROLB		2	1	89	ADCA		2	2
2A	BPL		3	2	5A	DECB		2	1	8A	ORA		2	2
2B	BMI		3	2	5B	*				8B	ADDA	↓	2	2
2C	BGE		3	2	5C	INCB		2	1	8C	CMPX	Immed	4	3
2D	BLT		3	2	5D	TSTB		2	1	8D	BSR	Relative	7	2
2E	BGT	↓	3	2	5E	*	↓			8E	LDX	Immed	3	3
2F	BLE	Relative	3	2	5F	CLRB	Implied	2	1	8F	*			

LEGEND:
~ Number of MPU cycles (less possible push pull or indexed-mode cycles)
# Number of program bytes
* Denotes unused opcode

(to be continued)

OP	Mnem	Mode	~	#	OP	Mnem	Mode	~	#	OP	Mnem	Mode	~	#
90	SUBA	Dırect	4	2	C6	LDB	Immed	2	2	FC	LDD	Extended	6	3
91	CMPA		4	2	C7	*				FD	STD		6	3
92	SBCA		4	2	C8	EORB		2	2	FE	LDU		6	3
93	SUBD		6	2	C9	ADCB		2	2	FF	STU	Extended	6	3
94	ANDA		4	2	CA	ORB		2	2					
95	BITA		4	2	CB	ADDB		2	2					
96	LDA		4	2	CC	LDD		3	3		2 Bytes Opcode			
97	STA		4	2	CD	*								
98	EORA		4	2	CE	LDU	Immed	3	3	1021	LBRN	Relative	5	4
99	ADCA		4	2	CF	*				1022	LBHI		5(6)	4
9A	ORA		4	2						1023	LBLS		5(6)	4
9B	ADDA		4	2	D0	SUBB	Dırect	4	2	1024	LBHS, LBCC		5(6)	4
9C	CMPX		6	2	D1	CMPB		4	2	1025	LBCS, LBLO		5(6)	4
9D	JSR		7	2	D2	SBCB		4	2	1026	LBNE		5(6)	4
9E	LDX		5	2	D3	ADDD		6	2	1027	LBEQ		5(6)	4
9F	STX	Direct	5	2	D4	ANDB		4	2	1028	LBVC		5(6)	4
					D5	BITB		4	2	1029	LBVS		5(6)	4
A0	SUBA	Indexed	4+	2+	D6	LDB		4	2	102A	LBPL		5(6)	4
A1	CMPA		4+	2+	D7	STB		4	2	102B	LBMI		5(6)	4
A2	SBCA		4+	2+	D8	EORB		4	2	102C	LBGE		5(6)	4
A3	SUBD		6+	2+	D9	ADCB		4	2	102D	LBLT		5(6)	4
A4	ANDA		4+	2+	DA	ORB		4	2	102E	LBGT		5(6)	4
A5	BITA		4+	2+	DB	ADDB		4	2	102F	LBLE	Relative	5(6)	4
A6	LDA		4+	2+	DC	LDD		5	2	103F	SWI2	Implied	20	2
A7	STA		4+	2+	DD	STD		5	2	1083	CMPD	Immed	5	4
A8	EORA		4+	2+	DE	LDU		5	2	108C	CMPY		5	4
A9	ADCA		4+	2+	DF	STU	Direct	5	2	108E	LDY	Immed	4	4
AA	ORA		4+	2+						1093	CMPD	Direct	7	3
AB	ADDA		4+	2+	E0	SUBB	Indexed	4+	2+	109C	CMPY		7	3
AC	CMPX		6+	2+	E1	CMPB		4+	2+	109E	LDY		6	3
AD	JSR		7+	2+	E2	SBCB		4+	2+	109F	STY	Direct	6	3
AE	LDX		5+	2+	E3	ADDD		6+	2+	10A3	CMPD	Indexed	7+	3+
AF	STX	Indexed	5+	2+	E4	ANDB		4+	2+	10AC	CMPY		7+	3+
					E5	BITB		4+	2+	10AE	LDY		6+	3+
B0	SUBA	Extended	5	3	E6	LDB		4+	2+	10AF	STY	Indexed	6+	3+
B1	CMPA		5	3	E7	STB		4+	2+	10B3	CMPD	Extended	8	4
B2	SBCA		5	3	E8	EORB		4+	2+	10BC	CMPY		8	4
B3	SUBD		7	3	E9	ADCB		4+	2+	10BE	LDY		7	4
B4	ANDA		5	3	EA	ORB		4+	2+	10BF	STY	Extended	7	4
B5	BITA		5	3	EB	ADDB		4+	2+	10CE	LDS	Immed	4	4
B6	LDA		5	3	EC	LDD		5+	2+	10DE	LDS	Dırect	6	3
B7	STA		5	3	ED	STD		5+	2+	10DF	STS	Direct	6	3
B8	EORA		5	3	EE	LDU		5+	2+	10EE	LDS	Indexed	6+	3+
B9	ADCA		5	3	EF	STU	Indexed	5+	2+	10EF	STS	Indexed	6+	3+
BA	ORA		5	3						10FE	LDS	Extended	7	4
BB	ADDA		5	3	F0	SUBB	Extended	5	3	10FF	STS	Extended	7	4
BC	CMPX		7	3	F1	CMPB		5	3	113F	SWI3	Implied	20	2
BD	JSR		8	3	F2	SBCB		5	3	1183	CMPU	Immed	5	4
BE	LDX		6	3	F3	ADDD		7	3	118C	CMPS	Immed	5	4
BF	STX	Extended	6	3	F4	ANDB		5	3	1193	CMPU	Direct	7	3
					F5	BITB		5	3	119C	CMPS	Dırect	7	3
C0	SUBB	Immed	2	2	F6	LDB		5	3	11A3	CMPU	Indexed	7+	3+
C1	CMPB		2	2	F7	STB		5	3	11AC	CMPS	Indexed	7+	3+
C2	SBCB		2	2	F8	EORB		5	3	11B3	CMPU	Extended	8	4
C3	ADDD		4	3	F9	ADCB		5	3	11BC	CMPS	Extended	8	4
C4	ANDB		2	2	FA	ORB		5	3					
C5	BITB	Immed	2	2	FB	ADDB	Extended	5	3					

(NOTE): All unused opcodes are both undefined and illegal

■ NOTE FOR USE

Execution Sequence of CLR Instruction

Cycle-by-cycle flow of CLR instruction (Direct, Extended, Indexed Addressing Mode) is shown below. In this sequence the content of the memory location specified by the operand is read before writing "00" into it. Note that status Flags, such as IRQ Flag, will be cleared by this extra data read operation when accessing the control/status register (sharing the same address between read and write) of peripheral devices.

Example: CLR (Extended)

```
$8000    CLR    $A000
$A000    FCB    $80
```

Cycle #	Address	Data	$R/\overline{W}$	Description
1	8000	7F	1	Opcode Fetch
2	8001	A0	1	Operand Address, High Byte
3	8002	00	1	Operand Address, Low Byte
4	FFFF	*	1	$\overline{\text{VMA}}$ Cycle
5	A000	80	1	Read the Data
6	FFFF	*	1	$\overline{\text{VMA}}$ Cycle
7	A000	00	0	Store Fixed "00" into Specified Location

* The data bus has the data at that particular address.

Section Three

HD64180 8-Bit Microprocessor Family

HD64180R/Z
8-BIT CMOS (Micro Processing Unit)

Based on a microcoded execution unit and advanced CMOS manufacturing technology, the HD64180 is an 8-bit MPU which provides the benefits of high performance, reduced system cost and low power operation while maintaining compatibility with the large base of industry standard 8-bit software

Performance is improved by virtue of high operating frequency, pipelining, enhanced instruction set and an integrated Memory Management Unit (MMU) with 1M or 512k bytes memory physical addiess space

System cost is reduced by incorporating key system functions on-chip including the MMU, two channel refresh, two channel Asynchronous Serial Communication Interface (ASCI), Clocked Serial I/O Port (CSI/O), two channel 16-bit Programmable Reload Timer (PRT), Versatile 12 source interrupt controller and a 'dual' (68× ×, 80× ×) bus interface.

Low power consumption duiing normal CPU operation is supplemented by two specific software controlled low power operation modes.

The HD64180, when combined with CMOS VLSI memories and peripherals, is useful in system applications requiring high performance, battery power operation and standard software compatibility.

The HD64180Z is fully compatible with Z80180 (Z180) which is marketed by Zilog Inc

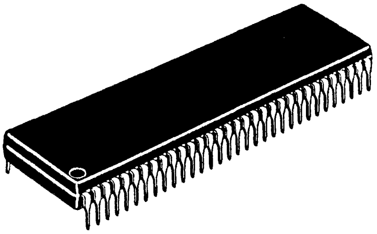
■ Software Features

- Enhanced standard 8-bit software architecture·
  Upward compatible with CP/M-80®

■ Hardware Features

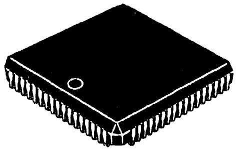
- On-chip MMU supporting 1 M byte memory (Provided 512K byte for DP-64S.
- Two channel DMAC with memory-memory, memory-I/O and memory-memory mapped I/O transfer capabilities
- Two channel, full duplex asynchronous serial communication interface (ASC) with programmable baud rate generator and modem control handshake signals
- One channel clocked serial I/O port with serial/parallel shift register
- Two channel 16-bit programmable reload timer for output waveform generation
- Four external and eight internal interrupts
- Dual bus interface compatible with Motorola 68 family and with Intel 80 family
- On-chip clock generator
- Operating Frequency up to 10 MHz
- Low power dissipation: 50 mW at 4 MHz Operation (typ.)
- R = Interface with 63/68, 80xx peripherals
- Z = Interface with Z80 peripherals

HD64180RP
HD64180ZP



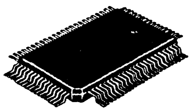
(DP-64S)

HD64180RCP
HD64180ZCP



(CP-68)

HD64180RF
HD64180ZF



(FP-80B)

Pin Function Differences in the HD64180 Series

Package * Type	Pin No.	HD64180R1	HD64180Z
CP-68	18	V_{SS}	V_{SS}
	35	A_{19}	A_{19}
	52	NC	TEST
FP-80B	12	V_{SS}	V_{SS}
	33	A_{19}	A_{19}
	53	NC	TEST

*"J" after package designation indicates industrial temperature parts.

HD64180R

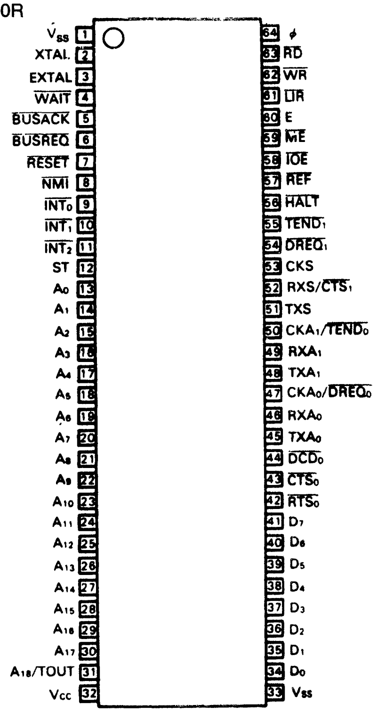
Part No.	Clock Frequency (MHz)	Package Type	Address Space
HD64180RP-6	6	DP-64S	512 K Byte
HD64180RP-8	8		
HD64180RP-10	10		
HD64180RF-6X	6	FP-80	1 M Byte
HD64180RF-8X	8		
HD64180RF-10X	10		
HD64180RCP-6X	6	CP-68	1 M Byte
HD64180RCP-8X	8		
HD64180RCP-10X	10		

HD64180Z

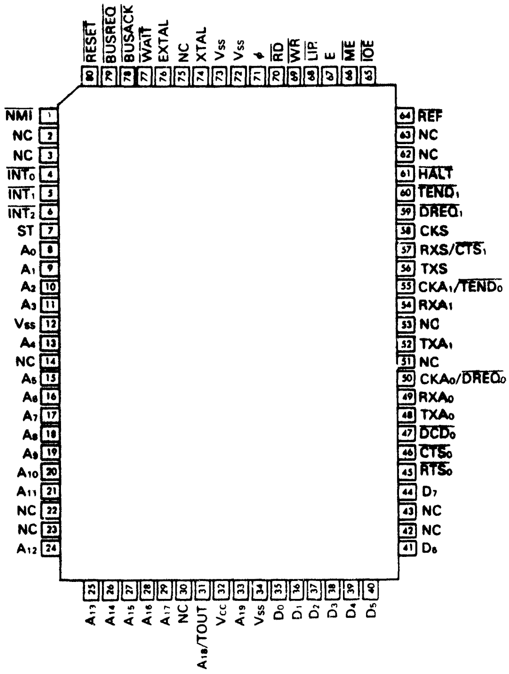
Part No.	Clock Frequency (MHz)	Package Type	Address Space
HD64180ZP-6	6	DP-64S	512 K Byte
HD64180ZP-8	8		
HD64180ZP-10	10		
HD64180ZF-6X	6	FP-80	1 M Byte
HD64180ZF-8X	8		
HD64180ZF-10X	10		
HD64180ZCP-6X	6	CP-68	1 M Byte
HD64180ZCP-8X	8		
HD64180ZCP-10X	10		

■ PIN ASSIGNMENT

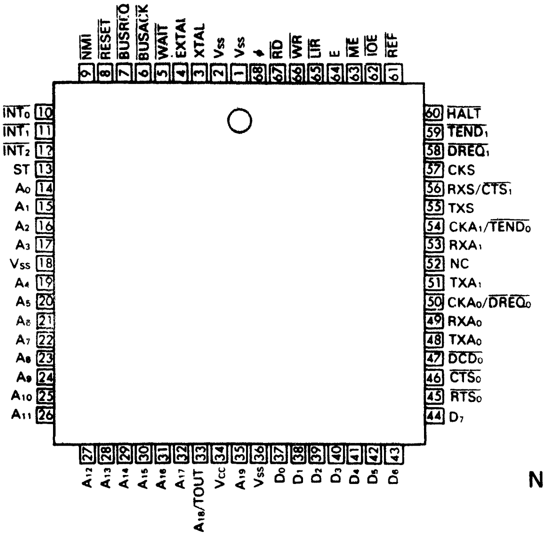
- HD64180R



(DP-64S)



(FP-80B)

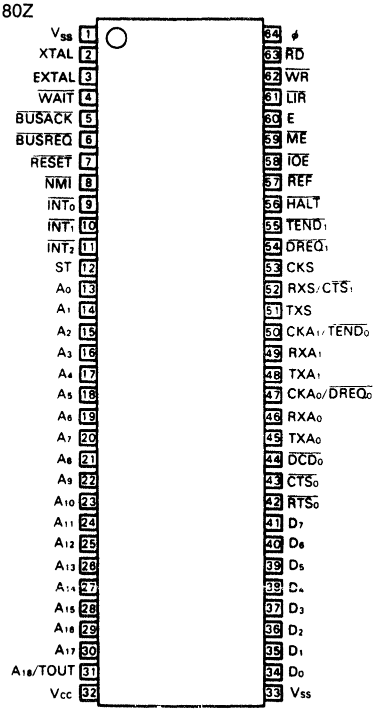


NC: Not connected.
Please leave the
NC pins open.

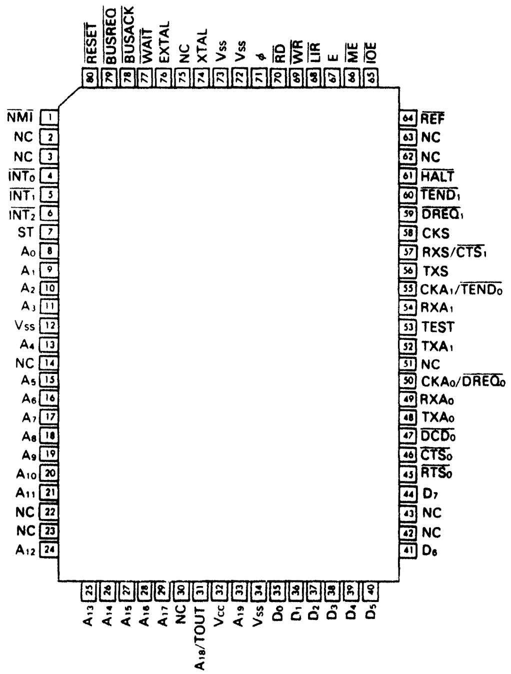
(CP-68)

■ PIN ASSIGNMENT

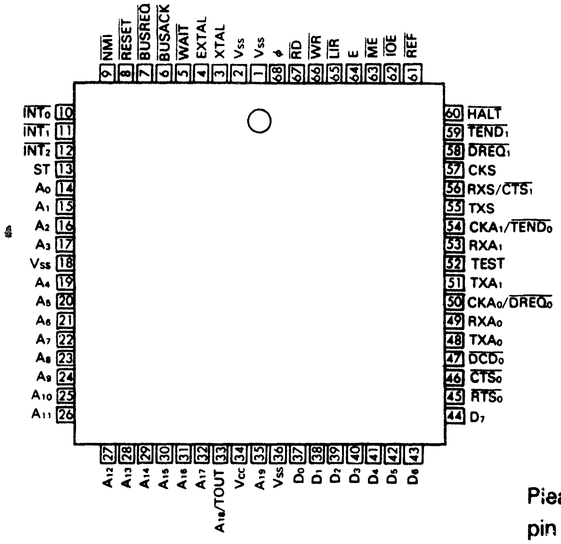
• HD64180Z



(DP-64S)



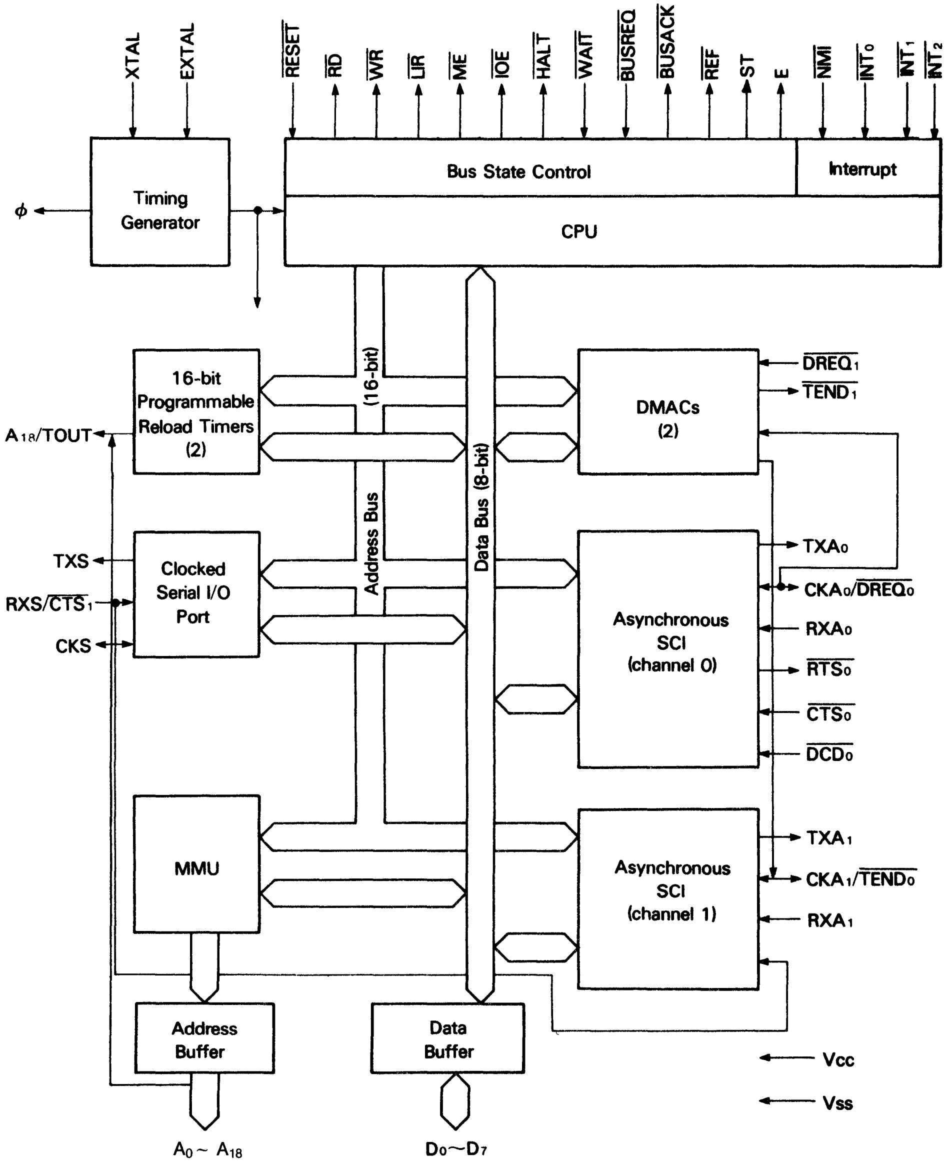
(FP-80B)



Please leave the TEST pin open.

(CP-68)

■ HD64180 BLOCK DIAGRAM



(A_0 ~ A_{19}: HD64180R1, HD64180Z; FP-80, CP-68)

■ ABSOLUTE MAXIMUM RATINGS

Item	Symbol	Value	Unit
Supply Voltage	V_{cc}	-0.3 ~ +7.0	V
Input Voltage	V_{in}	-0.3 ~ V_{cc} + 0.3	V
Operating Temperature	T_{opr}	-20 ~ +75* -40 ~ +85**	°C
Storage Temperature	T_{stg}	-55 ~ +150	°C

(NOTE) Permanent LSI damage may occur if maximum ratings are exceeded. Normal operation should be under recommended operating conditions. If these conditions are exceeded, it could affect reliability of LSI

*Standard Temp.

**Industrial Temp.

■ ELECTRICAL CHARACTERISTICS

• DC CHARACTERISTICS (V_{cc}=5V ± 10%, V_{ss}=0V, ta=-20 ~ +75°C, Industrial Temp Ta=-40 ~ +85°C, unless otherwise noted.)

Item	Symbol	Condition	min.	typ.	max.	Unit
Input "H" Voltage $\overline{\text{RESET}}$, EXTAL, $\overline{\text{NMI}}$	V_{IH1}		V_{CC}-0.6	—	V_{CC} + 0.3	V
Input "H" Voltage Except $\overline{\text{RESET}}$, EXTAL, NMI	V_{IH2}		2.0	—	V_{CC} + 0.3	V
Input "L" Voltage $\overline{\text{RESET}}$, EXTAL, $\overline{\text{NMI}}$	V_{IL1}		-0.3	—	0.6	V
Input "L" Voltage Except $\overline{\text{RESET}}$, EXTAL, $\overline{\text{NMI}}$	V_{IL2}		-0.3	—	0.8	V
Output "H" Voltage All Outputs	V_{OH}	I_{OH} = -200μA	2.4	—	—	V
		I_{OH} = -20μA	V_{CC}-1.2			
Output "L" Voltage All Outputs	V_{OL}	I_{OL} = 2.2mA	—	—	0.45	V
Input Leakage Current All Inputs Except XTAL, EXTAL	I_{IL}	V_{in} = 0.5 ~ V_{CC}-0.5	—	—	1.0	μA
Three State Leakage Current	I_{TL}	V_{in} = 0.5 ~ V_{CC}-0.5	—	—	1.0	μA
Power Dissipation (Normal Operation)	I_{CC}*	f = 4MHz	—	10	20	mA
		f = 6MHz	—	15	30	
		f = 8HMz	—	20	40	
		f = 10MHz	—	25	50	
Power Dissipation (SYSTEM STOP mode)		f = 4HMz	—	2.5	5.0	
		f = 6MHz	—	3.3	7.5	
		f = 8HMz	—	5.0	10.0	
		f = 10MHz	—	6.3	12.5	
Pin Capacitance	Cp	V_{in} = 0V, f = 1HMz Ta = 25°C	—	—	12	pF

*V_{IH} min. = V_{CC}-1.0V, V_{IL} max. = 0.8V (all output terminal are at no load)

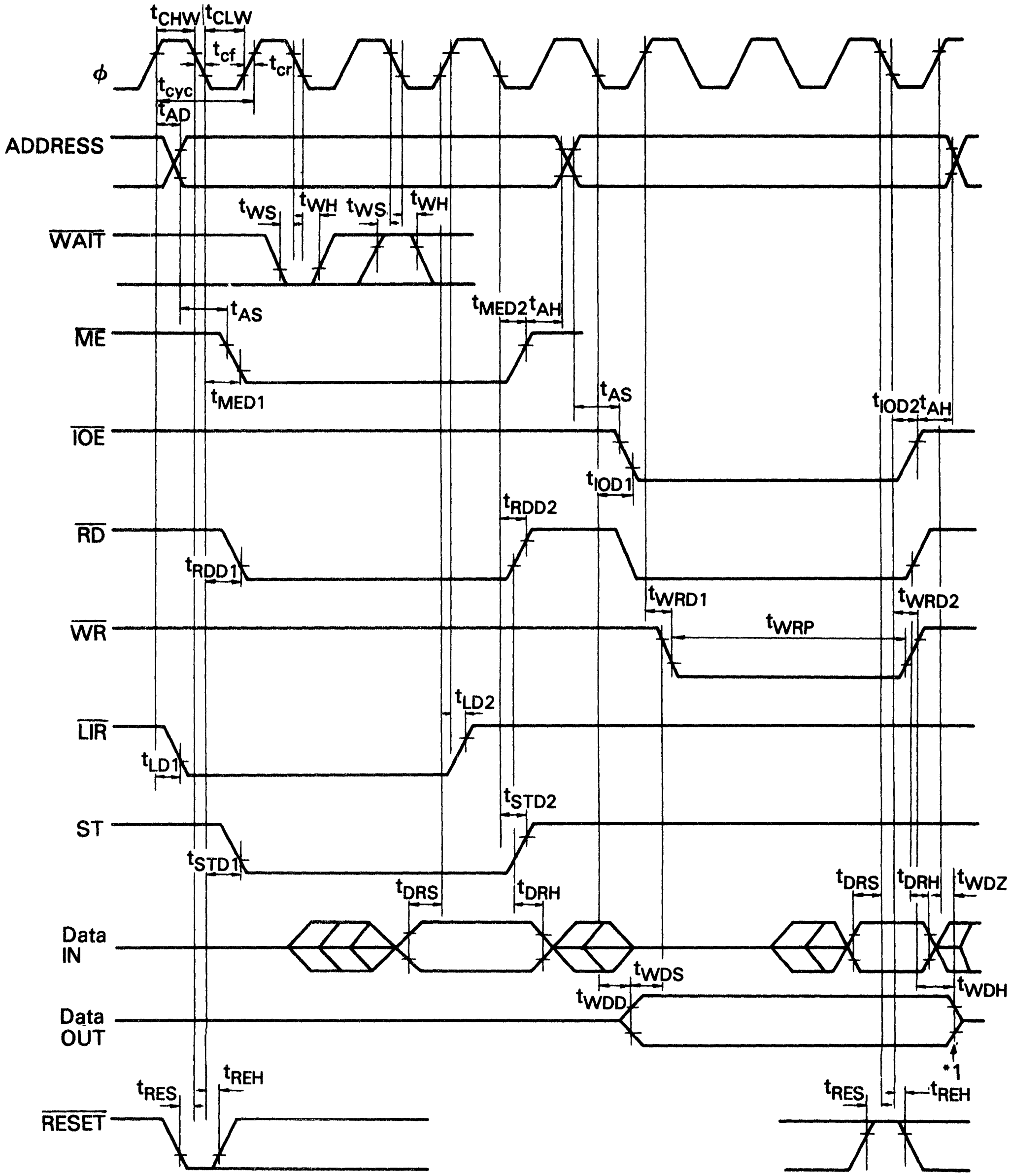
- **DC CHARACTERISTICS (V_{cc}=5V ± 10%, V_{ss}=0V, ta=-20 ~ +75°C, Industrial Temp Ta=-40 ~ +85°C, unless otherwise noted.)**

		HD64180R/Z-4		HD64180R/Z-6		HD64180R/Z-8		HD64180R/Z-10		
Item	**Symbol**	**min.**	**max.**	**min.**	**max.**	**min.**	**max.**	**min.**	**max.**	**unit**
Clock Cycle Time	t_{cyc}	250	2000	162	2000	125	2000	100	2000	ns
Clock "H" Pulse Width	t_{CHW}	110		65		50		40		ns
Clock "L" Pulse Width	t_{CLW}	110		65		50		40		ns
Clock Fall Time	t_{cf}		15		15		15		10	ns
Clock Rise Time	t_{cr}		15		15		15		10	ns
Address Delay Time	t_{AD}		110		90		80		70	ns
Address Set-up Time ($\overline{ME}$ or $\overline{IQE}$ ↓)	t_{AS}	50		30		20			70	ns
$\overline{ME}$ Delay Time 1	t_{MED1}		85		60		45	10		ns
$\overline{RD}$ Delay Time 1 IO C=1	t_{RDD1}		85		60		45		50	ns
$\overline{RD}$ Delay Time 1 IO C=0			85		65		60		55	
$\overline{LIR}$ Delay Time 1	t_{LD1}		100		80		70*		60	ns
Address Hold Time 1 ($\overline{ME}$, $\overline{IQE}$, $\overline{RD}$ or $\overline{WR}$ ↑)	t_{AH}	80		35		20		10		ns
$\overline{ME}$ Delay Time 2	t_{MED} 2		85		60		45		50	ns
$\overline{RD}$ Delay Time 2	t_{RDD2}		85		60		45		50	ns
$\overline{LIR}$ Delay Time 2	t_{LD2}		100		80		70*		60	ns
Data Read Set-up Time	t_{DRS}	50		40		30		25		ns
Data Read Hold Time	t_{DRH}	0		0		0		0		ns
ST Delay Time 1	t_{STD1}		110		90		70		60	ns
ST Delay Time 2	t_{STD2}		110		90		70		60	ns
$\overline{WAIT}$ Set-up Time	t_{WS}	80		40		40		30		ns
$\overline{WAIT}$ Hold Time	t_{WH}	70		40		40		30		ns
Write Data Floating Delay Time	t_{WDZ}		100		95		70		60	ns
$\overline{WR}$ Delay Time 1	t_{WRD1}		90		65		60		50	ns
Write Data Delay Time	t_{WDD}		110		90		80		60	ns
Write Data Set-up Time ($\overline{WR}$ ↓)	t_{WDS}	60		40		20		15		ns
$\overline{WR}$ Delay Time 2	t_{WRD2}		90		80		60		50	ns
$\overline{WR}$ Pulse Width	t_{WRP}	280		170		130		110		ns

*For a loading capacitance of less than or equal to 40 picofarads and operating temperature from 0 to 50 degrees, subtract 10 nanoseconds from the value given in the maximum columns.

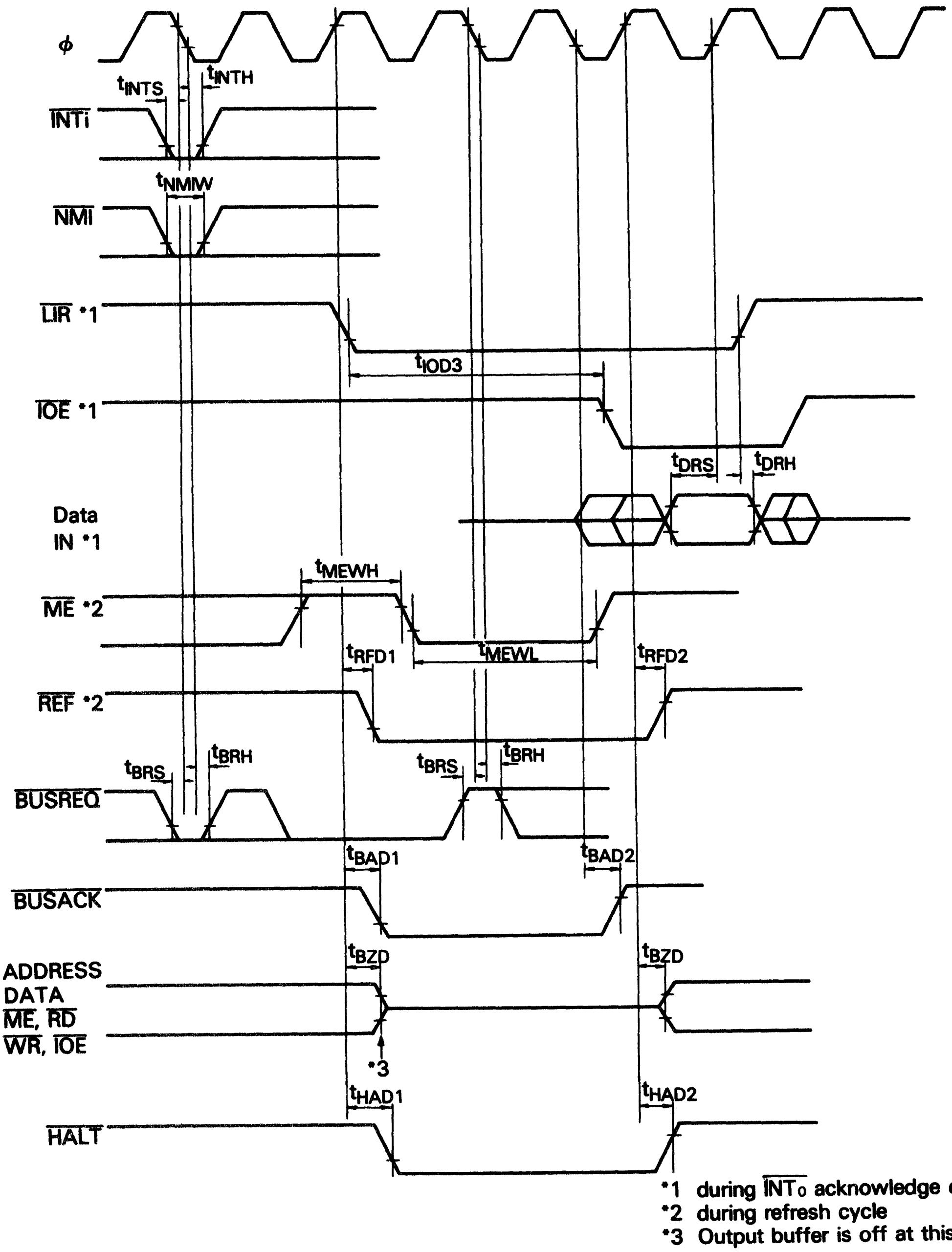
Item	Symbol	HD64180R/Z-4		HD64180R/Z-6		HD64180R/Z-8		HD64180R/Z-10		unit
		min.	max.	min.	max.	min.	max.	min.	max.	
Write Data Hold Time ($\overline{\text{WR}}$ ↑)	t_{WDH}	60		40		15		10		ns
$\overline{\text{IOE}}$ Delay Time 1	t_{IOD1}		85		60		45		50	ns
			85		65		60		55	ns
$\overline{\text{IOE}}$ Delay Time 2	t_{IOD2}		85		60		45		50	ns
$\overline{\text{IOE}}$ Delay Time 2 ($\overline{\text{LIR}}$ ↓)	t_{IOD3}	540		340		250		200		ns
$\overline{\text{INT}}$ Set-up Time (ϕ ↓)	t_{INTS}	80		40		40		30		ns
$\overline{\text{INT}}$ Hold Time (ϕ ↓)	t_{INTH}	70		40		40		30		ns
$\overline{\text{NMI}}$ Pulse Width	t_{NMIW}	120		120		100		80		ns
$\overline{\text{BUSREQ}}$ Set-up Time (ϕ ↓)	t_{BRS}	80		40		40		30		ns
$\overline{\text{BUSREQ}}$ Hold Time	t_{BRH}	70		40		40		30		ns
$\overline{\text{BUSACK}}$ Delay Time 1	t_{BAD1}		100		95		70		60	ns
$\overline{\text{BUSACK}}$ Delay Time 2	t_{BAD2}		100		95		70		60	ns
Bus Floating Delay Time	t_{BZD}		130		125		90		70	ns
$\overline{\text{ME}}$ Pulse Width (HIGH)	t_{MEWH}	200		110		90		70		ns
$\overline{\text{ME}}$ Pulse Width (LOW)	t_{MEWL}	210		125		100		80		ns
$\overline{\text{REF}}$ Delay Time 1	t_{RFD1}		110		90		80		60	ns
$\overline{\text{REF}}$ Delay Time 2	t_{RFD2}		110		90		80		60	ns
$\overline{\text{HALT}}$ Delay Time 1	t_{HAD1}		110		90		80		50	ns
$\overline{\text{HALT}}$ Delay Time 2	t_{HAD2}		110		90		80		50	ns
$\overline{\text{DREQ}}$ i Set-up Time	t_{DRQS}	80		40		40		30		ns
$\overline{\text{DREQ}}$ i Hold Time	t_{DRQH}	70		40		40		30		ns
$\overline{\text{TEND}}$ i Delay Time 1	t_{TED1}		85		70		60		50	ns
$\overline{\text{TEND}}$ i Delay Time 2	t_{TED2}		85		70		60		50	ns
Enable Delay Time 1	t_{ED1}		100		95		70		60	ns
Enable Delay Time 2	t_{ED2}		100		95		70		60	ns
E Pulse Width (HIGH)	P_{WEH}	150		75		65		55		ns
E Pulse Width (LOW)	P_{WEL}	300		180		130		110		ns

		HD64180R/Z-4		HD64180R/Z-6		HD64180R/Z-8		HD64180R/Z-10		
Item	Symbol	min.	max.	min.	max.	min.	max.	min.	max.	unit
Enable Rise Time	t_{Er}		25		20		20		20	ns
Enable Fall Time	t_{Ef}		25		20		20		20	ns
Timer Output Delay Time	t_{TOD}		300		300		200		150	ns
CSI/O Transmit Data Delay Time (Internal Clock Operation)	t_{STDI}		200		200		200		150	ns
CSI/O Transmit Data Delay Time (External Clock Operation)	t_{STDE}		7.5tcyc +300		7.5tcyc +300		7.5tcyc +200		7.5tcyc +150	ns
CSI/O Receive Data Set-up Time (Internal Clock Operation)	t_{SRSI}	1		1		1		1		tcyc
CSI/O Receive Data Hold Time (Internal Clock Operation)	t_{SRHI}	1		1		1		1		tcyc
CSI/O Receive Data Set-up Time (External Clock Operation)	t_{SRSE}	1		1		1		1		tcyc
CSI/O Receive Data	t_{SRHE}	1		1		1		1		tcyc
$\overline{\text{RESET}}$ Set-up Time	t_{RES}	120		120		100		80		ns
$\overline{\text{RESET}}$ Hold Time	t_{REH}	80		80		70		60		ns
Oscillator Stabilization Time	t_{OSC}		20		20		20		40	ms
External Clock Rise Time (EXTAL)	t_{EXr}		25		25		25		25	ns
External Clock Fall Time (EXTAL)	t_{EXf}		25		25		25		25	ns
$\overline{\text{RESET}}$ Rise Time	t_{Rr}		50		50		50		50	ms
$\overline{\text{RESET}}$ Fall Time	t_{Rf}		50		50		50		50	ms
Input Rise Time (except EXTAL, $\overline{\text{RESET}}$)	t_{Ir}		10		100		100		100	ns
Input Fall Time (except EXTAL, $\overline{\text{RESET}}$)	t_{If}		100		100		100		100	ns



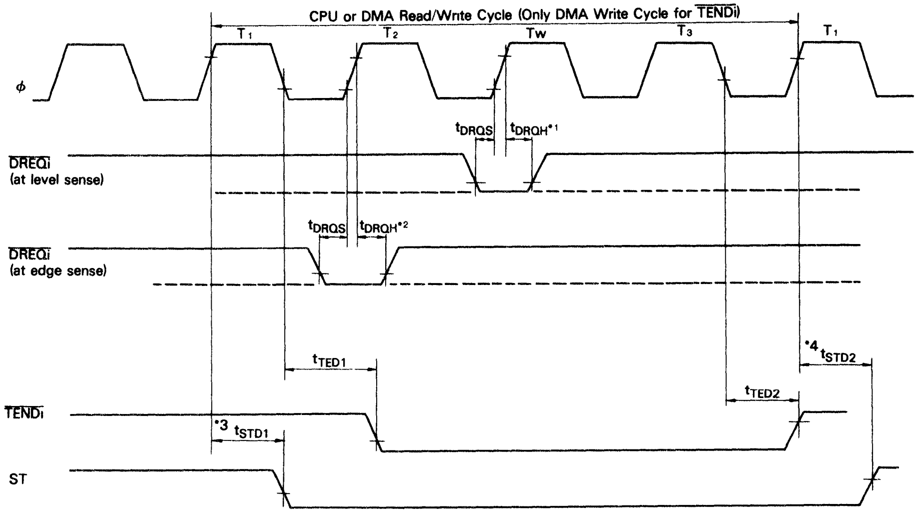
*1 Output buffer is off at this point.

Figure 1 CPU Timing (1)



*1 during INTo acknowledge cycle
*2 during refresh cycle
*3 Output buffer is off at this point.

Figure 1 CPU Timing (2)



*1 t_{DRQS} and t_{DRQH} are specified for the rising edge of clock followed by T3
*2 t_{DRQS} and t_{DRQH} are specified for the rising edge of clock
*3 DMA cycle starts
*4 CPU cycle starts

Figure 2 DMA Control Signals

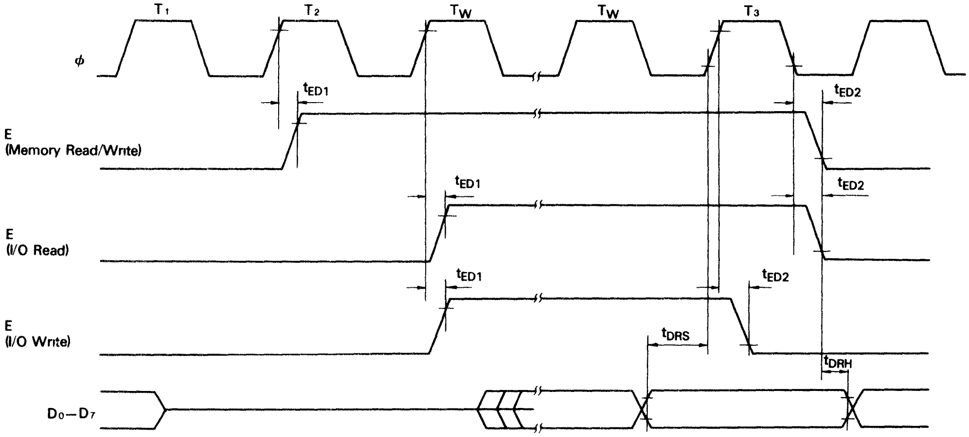


Figure 3 E Clock Timing (1)

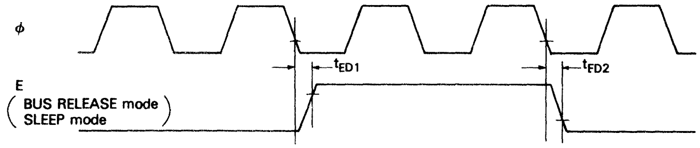


Figure 3 E Clock Timing (2)

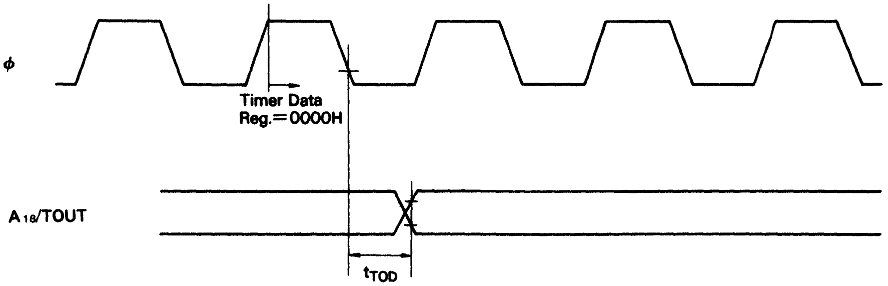


Figure 4 Timer Output Timing

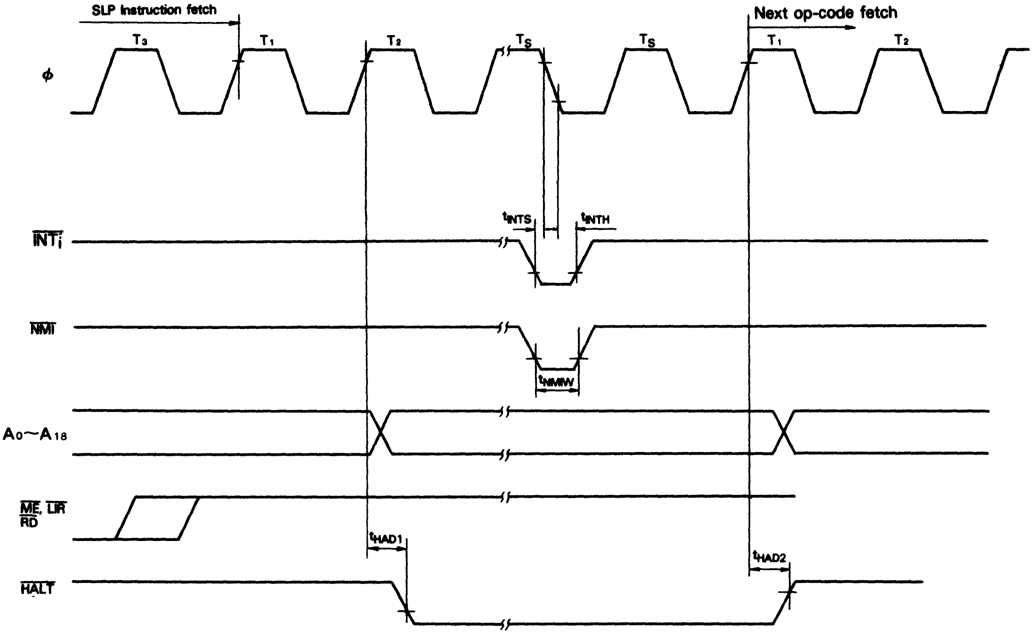


Figure 5 SLP Execution Cycle

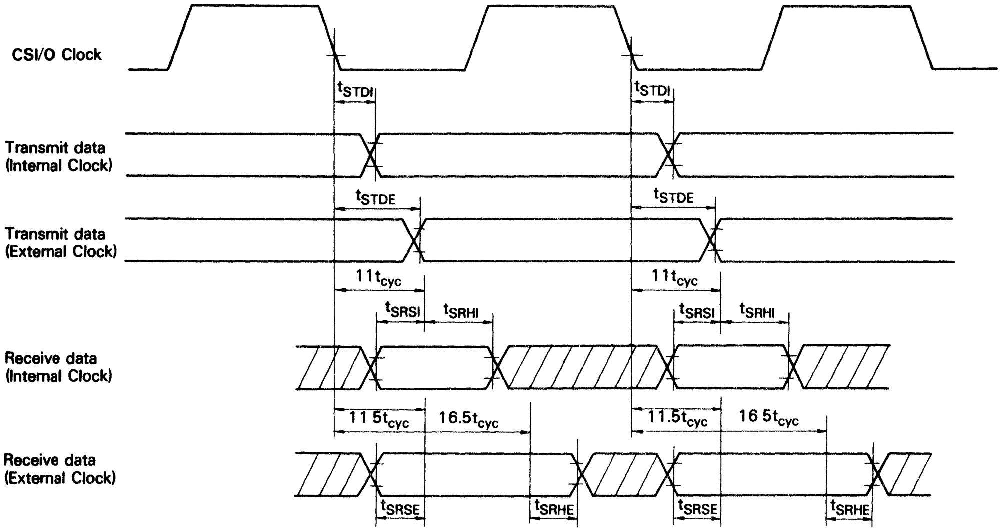
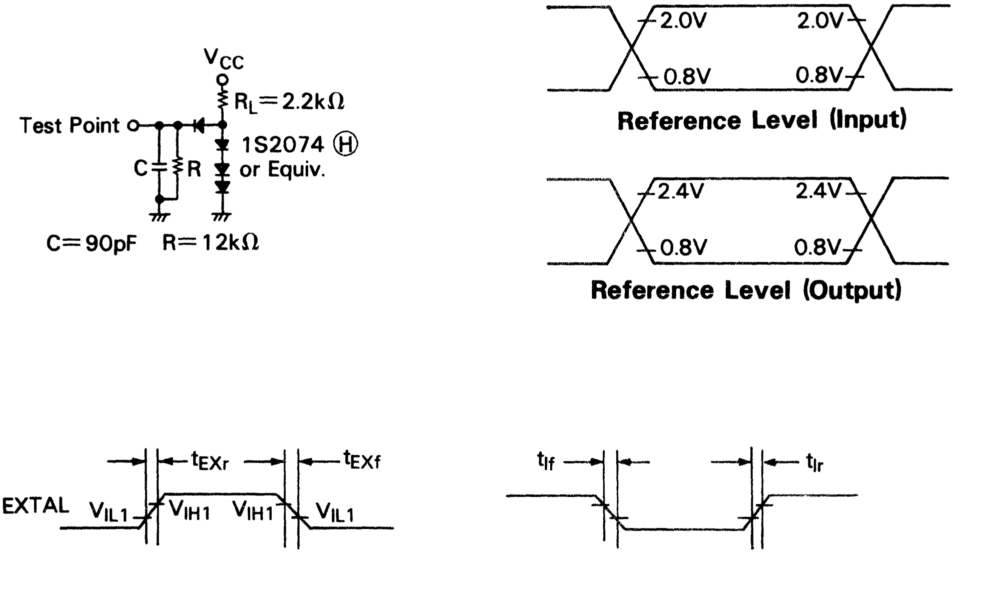


Figure 6 CSI/O Receive/Transmit Timing



Reference Level (Input)

Reference Level (Output)

EXTAL Rise time and Fall time

Inputs, other than EXTAL, Rise time and Fall time

Figure 7 Bus Timing Test Load (TTL Load)

1 PIN DESCRIPTION

XTAL (IN)

Crystal oscillator connection. Should be left open if an external TTL clock is used. It is noted this input is not a TTL level input. See Table D.C. characteristics.

EXTAL (IN)

Crystal oscillator connection. An external TTL clock can be input on this line. This input is schmitt triggered.

ϕ (OUT)

System Clock. The frequency is equal to one-half of crystal oscillator.

$\overline{\text{RESET}}$ – CPU Reset (IN)

When LOW, initializes the HD64180 CPU. All output signals are held inactive during RESET.

A_0-A_{17} – Address Bus (OUT, 3-STATE)
A_{18}/TOUT

19-bit address bus provides physical memory addresses of up to 512k bytes. The address bus enters the high impedance state during $\overline{\text{RESET}}$ and when another device acquires the bus as indicated by $\overline{\text{BUSREQ}}$ and $\overline{\text{BUSACK}}$ LOW. A_{18} is multiplexed with the TOUT output from PRT channel 1. During RESET, the address bus function is selected. TOUT function can be selected under software control.

D_0-D_7 – Data Bus (IN/OUT, 3-STATE)

Bidirectional 8-bit data bus. The data bus enters the high impedance state during $\overline{\text{RESET}}$ and when another device acquires the bus as indicated by $\overline{\text{BUSREQ}}$ and $\overline{\text{BUSACK}}$ LOW.

$\overline{\text{RD}}$ – Read (OUT, 3-STATE)

Used during a CPU read cycle to enable transfer from the external memory or I/O device to the CPU data bus.

$\overline{\text{WR}}$ – Write (OUT, 3-STATE)

Used during a CPU write cycle to enable transfer from the CPU data bus to the external memory or I/O device.

$\overline{\text{ME}}$ – Memory Enable (OUT, 3-STATE)

Indicates memory read or write operation. The HD64180 asserts $\overline{\text{ME}}$ LOW in the following cases.
(a) When fetching instructions and operands.
(b) When reading or writing memory data.
(c) During memory access cycles of DMA.
(d) During dynamic RAM refresh cycles.

$\overline{\text{IOE}}$ – I/O Enable (OUT, 3-STATE)

Indicates I/O read or write operation. The HD64180 asserts $\overline{\text{IOE}}$ LOW in the following cases.
(a) When reading or writing I/O data.
(b) During I/O access cycles of DMA.
(c) During $\overline{\text{INT}}_0$ acknowledge cycle

$\overline{\text{WAIT}}$ – Bus Cycle Wait (IN)

Introduces wait states to extend memory and I/O cycles. If LOW at the falling edge of T_2, a wait state (Tw) is inserted. Wait states will continue to be inserted until the $\overline{\text{WAIT}}$ input is sampled HIGH at the falling edge of Tw, at which time the bus cycle will proceed to completion.

E – Enable (OUT)

Synchronous clock for connection to HD63× × series and other 6800/6500 series compatible peripheral LSIs.

$\overline{\text{BUSREQ}}$ – Bus Request (IN)

Another device may request use of the bus by asserting $\overline{\text{BUSREQ}}$ LOW. The CPU will stop executing instructions and places the address bus, data bus, $\overline{\text{RD}}$, $\overline{\text{WR}}$, $\overline{\text{ME}}$ and $\overline{\text{IOE}}$ in the high impedance state.

$\overline{\text{BUSACK}}$ – Bus Acknowledge (OUT)

When the CPU completes bus release (in response to $\overline{\text{BUSREQ}}$ LOW), it will assert $\overline{\text{BUSACK}}$ LOW. This acknowledges that the bus is free for use by the requesting device

$\overline{\text{HALT}}$ – Halt/Sleep Status (OUT)

Asserted LOW after execution of the HALT or SLP instructions. Used with $\overline{\text{LIR}}$ and ST output pins to encode CPU status.

$\overline{\text{LIR}}$ – Load Instruction Register (OUT)

Asserted LOW when the current cycle is an op-code fetch cycle. Used with $\overline{\text{HALT}}$ and ST output pins to encode CPU status.

ST – Status (OUT)

Used with the $\overline{\text{HALT}}$ and $\overline{\text{LIR}}$ output pins to encode CPU status.

Table 1 Status Summary

ST	$\overline{\text{HALT}}$	$\overline{\text{LIR}}$	Operation
0	1	0	CPU operation (1st op-code fetch)
1	1	0	CPU operation (2nd op-code and 3rd op-code fetch)
1	1	1	CPU operation (MC except for op-code fetch)
0	X	1	DMA operation
0	0	0	HALT mode
1	0	1	SLEEP mode (including SYSTEM STOP mode)

NOTE) X: Don't care
MC. Machine cycle

$\overline{\text{REF}}$ – Refresh (OUT)

When LOW, indicates the CPU is in the dynamic RAM refresh cycle and the low-order 8 bits (A_0-A_7) of the address bus contain the refresh address.

$\overline{\text{NMI}}$ – Non-Maskable Interrupt (IN)

When edge transition from HIGH to LOW is detected, forces the CPU to save certain state information and vector to an interrupt service routine at address 0066H. The saved state information is restored by executing the RETN (Return from Non-Maskable Interrupt) instruction.

$\overline{\text{INT}}_0$ – Maskable Interrupt Level 0 (IN)

When LOW, requests a CPU interrupt (unless masked) and saves certain state information unless masked by software. $\overline{\text{INT}}_0$ requests service using one of three software programmable interrupt modes.

Mode	Operation
0	Instruction fetched and executed from data bus.
1	Instruction fetched and executed from address 0038H.
2	Vector System – Low-order 8 bits vector table address fetched from data bus.

In all modes, the saved state information is restored by executing RETI (Return from Interrupt) instruction.

$\overline{INT_1}$, $\overline{INT_2}$ – Maskable Interrupt Level 1, 2 (IN)

When LOW, requests a CPU interrupt (unless masked) and saves certain state information unless masked by software. $\overline{INT_1}$ and $\overline{INT_2}$ (and internally generated interrupts) request interrupt service using a vector system similar to Mode 2 of $\overline{INT_0}$.

$\overline{DREQ_0}$ – DMA Request – Channel 0 (IN)

When LOW (programmable edge or level sensitive), requests DMA transfer service from channel 0 of the HD64180 DMAC. $\overline{DREQ_0}$ is used for Channel 0 memory <—> I/O and memory <—> memory mapped I/O transfers. $\overline{DREQ_0}$ is not used for memory <—> memory transfers. This pin is multiplexed with CKA_0.

$\overline{TEND_0}$ – Transfer End – Channel 0 (OUT)

Asserted LOW synchronous with the last write cycle of channel 0 DMA transfer to indicate DMA completion to an external device. This pin is multiplexed with CKA_1.

$\overline{DREQ_1}$ – DMA Request – Channel 1 (IN)

When LOW (programmable edge or level sense), requests DMA transfer service from channel 1 of the HD64180 DMAC. Channel 1 supports Memory <—> I/O transfers.

$\overline{TEND_1}$ – Transfer End – Channel 1 (OUT)

Asserted LOW synchronous with the last write cycle of channel 1 DMA transfer to indicate DMA completion to an external device.

TXA_0 – Asynchronous Transmit Data – Channel 0 (OUT)

Asynchronous transmit data from channel 0 of the Asynchronous Serial Communication Interface (ASCI).

RXA_0 – Asynchronous Receive Data – Channel 0 (IN)

Asynchronous receive data to channel 0 of the ASCI.

CKA_0 – Asynchronous Clock – Channel 0 (IN/OUT)

Clock input/output for channel 0 of the ASCI. This pin is multiplexed (software selectable) with $\overline{DREQ_0}$.

$\overline{RTS_0}$ – Request to Send – Channel 0 (OUT)

Programmable modem control output signal for channel 0 of the ASCI.

$\overline{CTS_0}$ – Clear to Send – Channel 0 (IN)

Modem control input signal for channel 0 of the ASCI.

$\overline{DCD_0}$ – Data Carrier Detect – Channel 0 (IN)

Modem control input signal for channel 0 of the ASCI.

TXA_1 – Asynchronous Transmit Data – Channel 1 (OUT)

Asynchronous transmit data from channel 1 of the ASCI.

RXA_1 – Asynchronous Receive Data – Channel 1 (IN)

Asynchronous receive data to channel 1 of the ASCI.

CKA_1 – Asynchronous Clock – Channel 1 (IN/OUT)

Clock input/output for channel 1 of the ASCI. This pin is multiplexed (software selectable) with $\overline{TEND_0}$.

$\overline{CTS_1}$ – Clear to Send – Channel 1 (IN)

Modem control input signal for channel 1 of the ASCI. This pin is multiplexed (software selectable) with RXS.

TXS – Clocked Serial Transmit Data (OUT)

Clocked serial transmit data from the Clocked Serial I/O Port (CSI/O).

RXS – Clocked Serial Receive Data (IN)

Clocked serial receive data to the CSI/O. This pin is multiplexed (software selectable) with ASCI channel 1 $\overline{CTS_1}$ modem control input.

CKS – Serial Clock (IN/OUT)

Input or output clock for the CSI/O.

TOUT – Timer Output (OUT)

Pulse output from Programmable Reload Timer channel 1. This pin is multiplexed (software selectable) with A_{18} (Address 18).

V_{CC} – Power Supply

V_{SS} – Ground

Multiplexed pin descriptions

A_{18}/TOUT

During RESET, this pin is initialized as A_{18} pin. If either TOC1 or TOC0 bit in Timer Control Register (TCR) is set to 1, TOUT function is selected.

If TOC1 and TOC0 bits are cleared to 0, A_{18} function is selected.

CKA_0/$\overline{DREQ_0}$

During RESET, this pin is initialized as CKA_0 pin. If either DM1 or SM1 in DMA Mode Register (DMODE) is set to 1, $\overline{DREQ_0}$ function is always selected.

CKA_1/$\overline{TEND_0}$

During RESET, this pin is initialized as CKA_1 pin. If $\overline{CKA1D}$ bit in ASCI control register ch 1 (CNTLA1) is set to 1, $\overline{TEND_0}$ function is selected. If $\overline{CKA1D}$ bit is set to 0, CKA_1 function is selected.

RXS/$\overline{CTS_1}$

During RESET, this pin is initialized as RXS pin. If CTS1E bit in ASCI status register ch1 (STAT1) is set to 1, $\overline{CTS_1}$ function is selected.

If CTS1E bit is set to 0, RXS function is selected.

2 CPU REGISTERS

The HD64180 CPU registers consist of Register Set GR, Register Set GR' and Special Registers.

The Register Set GR consists of 8-bit Accumulator (A), 8-bit Flag Register (F), and three General Purpose Registers (BC, DE, and HL) which may be treated as 16-bit registers (BC, DE, and HL) or as individual 8-bit registers (B, C, D, E, H, and L) depending on the instruction to be executed. The Register Set GR' is alternate register set of Register Set GR and also contains Accumulator (A'), Flag Register (F') and three General Purpose Registers (BC', DE', and HL'). While the alternate Register Set GR' contents are not directly accessible, the contents can be programmably exchanged at high speed with those of Register Set GR.

The Special Registers consist of 8-bit Interrupt Vector Register (I), 8-bit R Counter (R), two 16-bit Index Registers (IX and IY), 16-bit Stack Pointer (SP), and 16-bit Program Counter (PC).

Fig. 8 shows CPU registers configuration.

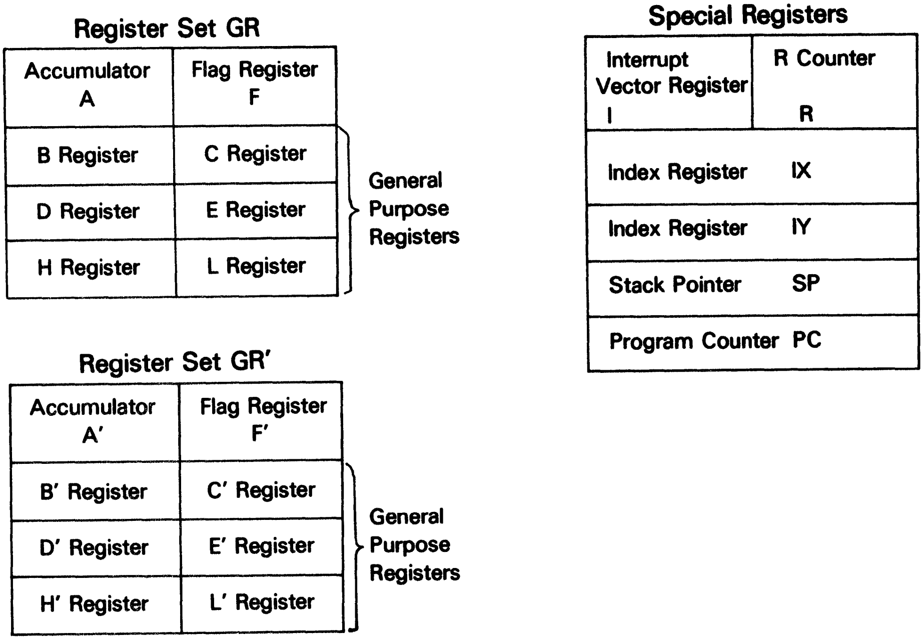


Figure 8 CPU Register Configuration

2.1 Register Description

(1) Accumulator (A, A')

The Accumulator (A) serves as the primary register used for many arithmetic, logical and I/O instructions.

(2) Flag Registers (F, F')

The flag register stores various status bits (described in the next section) which reflect the results of instruction execution.

(3) General Purpose Registers (BC, BC', DE, DE', HL, HL')

The General Purpose Registers are used for both address and data operation. Depending on instruction, each half (8 bits) of these registers (B, C, D, E, H, and L) may also be used.

(4) Interrupt Vector Register (I)

For interrupts which require a vector table address to be calculated ($\overline{\text{INT}_0}$ Mode 2, $\overline{\text{INT}_1}$, $\overline{\text{INT}_2}$ and internal interrupts), the Interrupt Vector Register (I) provides the most significant byte of the vector table address.

(5) R Counter (R)

The least significant seven bits of the R Counter (R) serve to count the number of instructions executed by the HD64180. R is incremented for each CPU op-code fetch cycles (each LIR cycles).

(6) Index Registers (IX, and IY)

The Index Registers are used for both address and data operations. For addressing, the contents of a displacement specified in the instruction are added to or subtracted from the Index Register to determine an effective operand address.

(7) Stack Pointer (SP)

The Stack Pointer (SP) contains the memory address based LIFO stack.

(8) Program Counter (PC)

The Program Counter (PC) contains the address of the instruction to be executed and is automatically updated after each instruction fetch.

(9) Flag Register (F)

The Flag Register stores the logical state reflecting the results of instruction execution. The contents of the Flag Register are used to control program flow and instruction operation.

bit	7	6	5	4	3	2	1	0	
	S	Z	–	H	–	P/V	N	C	Flag Register (F)

S: Sign (bit 7)

S stores the state of the most significant bit (bit 7) of the result. This is useful for operations with signed numbers in which values with bit 7 = 1 are interpreted as negative.

Z: Zero (bit 6)

Z is set to 1 when instruction execution results containing 0. Otherwise, Z is reset to 0.

H: Half Carry (bit 4)

H is used by the DAA (Decimal Adjust Accumulator) instruction to reflect borrow or carry from the least significant 4 bits and thereby adjust the results of BCD addition and subtraction.

P/V: Parity/Overflow (bit 2)

P/V serves a dual purpose. For logical operations P/V is set to 1 if the number of 1 bit in the result is even and P/V is reset to 0 if the number of 1 bit in the result is odd. For two complement arithmetic, P/V is set to 1 if the operation produces a result which is outside the allowable range (+127 to −128 for 8-bit operations, +32767 to −32768 for 16-bit operations).

N: Negative (bit 1)

N is set to 1 if the last arithmetic instruction was a subtract operation (SUB, DEC, CP, etc.) and N is reset to 0 if the last arithmetic instruction was an addition operation (ADD, INC, etc.).

C: Carry (bit 0)

C is set to 1 when a carry (addition) or borrow (subtraction) from the most significant bit of the result occurs. C is also affected by Accumulator logic operations such as shifts and rotates.

3 ADDRESSING MODES

The HD64180 instruction set includes eight addressing modes.

Implied Register
Register Direct
Register Indirect
Indexed
Extended
Immediate
Relative
IO

(1) Implied Register (IMP)

Certain op-codes automatically imply register usage, such as the arithmetic operations which inherently reference the Accumulator, Index Registers, Stack Pointer and General Purpose Registers.

(2) Register Direct (REG)

Many op-codes contain bit fields specifying registers to be used for the operation. The exact bit field definition vary depending on instruction as follows.

8-bit Register

g or g' field	Register
0 0 0	B
0 0 1	C
0 1 0	D
0 1 1	E
1 0 0	H
1 0 1	L
1 1 0	–
1 1 1	A

16-bit Register

ww field	Register
0 0	B C
0 1	D E
1 0	H L
1 1	S P

xx field	Register
0 0	B C
0 1	D E
1 0	I X
1 1	S P

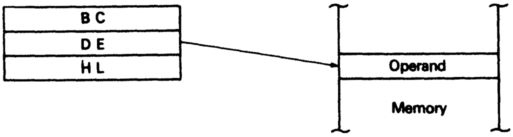
zz field	Register
0 0	B C
0 1	D E
1 0	H L
1 1	A F

yy field	Register
0 0	B C
0 1	D E
1 0	I Y
1 1	S P

Suffixed H and L to ww,xx,yy,zz (ex. wwH,IXL) indicate upper and lower 8-bit of the 16-bit register respectively.

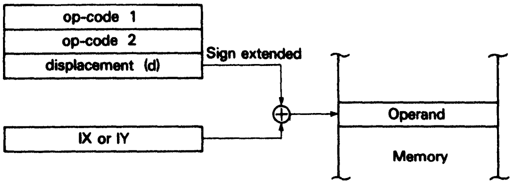
(3) Register Indirect (REG)

The memory operand address is contained in one of the 16-bit General Purpose Registers (BC, DE and HL).



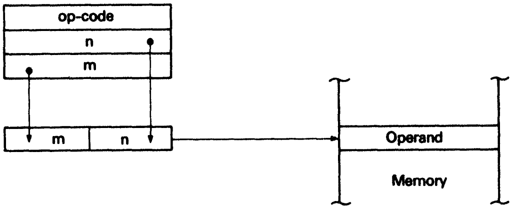
(4) Indexed (INDX)

The memory operand address is calculated using the contents of an Index Register (IX or IY) and an 8-bit signed displacement specified in the instruction.



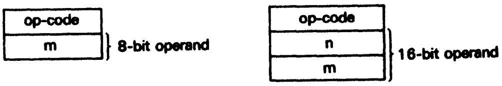
(5) Extended (EXT)

The memory operand address is specified by two bytes contained in the instruction.



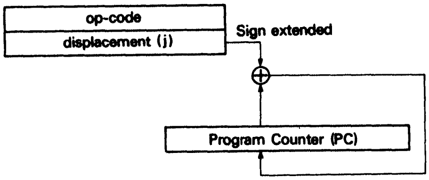
(6) Immediate (IMMED)

The memory operands are contained within one or two bytes of the instruction.



(7) Relative (REL)

Relative addressing mode is only used by the conditional and unconditional branch instructions. The branch displacement (relative to the contents of the program counter) is contained in the instruction.



(8) IO (IO)

IO addressing mode is used only by I/O instructions. This mode specifies I/O address ($\overline{IOE}$ = 0) and outputs them as follows.

(1) An operand is output to A_0-A_7. The Contents of Accumulator is output to A_8-A_{15}.

(2) The Contents of Register B is output to A_0-A_7. The Contents of Register C is output to A_8-A_{15}.

(3) An operand is output to A_0-A_7. 00H is output to A_8-A_{15}. (useful for internal I/O register access)

(4) The Contents of Register C is output to A_0-A_7. 00H is output to A_8-A_{15}. (useful for internal I/O register access)

■ CPU BUS TIMING

This section explains the HD64180 CPU timing for the following operations.

(1) Instruction (op-code) fetch timing.
(2) Operand and data read/write timing.
(3) I/O read/write timing.
(4) Basic instruction (fetch and execute) timing.
(5) RESET timing.
(6) BUSREQ/BUSACK bus exchange timing.

The basic CPU operation consists of one or more "machine cycles" (MC). A machine cycle consists of three system clocks, T_1, T_2 and T_3 while accessing memory or I/O, or it consists of one system clock, Ti while the CPU internal operation. The system clock (ϕ) is half frequency of crystal oscillation (Ex. 8 MHz crystal ⟶ ϕ of 4 MHz, 250 nsec). For interfacing to slow memory or peripherals, optional wait states (Tw) may be inserted between T_2 and T_3.

• Instruction (op-code) Fetch Timing

Fig. 9 shows the instruction (op-code) fetch timing with no wait states.

An op-code fetch cycle is externally indicated when the LIR (Load Instruction Register) output pin is LOW.

In the first half of T_1, the address bus (A_0-A_{18}) is driven with the contents of the Program Counter (PC). Note that this is the translated address output of the HD64180 on-chip MMU.

In the second half of T_1, the ME (Memory Enable) and RD (Read) signals are asserted LOW, enabling the memory.

The op-code on the data bus is latched at the rising edge of T_3 and the bus cycle terminates at the end of T_3.

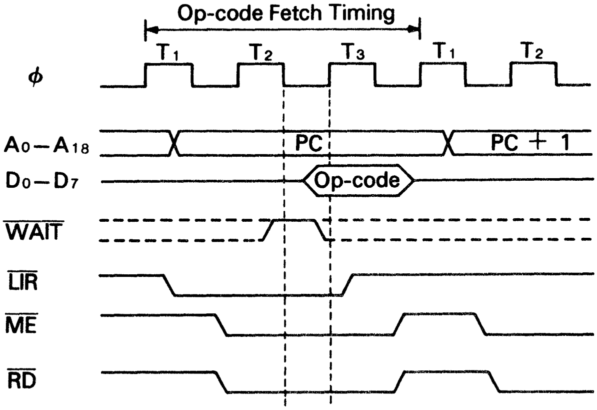


Figure 9 Op-Code Fetch Timing

Fig. 10 illustrates the insertion of wait states (Tw) into the op-code fetch cycle. Wait states (Tw) are controlled by the external WAIT input combined with an on-chip programmable wait state generator.

At the falling edge of T_2 the combined WAIT input is sampled. If WAIT input is asserted LOW, a wait state (Tw) is inserted. The address bus, ME, RD and LIR are held stable during wait states. When the WAIT is sampled inactive HIGH at the falling edge of Tw, the bus cycle enters T_3 and completes at the end of T_3.

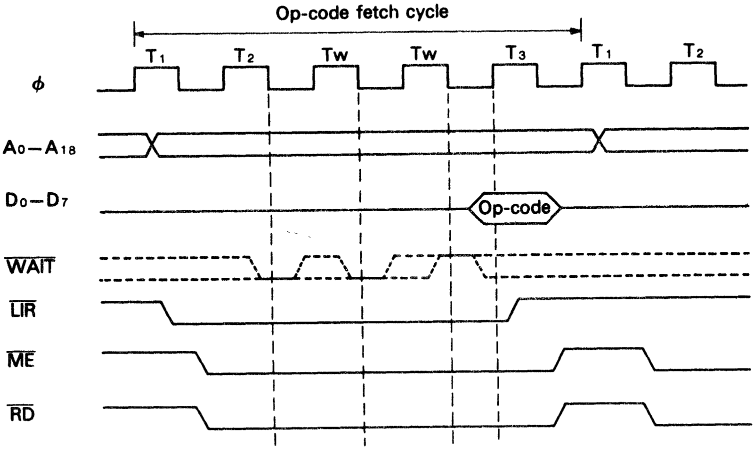


Figure 10 Op-Code Fetch Timing (with wait state)

- **Operand and Data Read/Write Timing**

The instruction operand and data read/write timing differs from op-code fetch timing in two ways. First, the $\overline{LIR}$ output is held inactive. Second, the read cycle timing is relaxed by one-half clock cycle since data is latched at the falling edge of T_3.

Instruction operands include immediate data, displacement and extended addresses and have the same timing as memory data reads.

During memory write cycles the $\overline{ME}$ signal goes active in the second half of T_1. At the end of T_1, the data bus is driven with the write data.

At the start of T_2, the $\overline{WR}$ signal is asserted LOW enabling the memory. $\overline{ME}$ and $\overline{WR}$ go inactive in the second half of T_3 followed by deactivation of the write data on the data bus.

Wait states (Tw) are inserted as previously described for op-code fetch cycles.

Fig. 11 illustrates the read/write timing without wait states (Tw), while Fig. 12 illustrates read/write timing with wait states (Tw).

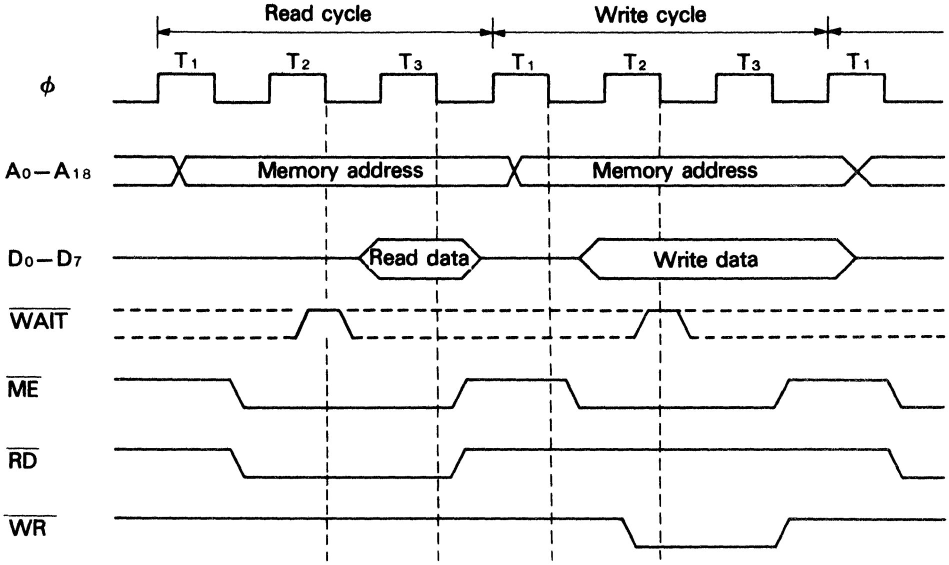


Figure 11 Memory Read/Write Timing (without wait state)

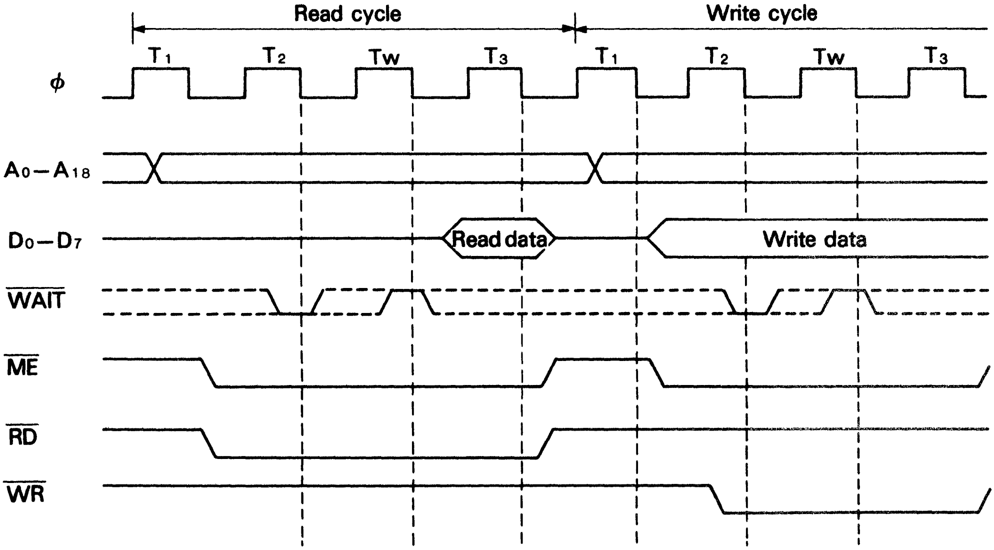


Figure 12 Memory Read/Write Timing (with wait state)

4.3 I/O Read/Write Timing

I/O instructions cause data read/write transfer which differs from memory data transfer in the following three ways. The $\overline{IOE}$ (I/O Enable) signal is asserted LOW instead of the $\overline{ME}$ signal. The 16-bit I/O address is not translated by the MMU and A_{16}-A_{18} are held LOW. At least one wait state (Tw) is always inserted for I/O read and write cycles (except internal I/O cycles).

Fig. 13 shows I/O read/write timing with the automatically inserted wait state (Tw).

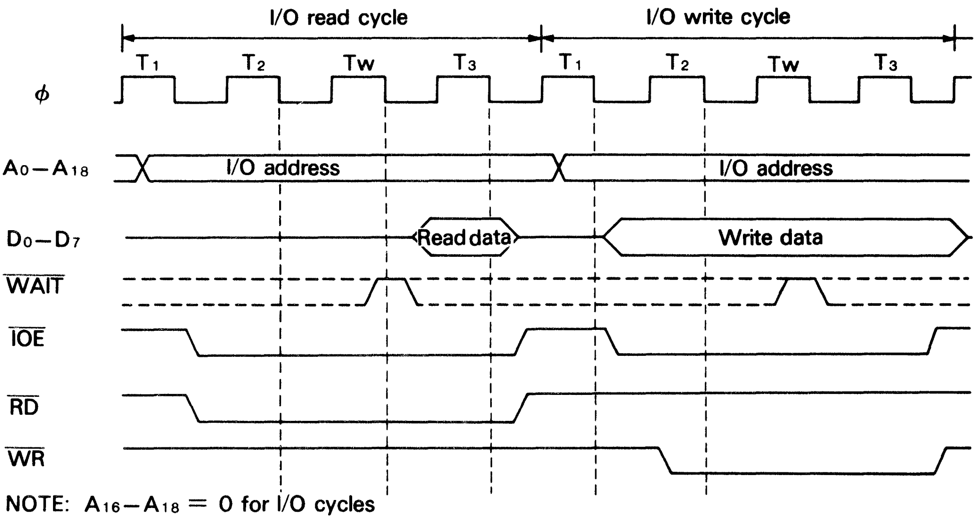


Figure 13 I/O Read/Write Timing

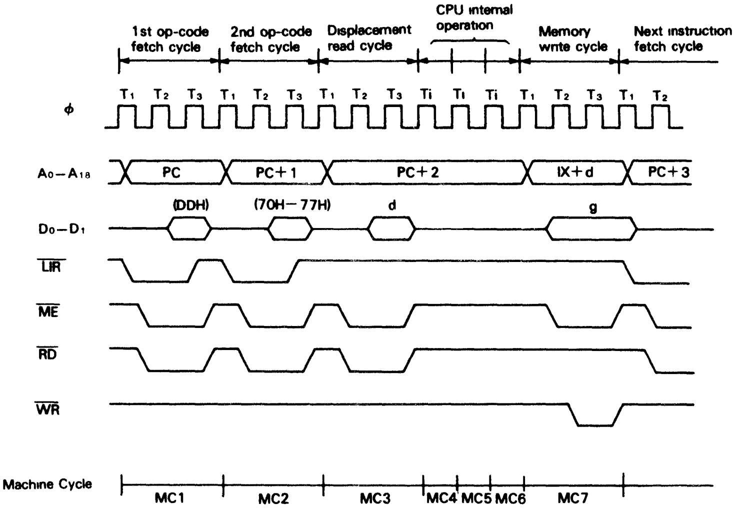
4.4 Basic Instruction Timing

An instruction may consist of a number of machine cycles including op-code fetch, operand fetch and data read/write cycles. An instruction may also include cycles for internal processing in which case the bus is idle.

The example in Fig. 14 illustrates the bus timing for the data transfer instruction LD (IX+d),g. This instruction moves the contents of a CPU register (g) to the memory location with address computed by adding an signed 8-bit displacement (d) to the contents of an index register (IX).

The instruction cycle starts with the two machine cycles to read the two bytes instruction op-code as indicated by $\overline{LIR}$ LOW. Next, the instruction operand (d) is fetched.

The external bus is idle while the CPU computes the effective address. Finally, the computed memory location is written with the contents of the CPU register (g).



NOTE d = displacement
g = register contents

Figure 14 LD (IX+d), g Instruction Timing

4.5 RESET Timing

Fig. 15 shows the HD64180 hardware RESET timing. If the RESET pin is LOW for at least six clock cycles, processing is terminated and the HD64180 restarts execution from (logical and physical) address 00000H.

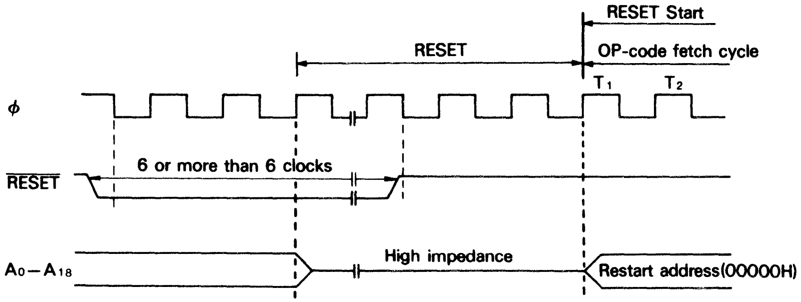


Figure 15 RESET Timing

4.6 $\overline{\text{BUSREQ}}$/$\overline{\text{BUSACK}}$ Bus Exchange Timing

The HD64180 can coordinate the exchange of control, address and data bus ownership with another bus master. The alternate bus master can request the bus release by asserting the $\overline{\text{BUSREQ}}$ (Bus Request) input LOW. After the HD64180 releases the bus, it relinquishes control to the alternate bus master by asserting the $\overline{\text{BUSACK}}$ (Bus Acknowledge) output LOW.

The bus may be released by the HD64180 at the end of each machine cycle. In this context a machine cycle consists of a minimum of 3 clock cycles (more if wait states are inserted) for op-code fetch, memory read/write and I/O read/write cycles. Except for these cases, a machine cycle corresponds to one clock cycle.

When the bus is released, the address (A_0-A_{18}), data (D_0-D_7) and control ($\overline{\text{ME}}$, $\overline{\text{IOE}}$, $\overline{\text{RD}}$, and $\overline{\text{WR}}$) signals are placed in the high impedance state.

Note that dynamic RAM refresh is not performed when the HD64180 has released the bus. The alternate bus master must provide dynamic memory refreshing if the bus is released for long periods of time.

Fig. 16 illustrates $\overline{\text{BUSREQ}}$/$\overline{\text{BUSACK}}$ bus exchange during a memory read cycle. Fig. 17 illustrates bus exchange when the bus release is requested during an HD64180 CPU internal operation. $\overline{\text{BUSREQ}}$ is sampled at the falling edge of the system clock prior to T_3, Ti and Tx (BUS RELEASE state). If $\overline{\text{BUSREQ}}$ is asserted LOW at the falling edge of the clock state prior to Tx, another Tx is executed.

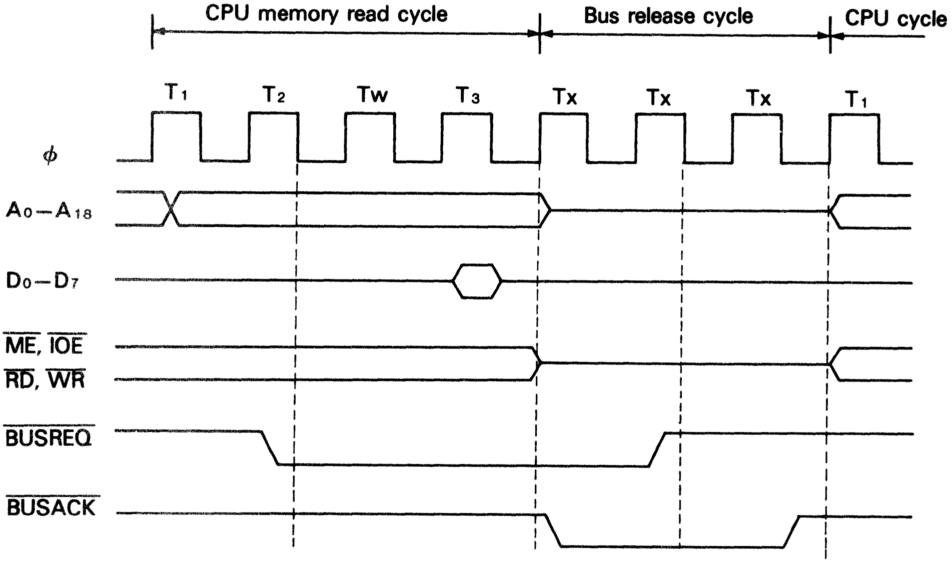


Figure 16 Bus Exchange Timing (1)

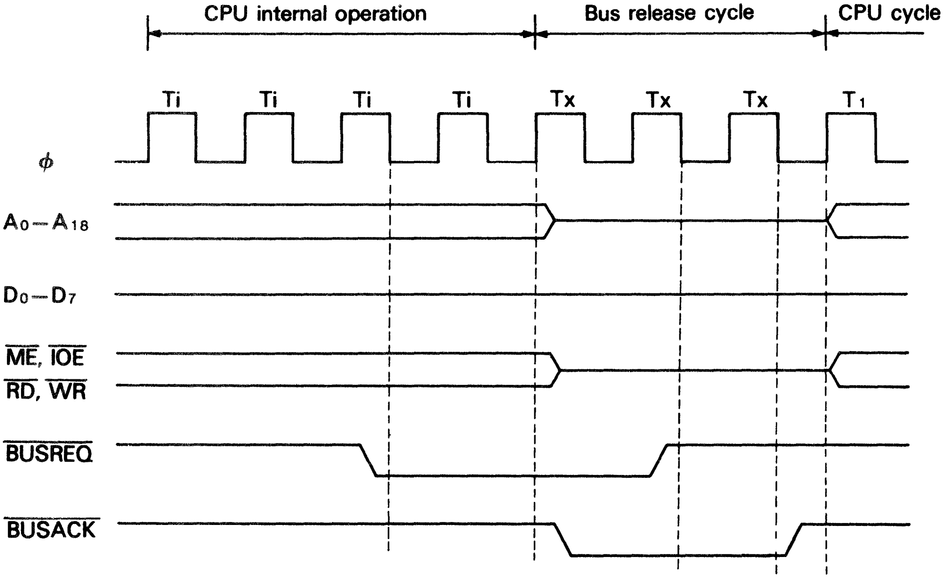


Figure 17 Bus Exchange Timing (2)

5 HALT AND LOW POWER OPERATION MODES

The HD64180 can operate in 4 different modes. HALT mode, IOSTOP mode and two low power operation modes – SLEEP and SYSTEM STCP. Note that in all operating modes, the basic CPU clock (XTAL, EXTAL) must remain active.

5.1 HALT Mode

HALT mode is entered by execution of the HALT instruction (op-code = 76H) and has the following characteristics.

(1) The internal CPU clock remains active.
(2) All internal and external interrupts can be received.
(3) Bus exchange ($\overline{\mathrm{BUSREQ}}$ and $\overline{\mathrm{BUSACK}}$) can occur.
(4) Dynamic RAM refresh cycle ($\overline{\mathrm{REF}}$) insertion continues at the programmed interval.
(5) I/O operations (ASCI, CSI/O and PRT) continue.
(6) The DMAC can operate.
(7) The $\overline{\mathrm{HALT}}$ output pin is asserted LOW.
(8) The external bus activity consists of repeated 'dummy' fetches of the op-code following the HALT instruction.

Essentially, the HD64180 operates normally in HALT mode, except that instruction execution is stopped.

HALT mode can be exited in the following two ways.

RESET Exit from HALT Mode

If the $\overline{\mathrm{RESET}}$ input is asserted LOW for at least six clock cycles, HALT mode is exited and the normal RESET sequence (restart at address 00000H) is initiated.

Interrupt Exit from HALT Mode

When an internal or external interrupt is generated, HALT mode is exited and the normal interrupt response sequence is initiated.

If the interrupt source is masked (individually by enable bit, or globally by IEF_1 state), the HD64180 remains in HALT mode. However, $\overline{\mathrm{NMI}}$ interrupt will initiate the normal $\overline{\mathrm{NMI}}$ interrupt response sequence independent of the state of IEF_1.

HALT timing is shown in Fig. 18

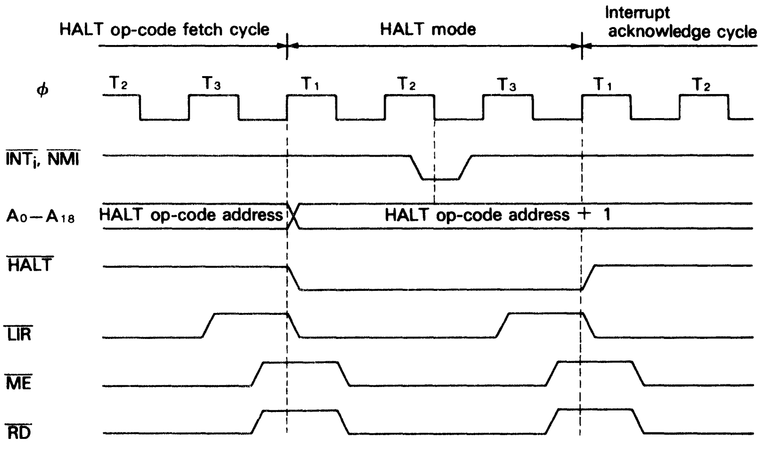


Figure 18 HALT Timing

5.2 SLEEP Mode

SLEEP mode is entered by execution of the 2 byte SLP instruction. SLEEP mode has the following characteristics

(1) The internal CPU clock stops, reducing power consumption.
(2) The internal crystal oscillator does not stop.
(3) Internal and external interrupt inputs can be received.
(4) DRAM refresh cycles stop.
(5) I/O operations using on-chip peripherals continue.
(6) The internal DMAC stop.
(7) $\overline{\mathrm{BUSREQ}}$ can be received and acknowledged.
(8) Address outputs go HIGH and all other control signal output become inactive HIGH.
(9) Data Bus, 3-state.

SLEEP mode is exited in one of two ways as shown below.

RESET Exit from SLEEP Mode

If the $\overline{\mathrm{RESET}}$ input is held LOW for at least six clock cycles, the HD64180 will exit SLEEP mode and begin the normal RESET sequence with execution starting at address (logical and physical) 00000H.

Interrupt Exit from SLEEP Mode

The SLEEP mode is exited by detection of an external ($\overline{\mathrm{NMI}}$, $\overline{\mathrm{INT}_0}$, $\overline{\mathrm{INT}_1}$, $\overline{\mathrm{INT}_2}$) or internal (ASCI, CSI/O, PRT) interrupt.

In the case of $\overline{\mathrm{NMI}}$, SLEEP Mode is exited and the CPU begins the normal $\overline{\mathrm{NMI}}$ interrupt response sequence.

In the case of all other interrupts, the interrupt response depends on the state of the global interrupt enable flag (IEF_1) and the individual interrupt source enable bit.

If the individual interrupt condition is disabled by the corresponding enable bit, occurrence of that interrupt is ignored and the CPU remains in the SLEEP state.

Assuming the individual interrupt condition is enabled, the response to that interrupt depends on the global interrupt enable flag (IEF_1). If interrupts are globally enabled ($IEF_1 = 1$) and an individually enabled interrupt occurs, SLEEP mode is exited and the appropriate normal interrupt response sequence is executed.

If interrupts are globally disabled ($IEF_1 = 0$) and an individually enabled interrupt occurs, SLEEP mode is exited and instruction execution begins with the instruction following the SLP instruction. Note that this provides a technique for synchronization with high speed external events without incurring the latency imposed by an interrupt response sequence.

Fig. 19 shows SLEEP timing.

5.3 IOSTOP Mode

IOSTOP mode is entered by setting the IOSTP bit of the I/O Control Register (ICR) to 1. In this case, on-chip I/O (ASCI, CSI/O, PRT) stops operating. However, the CPU continues to operate. Recovery from IOSTOP mode is by clearing the IOSTP bit in ICR to 0.

5.4 SYSTEM STOP Mode

SYSTEM STOP mode is the combination of SLEEP and IOSTOP modes. SYSTEM STOP mode is entered by setting the IOSTP bit in ICR to 1 followed by execution of the SLP instruction. In this mode, on-chip I/O and CPU stop operating, reducing power consumption. Recovery from SYSTEM STOP mode is the same as recovery from SLEEP mode, noting that internal I/O sources (disabled by IOSTOP) cannot generate a recovery interrupt.

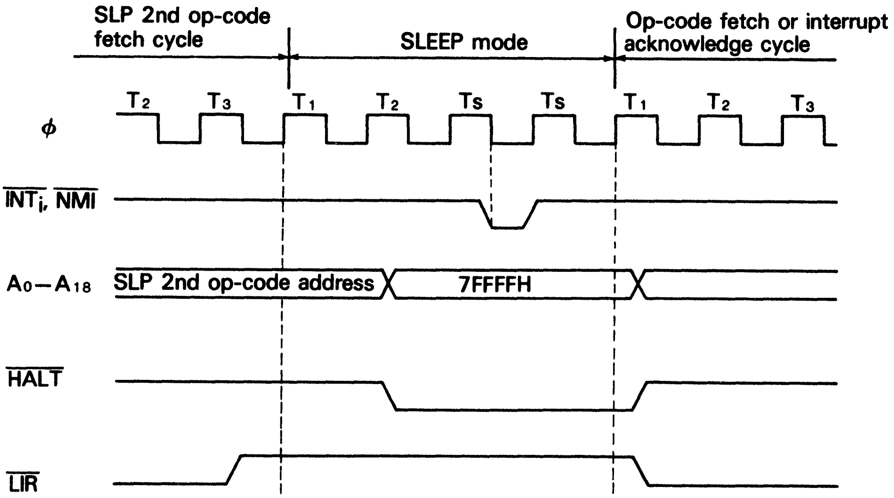


Figure 19 SLEEP Timing

6 INTERRUPTS

The HD64180 CPU has twelve interrupt sources, four external and eight internal, with fixed priority.

This section explains the CPU registers associated with interrupt processing, the TRAP interrupt, interrupt response modes and the external interrupts. The detailed discussion of internal interrupt generation (except TRAP) is presented in the appropriate hardware section (i.e. PRT, DMAC, ASCI and CSI/O).

Priority		Interrupt	
Higher Priority	1	TRAP (Undefined Op-code Trap)	...... Internal Interrupt
↑	2	$\overline{NMI}$ (Non Maskable Interrupt)	External Interrupt
	3	$\overline{INT}_0$ (Maskable Interrupt Level 0)	
	4	$\overline{INT}_1$ (Maskable Interrupt Level 1)	
	5	$\overline{INT}_2$ (Maskable Interrupt Level 2)	
	6	Timer 0	Internal Interrupt
	7	Timer 1	
	8	DMA channel 0	
	9	DMA channel 1	
↓	10	Clocked Serial I/O Port	
Lower	11	Asynchronous SCI channel 0	
Priority	12	Asynchronous SCI channel 1	

Figure 20 Interrupt Sources

6.1 Interrupt Control Registers and Flags

The HD64180 contains three registers and two flags which are associated with interrupt processing.

Register and Flag Name	Function	Access Method
I	Contains upper 8-bit of interrupt vector	LD A, I and LD I, A instructions
IL	Contains lower 8-bit of interrupt vector	I/O instruction (addr = 33H)
ITC	Interrupt/Trap control	I/O instruction (addr = 34H)
IEF_1, IEF_2	Enable/disable interrupt	EI, DI, LD A, I, and LD A, R instructions

(1) Interrupt Vector Register (I)

Mode 2 for $\overline{INT}_0$ external interrupt, $\overline{INT}_1$ and $\overline{INT}_2$ external interrupts and all internal interrupts (except TRAP) use a programmable vectored technique to determine the address at which interrupt processing starts. In response to the interrupt a 16-bit address is generated. This address accesses a vector table in memory to obtain the address at which execution restarts.

While the method for generation of the least significant byte of the table address differs, all vectored interrupts use the contents of I as the most significant byte of the table address. By programming the contents of I, vector tables can be relocated on 256 bytes boundaries throughout the 64k bytes logical address space.

Note that I is read/written with the LD A, I and LD I, A instructions rather than I/O (IN, OUT) instructions.

I is initialized to 00H during RESET.

(2) Interrupt Vector Low Register (IL)

Interrupt Vector Low Register (IL I/O Address = 33H)

bit	7	6	5	4	3	2	1	0
	IL7	IL6	IL5	–	–	–	–	–
	R/W	R/W	R/W					
	Programmable			Interrupt Source Dependent Code				

This register determines the most significant three bits of the low-order byte of the interrupt vector table address for external interrupts $\overline{INT}_1$ and $\overline{INT}_2$ and all internal interrupts (except TRAP). The five least significant bits are fixed for each specific interrupt source. By programming IL the vector table can be relocated on 32 bytes boundaries

IL is initialized to 00H during RESET

(3) INT/TRAP Control Register (ITC)

INT/TRAP Control Register (ITC I/O Address = 34H)

bit	7	6	5	4	3	2	1	0
	TRAP	UFO	–	–	–	ITE2	ITE1	ITE0
	R/W	R				R/W	R/W	R/W

ITC is used to handle TRAP interrupts and to enable or disable the external maskable interrupt inputs $\overline{INT}_0$, $\overline{INT}_1$, and $\overline{INT}_2$.

TRAP (bit 7)

This bit is set to 1 when an undefined op-code is fetched TRAP can be reset under program control by writing it with 0, however it cannot be written with 1 under program control. TRAP is cleared to 0 during RESET.

UFO: Undefined Fetch Object (bit 6)

When a TRAP interrupt occurs (TRAP bit is set to 1), the contents of UFO allow determination of the starting address of the undefined instruction. This is necessary since the TRAP may occur on either the second or third byte of the op-code. UFO allows the stacked PC value (stacked in response to TRAP) to be correctly adjusted. If UFO = 0, the first op-code should be interpreted as the stacked PC−1. If UFO = 1, the first op-code address is stacked PC−2. UFO is read-only

ITE2,1,0: Interrupt Enable 2,1,0 (bits 2-0)

ITE2, ITE1 and ITE0 enable and disable the external interrupt inputs $\overline{INT}_2$, $\overline{INT}_1$, and $\overline{INT}_0$ respectively. If cleared to 0, the interrupt is masked. During RESET, ITE0 is initialized to 1 while ITE1 and ITE2 are initialized to 0

· **Interrupt Enable Flag 1,2 (IEF_1, IEF_2)**

IEF_1 controls the overall enabling and disabling of all internal and external maskable interrupts (i e all interrupts except $\overline{NMI}$ and

TRAP).

If $IEF_1 = 0$, all maskable interrupts are disabled. IEF_1 can be reset to 0 by the DI (Disable Interrupts) instruction and set to 1 by the EI (Enable Interrupts) instruction.

The purpose of IEF_2 is to correctly manage the occurrence of $\overline{\text{NMI}}$. During $\overline{\text{NMI}}$, the prior interrupt reception state is saved and all maskable interrupts are automatically disabled (IEF_1 copied to IEF_2 and then IEF_1 cleared to 0). At the end of the $\overline{\text{NMI}}$ interrupt service routine, execution of the RETN (Return from Non-maskable Interrupt) will automatically restore the interrupt receiving state (by copying IEF_2 to IEF_1) prior to the occurrence of $\overline{\text{NMI}}$.

IEF_2 state can be reflected in the P/V bit of the CPU Status register by executing LD A, I or LD A, R instructions.

Table 2 shows the state of IEF_1 and IEF_2.

Table 2 State of IEF_1 and IEF_2

CPU Operation	IEF_1	IEF_2	REMARKS
RESET	0	0	Inhibits the interrupt except $\overline{\text{NMI}}$ and TRAP.
$\overline{\text{NMI}}$	0	IEF_1	Copies the contents of IEF_1 to IEF_2.
RETN	IEF_2	not affected	Returns from the $\overline{\text{NMI}}$ service routine.
Interrupt except $\overline{\text{NMI}}$ and TRAP	0	0	Inhibits the interrupt except $\overline{\text{NMI}}$ and TRAP
RETI	not affected	not affected	
TRAP	not affected	not affected	
EI	1	1	
DI	0	0	
LD A, I	not affected	not affected	Transfers the contents of IEF_2 to P/V flag
LD A, R	not affected	not affected	Transfers the contents of IEF_2 to P/V flag.

6.2 TRAP Interrupt

The HD64180 generates a non-maskable (not affected by the state of IEF_1) TRAP interrupt when an undefined op-code fetch occurs. This feature can be used to increase software reliability, implement an 'extended' instruction set, or both. TRAP may occur during op-code fetch cycles and also if an undefined op-code is fetched during the interrupt acknowledge cycle for $\overline{\text{INT}_0}$ when Mode 0 is used.

When a TRAP interrupt occurs the HD64180 operates as follows.

(1) The TRAP bit in the Interrupt TRAP/Control (ITC) register is set to 1.

(2) The current PC (Program Counter) value, reflecting the location of the undefined op-code, is saved on the stack.

(3) The HD64180 vectors to logical address 0. Note that if logical address 0000H is mapped to physical address 00000H, the vector is the same as for RESET. In this case, testing the TRAP bit in ITC will reveal whether the restart at physical address 00000H was caused by RESET or TRAP.

The state of the UFO (Undefined Fetch Object) bit in ITC allows TRAP manipulation software to correctly 'adjust' the stacked PC depending on whether the second or third byte of the op-code generated the TRAP. If UFO = 0, the starting address of the invalid instruction is equal to the stacked PC−1. If UFO = 1, the starting address of the invalid instruction is equal to the stacked PC−2. Fig. 21 shows TRAP Timing.

Note that Bus Release cycle, Refresh cycle, DMA cycle and WAIT cycle can't be inserted just after T_{TP} state which is inserted for TRAP interrupt sequence.

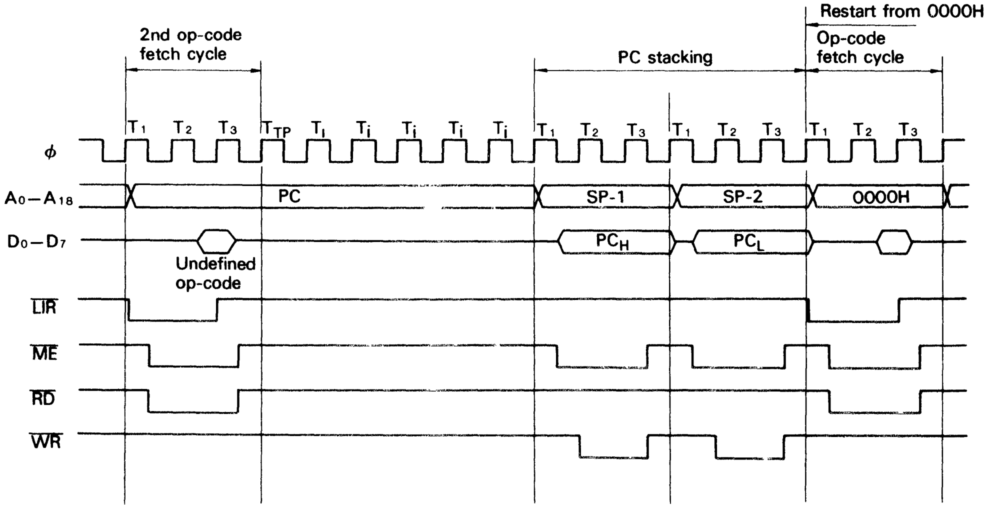


Figure 21 (a) TRAP Timing — 2nd Op-code Undefined

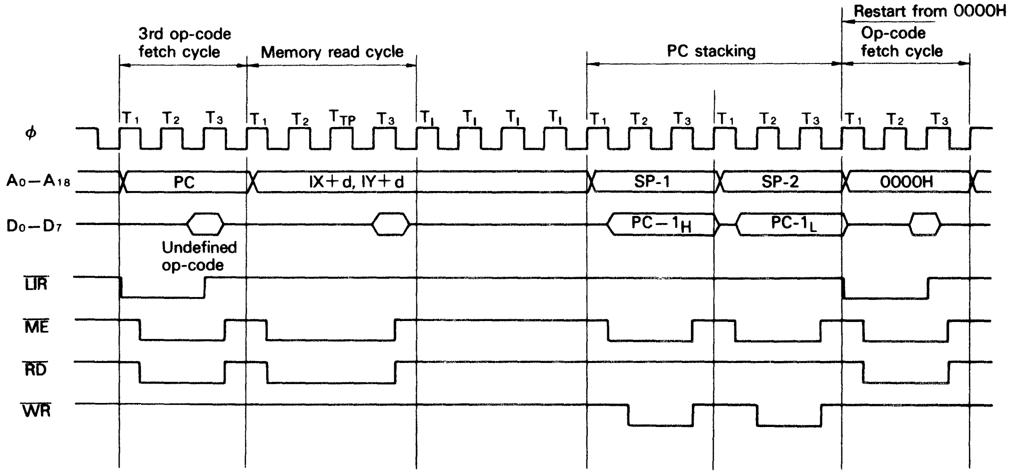


Figure 21 (b) TRAP Timing — 2nd Op-code Undefined

6.3 External Interrupts

The HD64180 has four external hardware interrupt inputs.

(1) $\overline{\text{NMI}}$ – Non-maskable Interrupt
(2) $\overline{\text{INT}_0}$ – Maskable Interrupt Level 0
(3) $\overline{\text{INT}_1}$ – Maskable Interrupt Level 1
(4) $\overline{\text{INT}_2}$ – Maskable Interrupt Level 2

$\overline{\text{NMI}}$, $\overline{\text{INT}_1}$ and $\overline{\text{INT}_2}$ have fixed interrupt response modes. $\overline{\text{INT}_0}$ has three different software programmable interrupt response modes – Mode 0, Mode 1 and Mode 2.

6.4 $\overline{\text{NMI}}$ – Non-Maskable Interrupt

The $\overline{\text{NMI}}$ interrupt input is edge sensitive and cannot be masked by software. When $\overline{\text{NMI}}$ is detected, the HD64180 operates as follows.

(1) DMAC operation is suspended by clearing the DME (DMA Main Enable) bit in DCNTL.
(2) The PC is pushed onto the stack.
(3) The contents of IEF$_1$ are copied to IEF$_2$. This saves the interrupt reception state that existed prior to $\overline{\text{NMI}}$.
(4) IEF$_1$ is cleared to 0. This disables all external and internal maskable interrupts (i.e. all interrupts except $\overline{\text{NMI}}$ and TRAP).
(5) Execution commences at logical address 0066H.

The last instruction of an $\overline{\text{NMI}}$ service routine should be RETN (Return from Non-maskable Interrupt). This restores the stacked PC, allowing the interrupted program to continue. Furthermore, RETN causes IEF$_2$ to be copied to IEF$_1$, restoring the interrupt reception state that existed prior to the $\overline{\text{NMI}}$.

Note that $\overline{\text{NMI}}$, since it can be accepted during HD64180 on-chip DMAC operation, can be used to externally interrupt DMA transfer. The $\overline{\text{NMI}}$ service routine can reactivate or abort the DMAC operation as required by the application.

For $\overline{\text{NMI}}$, special care must be taken to insure that interrupt inputs do not 'overrun' the $\overline{\text{NMI}}$ service routine. Unlimited $\overline{\text{NMI}}$ inputs without a corresponding number of RETN instructions will eventually cause stack overflow.

Fig. 22 shows the use of $\overline{\text{NMI}}$ and RETN while Fig. 23 details $\overline{\text{NMI}}$ response timing. $\overline{\text{NMI}}$ is edge sensitive and the internally latched $\overline{\text{NMI}}$ falling edge is held until it is sampled. If the falling edge of $\overline{\text{NMI}}$ is latched before the falling edge of clock state prior to T$_3$ or Ti in the last machine cycle, the internally latched $\overline{\text{NMI}}$ is sampled at the falling edge of the clock state prior to T$_3$ or Ti in the last machine cycle and $\overline{\text{NMI}}$ acknowledge cycle begins at the end of the current machine cycle.

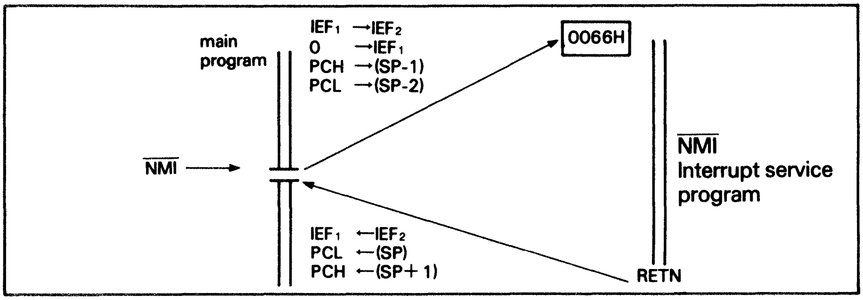


Figure 22 $\overline{\text{NMI}}$ Sequence

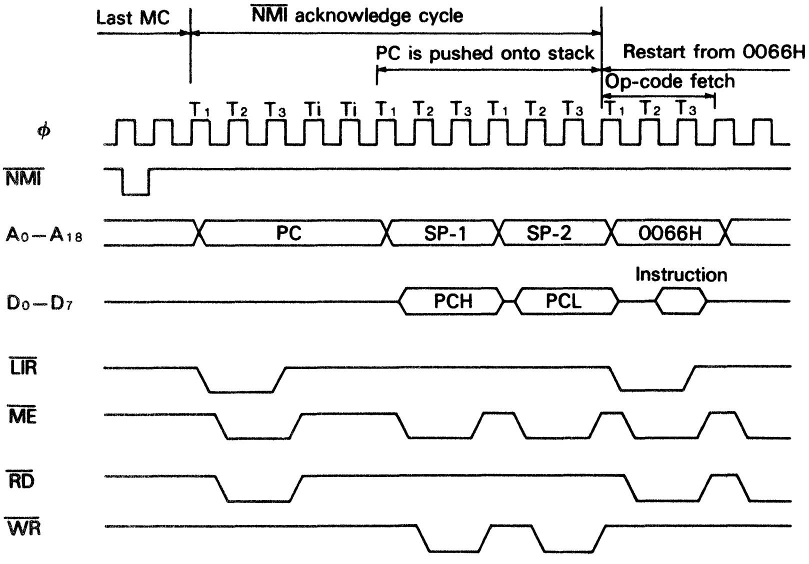


Figure 23 $\overline{\text{NMI}}$ Timing

6.5 $\overline{INT_0}$ – Maskable Interrupt Level 0

The next highest priority external interrupt after $\overline{NMI}$ is $\overline{INT_0}$. $\overline{INT_0}$ is sampled at the falling edge of the clock state prior to T_3 or Ti in the last machine cycle. If $\overline{INT_0}$ is asserted LOW at the falling edge of the clock state prior to T_3 or Ti in the last machine cycle, $\overline{INT_0}$ is accepted. The interrupt is masked if either the IEF_1 flag or the ITE0 (Interrupt Enable 0) bit in ITC are cleared to 0. Note that after RESET the state is as follows.

(1) IEF_1 is 0, so $\overline{INT_0}$ is masked.

(2) ITE0 is 1, so $\overline{INT_0}$ is enabled by execution of the EI (Enable Interrupts) instruction.

The $\overline{INT_0}$ interrupt is unique in that three programmable interrupt response modes are available – Mode 0, Mode 1, and Mode 2. The specific mode is selected with the IM 0, IM 1 and IM 2 (Set Interrupt Mode) instructions. During RESET, the HD64180 is initialized to use Mode 0 for $\overline{INT_0}$.

The three interrupt response modes for $\overline{INT_0}$ are...

(1) Mode 0 – Instruction fetch from data bus.

(2) Mode 1 – Restart at logical address 0038H.

(3) Mode 2 – Low byte vector table address fetch from data bus.

$\overline{INT_0}$ Mode 0

During the interrupt acknowledge cycle, an instruction is fetched from the data bus (D_0-D_7) at the rising edge of T_3. Often, this instruction is one of the eight single byte RST (RESTART) instructions which stack the PC and restart execution at a fixed logical address. However, multibyte instructions can be processed if the interrupt acknowledging device can provide a multibyte response. Unlike all other interrupts, the PC is not automatically stacked.

Note that TRAP interrupt will occur if an invalid instruction is fetched during $\overline{INT_0}$ Mode 0 interrupt acknowledge.

Fig. 24 shows $\overline{INT_0}$ Mode 0 Timing.

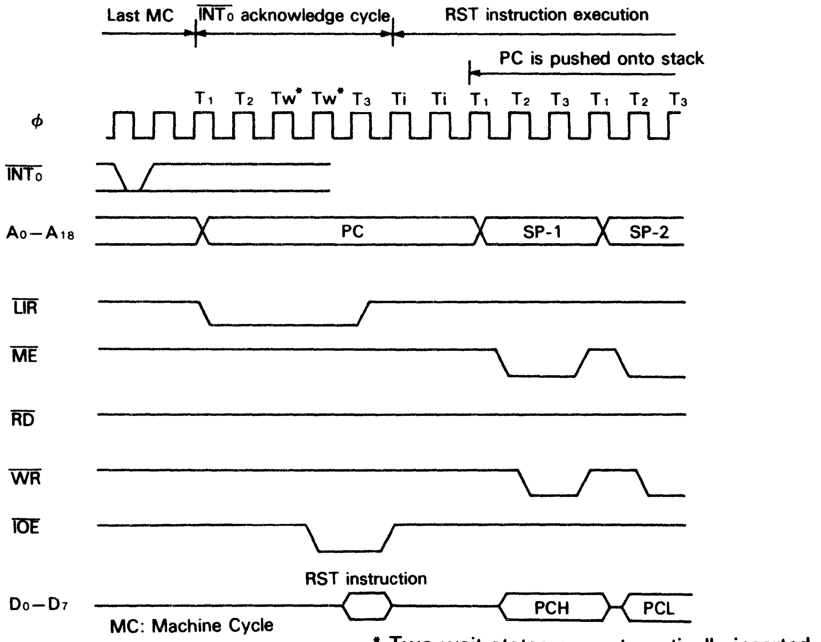


Figure 24 $\overline{INT_0}$ Mode 0 Timing (RST Instruction on the Data Bus)

$\overline{INT_0}$ Mode 1

When $\overline{INT_0}$ is received, the PC is stacked and instruction execution restarts at logical address 0038H. Both IEF_1 and IEF_2 flags are reset to 0, disabling all maskable interrupts. The interrupt service routine should normally terminate with the EI (Enable Interrupts) instruction followed by the RETI (Return from Interrupt) instruction, so that the interrupts are reenabled. Fig. 25 shows the use of $\overline{INT_0}$ (Mode 1) and RETI.

Fig. 26 shows $\overline{INT_0}$ Mode 1 timing.

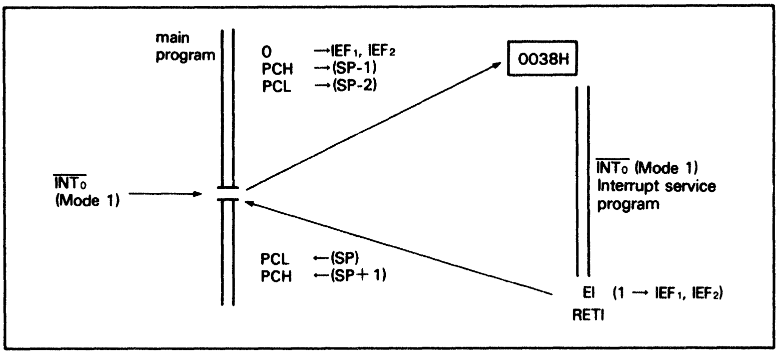
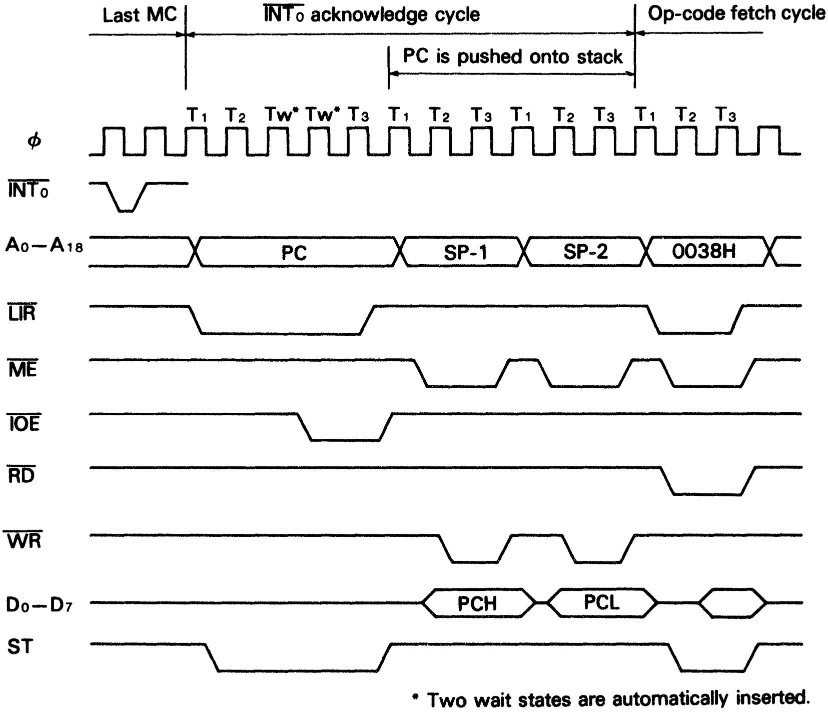


Figure 25 $\overline{INT_0}$ Mode 1 Interrupt Sequence



* Two wait states are automatically inserted.

Figure 26 $\overline{INT_0}$ Mode 1 Timing

$\overline{\text{INT}}_0$ Mode 2

This method determines the restart address by reading the contents of a table residing in memory. The vector table consists of up to 128 two-byte restart addresses stored in low byte, high byte order.

The vector table address is located on 256 bytes boundaries in the 64k bytes logical address space as programmed in the 8-bit Interrupt Vector Register (I). Fig. 27 shows the $\overline{\text{INT}}_0$ Mode 2 Vector acquisition.

During $\overline{\text{INT}}_0$ Mode 2 acknowledge cycle, first, the low-order 8 bits of vector is fetched from the data bus at the rising edge of T_3 and CPU acquires the 16-bit vector.

Next, the PC is stacked. Finally, the 16-bit restart address is fetched from the vector table and execution commences at that address.

Note that external vector acquisition is indicated by $\overline{\text{LIR}}$ and $\overline{\text{IOE}}$ both LOW. Two wait states (Tw) are automatically inserted for external vector fetch cycles

During RESET the Interrupt Vector Register (I) is initialized to 00H and, if necessary, should be set to a different value prior to the occurrence of a $\overline{\text{INT}}_0$ Mode 2 interrupt. Fig. 28 shows $\overline{\text{INT}}_0$ Mode 2 interrupt Timing.

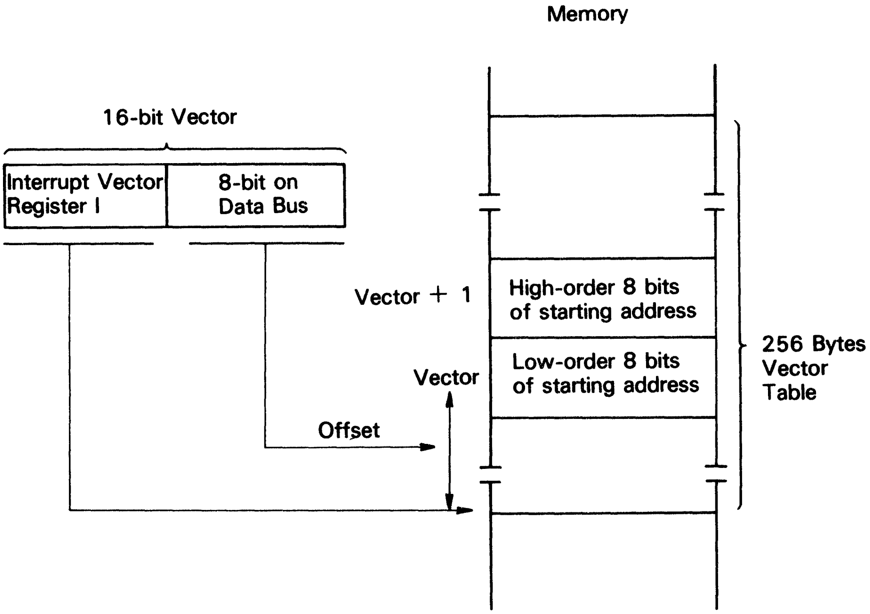
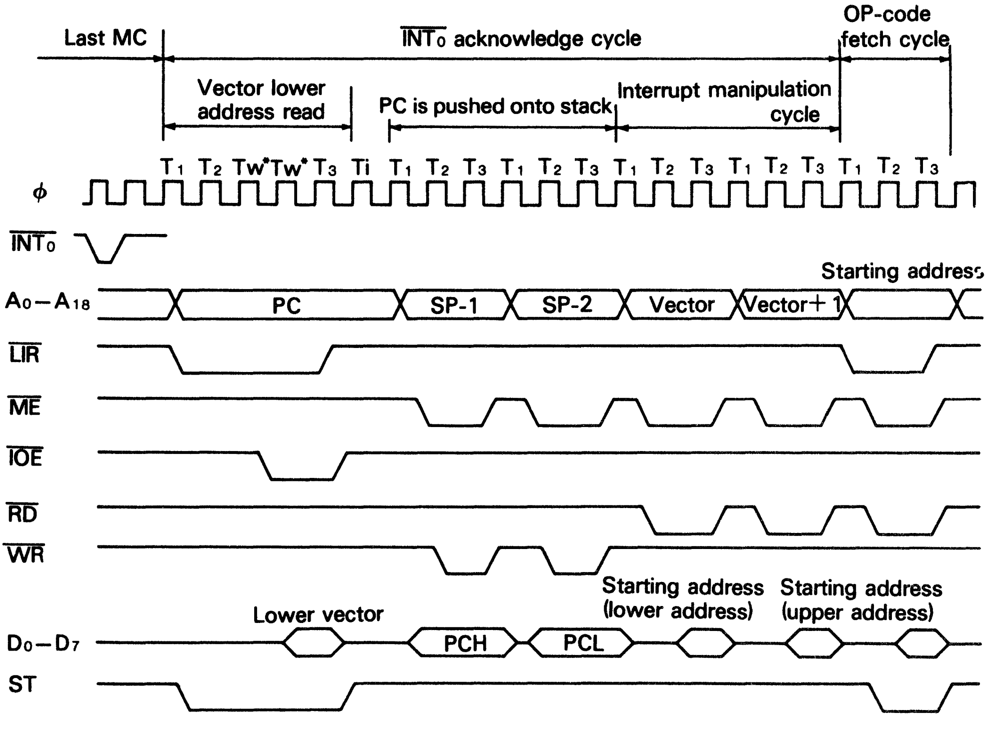

Memory
16-bit Vector
Interrupt Vector Register I
8-bit on Data Bus
Vector + 1
High-order 8 bits of starting address
Vector
Low-order 8 bits of starting address
Offset
256 Bytes Vector Table

Figure 27 $\overline{\text{INT}}_0$ Mode 2 Vector Acquisition


Last MC
$\overline{INT_0}$ acknowledge cycle
OP-code fetch cycle
Vector lower address read
PC is pushed onto stack
Interrupt manipulation cycle
T1 T2 Tw* Tw* T3 Ti T1 T2 T3 T1 T2 T3 T1 T2 T3 T1 T2 T3 T1 T2 T3
φ
$\overline{INT_0}$
Starting address
A0–A18
PC
SP-1
SP-2
Vector
Vector+1
$\overline{LIR}$
$\overline{ME}$
$\overline{IOE}$
$\overline{RD}$
$\overline{WR}$
Starting address (lower address)
Starting address (upper address)
Lower vector
D0–D7
PCH
PCL
ST

* Two wait states are automatically inserted.

Figure 28 $\overline{INT_0}$ Mode 2 Timing

6.6 $\overline{INT_1}$, $\overline{INT_2}$

The operation of external interrupts $\overline{INT_1}$ and $\overline{INT_2}$ is a vector mode similar to $\overline{INT_0}$ Mode 2. The difference is that $\overline{INT_1}$ and $\overline{INT_2}$ generate the low-order byte of vector table address using the IL (Interrupt Vector Low) register rather than fetching it from the data bus. This is also the interrupt response sequence used for all internal interrupts (except TRAP).

As shown in Fig. 29 the low-order byte of vector table address is comprised of the most significant three bits of the software programmable IL register and the least significant five bits which are a unique fixed value for each interrupt ($\overline{INT_1}$, $\overline{INT_2}$ and internal) source.

$\overline{INT_1}$ and $\overline{INT_2}$ are globally masked by $IEF_1 = 0$. Each is also individually maskable by respectively clearing the ITE1 and ITE2 (bits 1, 2) of the INT/TRAP control register to 0.

During RESET, IEF_1, ITE1 and ITE2 bits are initialized to 0.

6.7 Internal Interrupts

Internal interrupts (except TRAP) use the same vectored response mode as $\overline{INT_1}$ and $\overline{INT_2}$ (Fig 29). Internal interrupts are globally masked by $IEF_1 = 0$. Individual internal interrupts are enabled/disabled by programming each individual I/O (PRT, DMAC, CSI/O, ASCI) control register. The lower vector of $\overline{INT_1}$, $\overline{INT_2}$, and internal interrupt are summarized in Table 3.

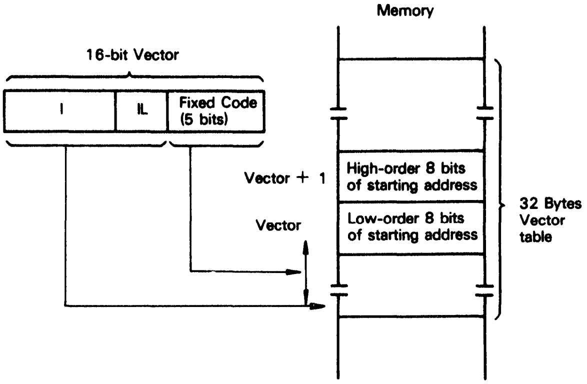


Figure 29 $\overline{\text{INT}}_1$, $\overline{\text{INT}}_2$ and Internal Interrupts Vector Acquisition

Table 3 Interrupt Source and Lower Vector

Interrupt Source	Priority	IL			Fixed Code				
		b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
$\overline{\text{INT}}_1$	Highest	•	•	•	0	0	0	0	0
$\overline{\text{INT}}_2$		•	•	•	0	0	0	1	0
PRT channel 0		•	•	•	0	0	1	0	0
PRT channel 1		•	•	•	0	0	1	1	0
DMA channel 0		•	•	•	0	1	0	0	0
DMA channel 1		•	•	•	0	1	0	1	0
CSI/O		•	•	•	0	1	1	0	0
ASCI channel 0		•	•	•	0	1	1	1	0
ASCI channel 1	Lowest	•	•	•	1	0	0	0	0

• Programmable

Interrupt Acknowledge Cycle Timing

Fig. 30 shows interrupt acknowledge cycle timing for internal interrupts, $\overline{INT}_1$, and $\overline{INT}_2$. $\overline{INT}_1$ and $\overline{INT}_2$ are sampled at the falling edge of clock state prior to T_3 or Ti in the last machine cycle. If $\overline{INT}_1$ or $\overline{INT}_2$ is asserted LOW at the falling edge of clock state prior to T_3 or Ti in the last machine cycle, the interrupt request is accepted.

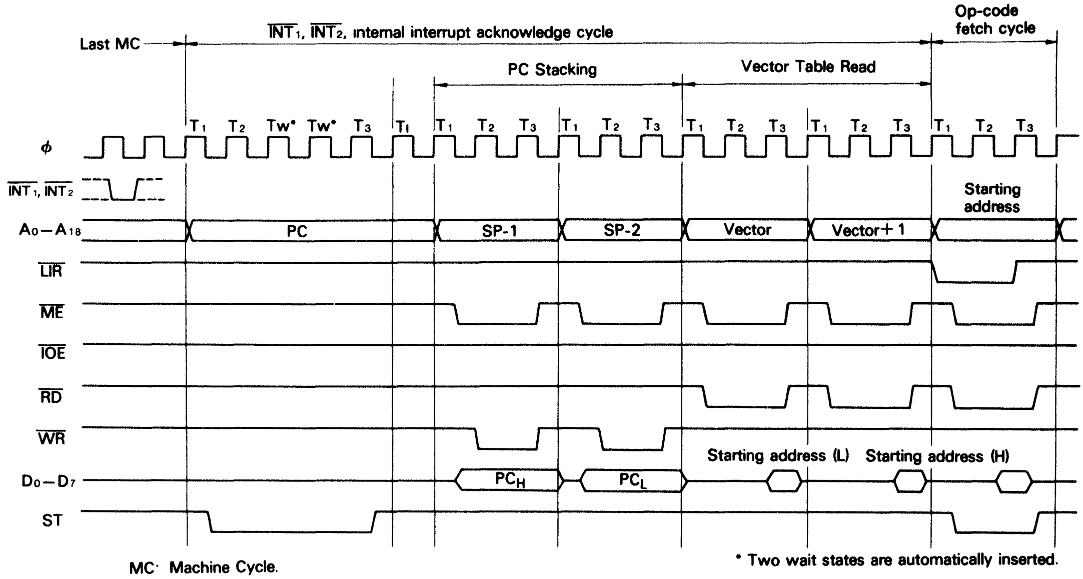


Figure 30 $\overline{INT}_1$, $\overline{INT}_2$ and Internal Interrupts Timing

6.8 Interrupt Sources and Reset

(1) Interrupt Vector Register (I)

All bits are reset to 0.

Since I = 0 locates the vector tables starting at logical address 0000H, vectored interrupts ($\overline{INT}_0$ Mode 2, $\overline{INT}_1$, $\overline{INT}_2$ and internal interrupts) will overlap with fixed restart interrupts like RESET (0), $\overline{NMI}$ (0066H), $\overline{INT}_0$ Mode 1 (0038H) and RST (0000H - 0038H). The vector table(s) can be built elsewhere in memory and located on 256 bytes boundaries by reprogramming I with the LD I, A instruction.

(2) IL Register

Bits 7 – 5 are reset to 0.

The IL Register can be programmed to locate the vector table for $\overline{INT}_1$, $\overline{INT}_2$ and internal interrupts on 32 bytes sub-boundaries within the 256 bytes area specified by I.

(3) IEF$_1$, IEF$_2$ Flags

Reset to 0.

Interrupts other than $\overline{NMI}$ and TRAP are disabled.

(4) ITC Register

ITE0 are set to 1. ITE1 and ITE2 are reset to 0.

$\overline{INT}_0$ can be enabled by the EI instruction, which sets IEF$_1$ = 1. To enable $\overline{INT}_1$ and $\overline{INT}_2$ also requires that the ITE1 and ITE2 bits be respectively set = 1 by writing to ITC.

(5) I/O Control Registers

Interrupt enable bits reset to 0.

All HD64180 on-chip I/O (PRT, DMAC, CSI/O, ASCI) interrupts are disabled and can be individually enabled by writing to each I/O control register interrupt enable bit.

6.9 Difference between $\overline{INT}_0$ interrupt and the other interrupts ($\overline{INT}_1$, $\overline{INT}_2$ and internal interrupts) in the interrupt acknowledge cycles

As shown in Fig. 24, Fig. 26, Fig. 28 and Fig. 30, the interrupt acknowledge cycle of $\overline{INT}_0$ is different from those of the other interrupts, that is, $\overline{INT}_1$, $\overline{INT}_2$ and internal interrupts concerning the state of control signals. The state of the control signals in each interrupt acknowledge cycle are shown below.

$\overline{INT}_0$ interrupt acknowledge cycle: $\overline{LIR}$ = 0, $\overline{IOE}$ = 0, ST = 0

$\overline{INT}_1$, $\overline{INT}_2$, and internal interrupt acknowledge cycle: $\overline{LIR}$ = 1, $\overline{IOE}$ = 1, ST = 0

7 MEMORY MANAGEMENT UNIT (MMU)

The HD64180 contains an on-chip MMU which performs the translation of the CPU 64k bytes (16-bit addresses- 0000H to FFFFH) logical memory address space into a 512k bytes (19-bit addresses- 00000H to 7FFFFH) physical memory address space. Address translation occurs internally in parallel with other CPU operation.

7.1 Logical Address Spaces

The 64k bytes CPU logical address space is interpreted by the MMU as consisting of up to three separate logical address areas, Common Area 0, Bank Area and Common Area 1.

As shown in Fig. 31 a variety of logical memory configurations are possible. The boundaries between the Common and Bank Areas can be programmed with 4k bytes resolution.

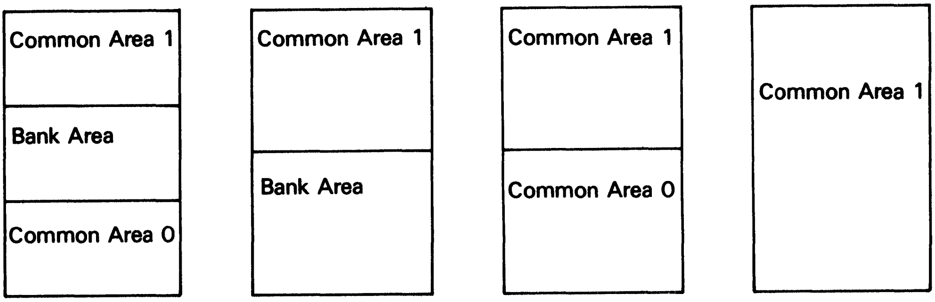


Figure 31 Logical Address Mapping Examples

7.2 Logical to Physical Address Translation

Fig. 32 shows an example in which the three logical address space portions are mapped into a 512k bytes physical address space. The important points to note are that Common and Bank Areas can overlap and that Common Area 1 and Bank Area can be freely relocated (on 4k bytes physical address boundaries). Common Area 0 (if it exists) is always based at physical address 00000H.

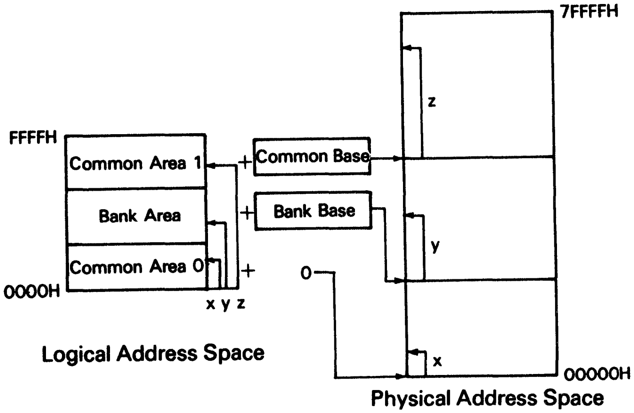


Figure 32 Logical → Physical Memory Mapping Example

7.3 MMU Block Diagram

The MMU block diagram is shown in Fig. 33. The MMU translates internal 16-bit logical addresses to external 19-bit physical addresses.

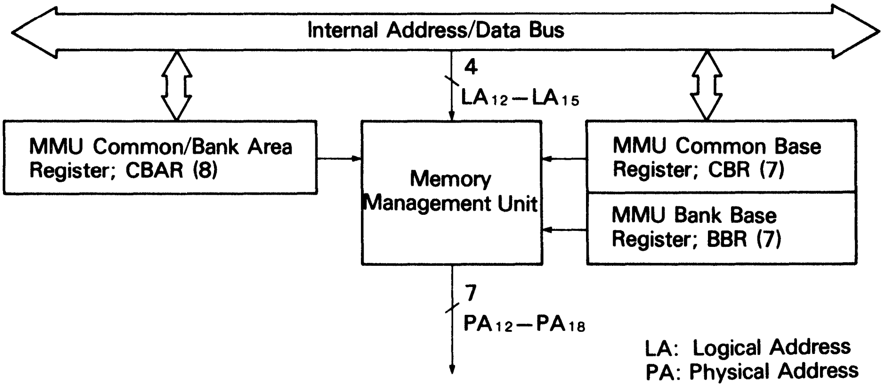

Internal Address/Data Bus
4
LA12—LA15
MMU Common/Bank Area Register; CBAR (8)
Memory Management Unit
MMU Common Base Register; CBR (7)
MMU Bank Base Register; BBR (7)
7
PA12—PA18
LA: Logical Address
PA: Physical Address

Figure 33 MMU Block Diagram

Whether address translation takes place depends on the type of CPU cycle as follows.

(1) Memory Cycles

Address Translation occurs for all memory access cycles including instruction and operand fetches, memory data reads and writes, hardware interrupt vector fetch and software interrupt restarts.

(2) I/O Cycles

The MMU is logically bypassed for I/O cycles. The 16-bit logical I/O address space corresponds directly with the 16-bit physical I/O address space. The upper three bits (A_{16}-A_{18}) of the physical address are always 0 during I/O cycles.

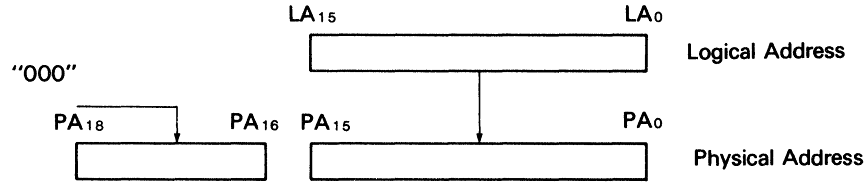


Figure 34 I/O Address Translation

(3) DMA Cycles

When the HD64180 on-chip DMAC is using the external bus, the MMU is physically bypassed. The 19-bit source and destination registers in the DMAC are directly output on the physical address bus (A_0-A_{18}).

7.4 MMU Registers

Three MMU registers are used to program a specific configuration of logical and physical memory.

(1) MMU Common/Bank Area Register (CBAR)
(2) MMU Common Base Register (CBR)
(3) MMU Bank Base Register (BBR)

CBAR is used to define the logical memory organization, while CBR and BBR are used to relocate logical areas within the 512k bytes physical address space. The resolution for both setting boundaries within the logical space and relocation within the physical space is 4k bytes.

The CAR field of CBAR determines the start address of Common Area 1 (Upper Common) and by default, the end address of the Bank Area. The BAR field determines the start address of the Bank Area and by default, the end address of Common Area 0 (Lower Common).

The CA and BA fields of CBAR may be freely programmed subject only to the restriction that CA may never be less than BA. Fig. 35 and Fig. 36 shows example of logical memory organizations associated with different values of CA and BA.

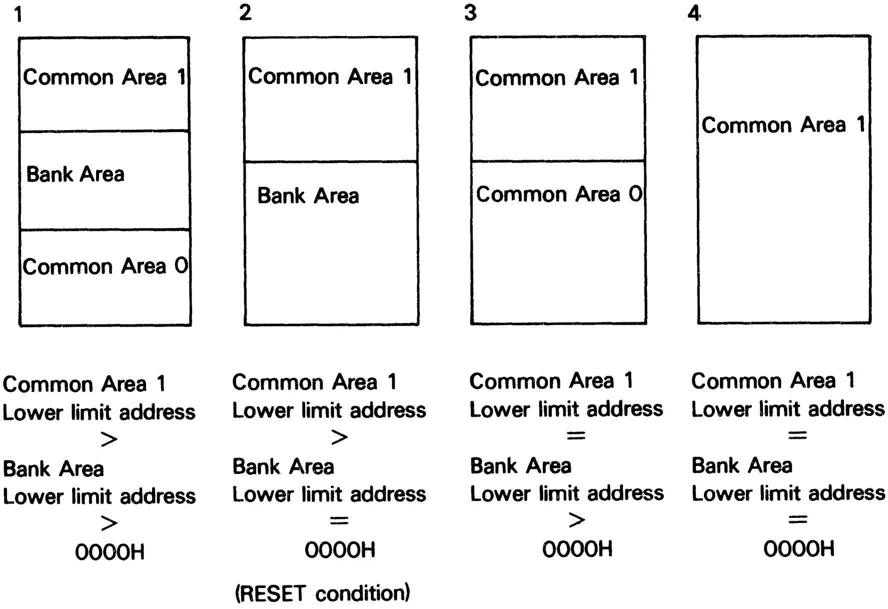


Figure 35 Logical Memory Organization

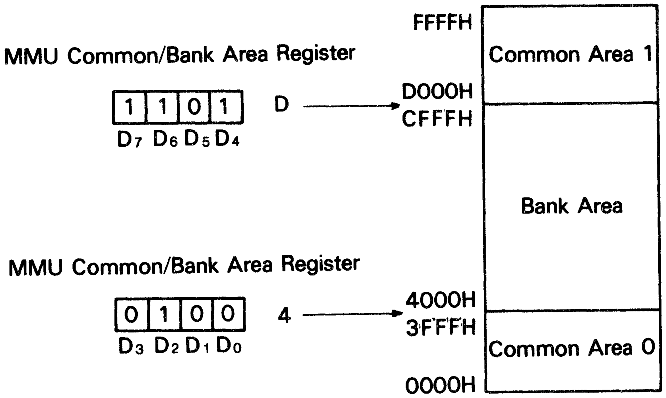


Figure 36 Logical Space Configuration (Example)

7.5 MMU Register Description

(1) MMU Common/Bank Area Register (CBAR)

CBAR specifies boundaries within the HD64180 64k bytes logical address space for up to three areas, Common Area 0, Bank Area, and Common Area 1.

MMU Common/Bank Area Register (CBAR . I/O Address = 3AH)

bit	7	6	5	4	3	2	1	0
	CA3	CA2	CA1	CA0	BA3	BA2	BA1	BA0
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

CA3-CA0: CA (bits 7-4)

CA specifies the start (low) address (on 4k bytes boundaries) for the Common Area 1. This also determines the last address of the Bank Area. All bits of CA are initialized to 1 during RESET

BA3-BA0: BA (bits 3-0)

BA specifies the start (low) address (on 4k bytes boundaries) for the Bank Area. This also determines the last address of the Common Area 0. All bits of BA are initialized to 0 during RESET.

(2) MMU Common Base Register (CBR)

CBR specifies the base address (on 4k bytes boundaries) used to generate a 19-bit physical address for Common Area 1 accesses. All bits of CBR are initialized to 0 during RESET.

MMU Common Base Register (CBR . I/O Address = 38H)

bit	7	6	5	4	3	2	1	0
	–	CB6	CB5	CB4	CB3	CB2	CB1	CB0
		R/W	R/W	R/W	R/W	R/W	R/W	R/W

(3) MMU Bank Base Register (BBR)

BBR specifies the base address (on 4k bytes boundaries) used to generate a 19-bit physical address for Bank Area accesses. All bits of BBR are initialized to 0 during RESET.

MMU Bank Base Register (BBR . I/O Address = 39H)

bit	7	6	5	4	3	2	1	0
	–	BB6	BB5	BB4	BB3	BB2	BB1	BB0
		R/W	R/W	R/W	R/W	R/W	R/W	R/W

7.6 Physical Address Translation

Fig. 37 shows the way in which physical addresses are generated based on the contents of CBAR, CBR and BBR. MMU comparators classify an access by logical area as defined by CBAR. Depending on which of the three potential logical areas (Common Area 1, Bank Area or Common Area 0) is being accessed, the appropriate 7-bit base address is added to the upper 4 bits of the logical address, yielding a 19-bit physical address. CBR is associated with Common Area 1 accesses. Common Area 0 accesses use a (non-accessible, internal) base register which contains 0. Thus, Common Area 0, if defined, is always based at physical address 00000H.

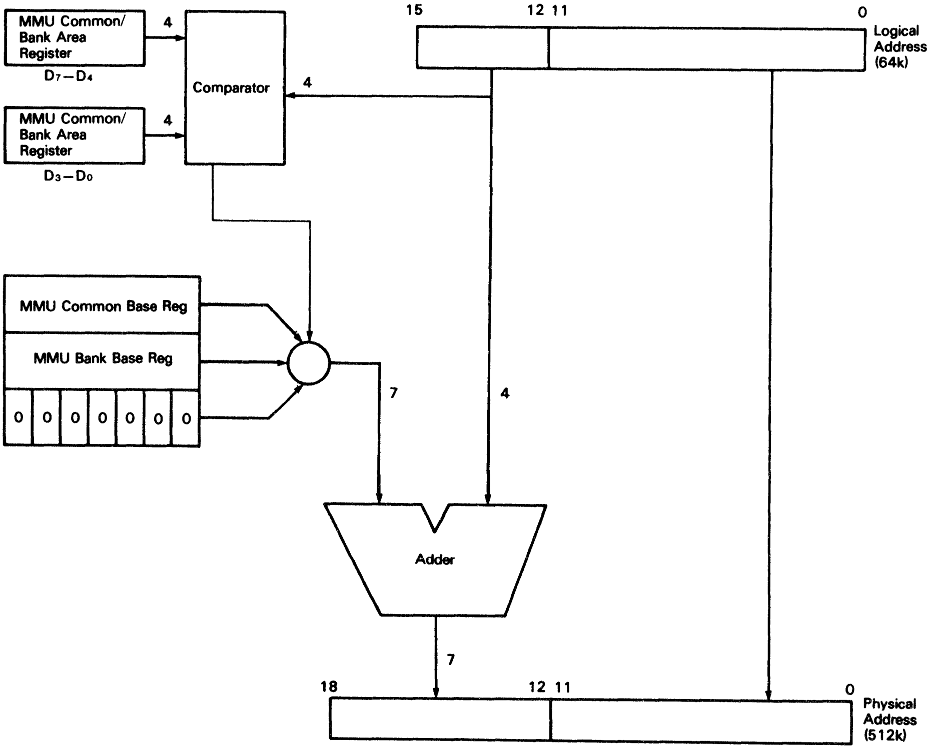


Figure 37 Physical Address Generation

7.7 MMU and RESET

During RESET, all bits of the CA field of CBAR are set to 1 while all bits of the BA field of CBAR, CBR, and BBR are cleared to 0. The logical 64k bytes address space corresponds directly with the first 64k bytes (0000H to FFFFH) of the 512k bytes (00000H to 7FFFFH) physical address space. Thus, after RESET, the HD64180 will begin execution at logical and physical address 0.

7.8 MMU Register Access Timing

When data is written into CBAR, CBR, or BBR, the value will be effective from the cycle immediately following the I/O write cycle which updates these registers.

Care must be taken during MMU programming to insure that CPU program execution is not disrupted. Observe that the next cycle following MMU register programming will normally be an opcode fetch from the newly translated address. One simple technique is to localize all MMU programming routines in a Common Area that is always enabled.

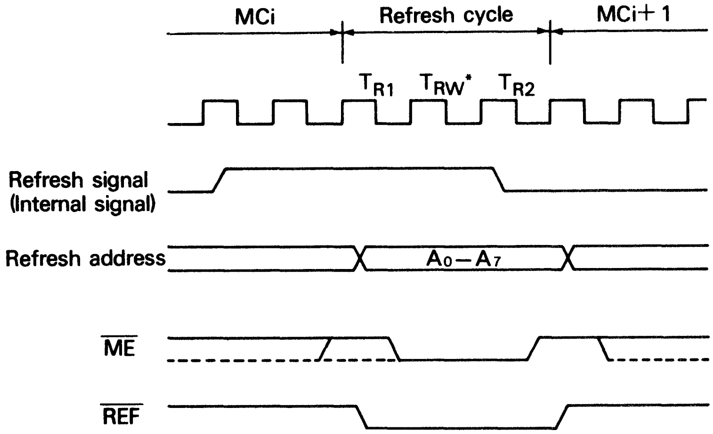
8 DYNAMIC RAM REFRESH CONTROL

The HD64180 incorporates a dynamic RAM refresh control circuit including 8-bit refresh address generation and programmable refresh timing. This circuit generates asynchronous refresh cycles inserted at the programmable interval independent of CPU program execution. For systems which don't use dynamic RAM, the refresh function can be disabled.

When the internal refresh controller determines that a refresh cycle should occur, the current instruction is interrupted at the first breakpoint between machine cycles. The refresh cycle is inserted by placing the refresh address on A_0-A_7 and the $\overline{\text{REF}}$ output is driven LOW.

Refresh cycles may be programmed to be either two or three clock cycles in duration by programming the REFW (Refresh Wait) bit in Refresh Control Register (RCR). Note that the external $\overline{\text{WAIT}}$ input and the internal wait state generator are not effective during refresh

Fig. 38 shows the timing of a refresh cycle with a refresh wait (T_{RW}) cycle.



NOTE: * If three refresh cycles are specified, T_{RW}, is inserted. Otherwise, T_{RW} is not inserted.
MC: Machine Cycle

Figure 38 Refresh Timing

8.1 Refresh Control Register (RCR)

RCR specifies the interval and length of refresh cycles, as well as enabling or disabling the refresh function.

Refresh Control Register (RCR. I/O Address = 36H)

bit	7	6	5	4	3	2	1	0
	REFE	REFW	–	–	–	–	CYC1	CYC0
	R/W	R/W					R/W	R/W

REFE: Refresh Enable (bit 7)

REFE = 0 disables the refresh controller while REFE = 1 enables refresh cycle insertion. REFE is set to 1 during RESET.

REFW: Refresh Wait (bit 6)

REFW = 0 causes the refresh cycle to be two clocks in duration. REFW = 1 causes the refresh cycle to be three clocks in duration by adding a refresh wait cycle (T_{RW}). REFW is set to 1 during RESET.

CYC1, 0: Cycle Interval (bits 1-0)

CYC1 and CYC0 specify the interval (in clock cycles) between refresh cycles.

In the case of dynamic RAMs requiring 128 refresh cycles every 2 ms (or 256 cycles every 4 ms), the required refresh interval is less than or equal to 15.625 μs. Thus, the underlined values indicate the best refresh interval depending on CPU clock frequency. CYC0 and CYC1 are cleared to 0 during RESET.

Table 4 Refresh Interval

CYC1	CYC0	Insertion interval	Time interval				
			ϕ: 10 MHz	8 MHz	6 MHz	4 MHz	2.5 MHz
0	0	10 states	(1.0 μs)*	(1.25 μs)*	1.66 μs	2.5 μs	4.0 μs
0	1	20 states	(2.0 μs)*	(2 5 μs)*	3.3 μs	5.0 μs	8.0 μs
1	0	40 states	(4.0 μs)*	(5.0 μs)*	6.6 μs	10.0 μs	16.0 μs
1	1	80 states	(8.0 μs)*	(10 0 μs)*	13 3 μs	20.0 μs	32.0 μs

* calculated interval

8.2 Refresh control and reset

After RESET, based on the initialized value of RCR, refresh cycles will occur with an interval of 10 clock cycles and be 3 clock cycles in duration.

8.3 Dynamic RAM refresh operation notes

(1) Refresh cycle insertion is stopped when the CPU is in the following states.

(a) During RESET

(b) When the bus is released in response to $\overline{\text{BUSREQ}}$

(c) During SLEEP mode

(d) During WAIT states

(2) Refresh cycles are suppressed when the bus is released in response to $\overline{\text{BUSREQ}}$. However, the refresh timer continues to operate. Thus, the time at which the first refresh cycle occurs after the HD64180 re-acquires the bus depends on the refresh timer, and has no timing relationship with the bus exchange.

(3) Refresh cycles are suppressed during SLEEP mode. If a refresh cycle is requested during SLEEP mode, the refresh cycle request is internally 'latched' (until replaced with the next refresh request). The 'latched' refresh cycle is inserted at the end of the first machine cycle after SLEEP mode is exited. After this initial cycle, the time at which the next refresh cycle will occur depending on the refresh time, and has no timing relationship with the exit from SLEEP mode.

(4) Regarding (2) and (3), the refresh address is incremented by 1 for each successful refresh cycle, not for each refresh request. Thus, independent of the number of 'missed' refresh requests, each refresh bus cycle will use a refresh address incremented by 1 from that of the previous refresh bus cycles.

9 WAIT STATE GENERATOR

9.1 Wait State Timing

To ease interfacing with slow memory and I/O devices, the HD64180 uses wait states (Tw) to extend bus cycle timing. A wait state(s) is inserted based on the combined (logical OR) state of the external $\overline{\text{WAIT}}$ input and an internal programmable wait state (Tw) generator. Wait states (Tw) can be inserted in both CPU execution and DMA transfer cycles.

9.2 $\overline{\text{WAIT}}$ Input

When the external $\overline{\text{WAIT}}$ input is asserted LOW, wait state (Tw) are inserted between T_2 and T_3 to extend the bus cycle duration. The $\overline{\text{WAIT}}$ input is sampled at the falling edge of the system clock in T_2 or Tw. If the $\overline{\text{WAIT}}$ input is asserted LOW at the falling edge of the system clock in Tw, another Tw is inserted into the bus cycle. Note that $\overline{\text{WAIT}}$ input transitions must meet specified set-up and hold times. This can easily be accomplished by externally synchronizing $\overline{\text{WAIT}}$ input transitions with the rising edge of the system clock.

Dynamic RAM refresh is not performed during wait states (Tw) and thus systems designs which uses the automatic refresh function must consider the affects of the occurrence and duration of wait states (Tw).

Fig. 39 shows $\overline{\text{WAIT}}$ timing.

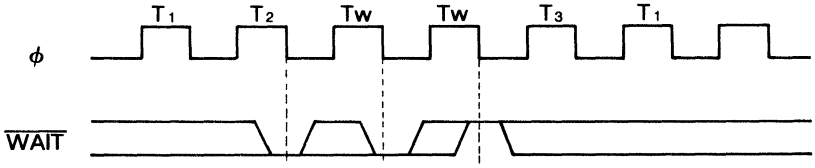


Figure 39 $\overline{\text{WAIT}}$ Timing

9.3 Programmable WAIT State Insertion

In addition to the $\overline{\text{WAIT}}$ input, wait states (Tw) can also be programmably inserted using the HD64180 on-chip wait state generator. Wait state (Tw) timing applies for both CPU execution and on-chip DMAC cycles.

By programming the 4 significant bits of the DMA/WAIT Control Register (DCNTL), the number of wait states (Tw) automatically inserted in memory and I/O cycles can be separately specified. Bits 4-5 specify the number of wait states (Tw) inserted for I/O access and bits 6-7 specify the number of wait states (Tw) inserted for memory access.

DMA/WAIT Control Register
(DCNTL I/O Address = 32H)

bit	7	6	5	4
	MWI1	MWI0	IWI1	IWI0
	R/W	R/W	R/W	R/W

The number of wait states (Tw) inserted in a specific cycle is the maximum of the number requested by the $\overline{\text{WAIT}}$ input, and the number automatically generated by the on-chip wait state generator.

MWI1, MWI0: Memory Wait Insertion (bits 7-6)

For CPU and DMAC cycles which access memory (including memory mapped I/O), 0 to 3 wait states may be automatically inserted depending on the programmed value in MWI1 and MWI0.

MWI1	MWI0	The number of wait states
0	0	0
0	1	1
1	0	2
1	1	3

IWI1, IWI0: I/O Wait Insertion (bits 5-4)

For CPU and DMA cycles which access external I/O (and interrupt acknowledge cycles), 1 to 6 wait states (Tw) may be automatically inserted depending on the programmed value in IWI1 and IWI0.

		The number of wait states				
IWI1	IWI0	For external I/O registers accesses	For internal I/O registers accesses	For $\overline{\text{INT}}_0$ interrupt acknowledge cycles when $\overline{\text{LIR}}$ is LOW	For $\overline{\text{INT}}_1$, $\overline{\text{INT}}_2$ and internal interrupts acknowledge cycles (Note (2))	For $\overline{\text{NMI}}$ interrupt acknowledge cycles when $\overline{\text{LIR}}$ is LOW (Note (2))
0	0	1	0 (Note (1))	2	2	0
0	1	2		4		
1	0	3		5		
1	1	4		6		

NOTE. (1) For HD64180 internal I/O register access (I/O addresses 0000H-003FH), IWI1 and IWI0 do not determine wait state (Tw) timing For ASCI, CSI/O and PRT Data Register accesses, 0 to 4 wait states (Tw) will be generated Wait states inserted during access to these registers is a function of internal synchronization requirements and CPU state
All other on-chip I/O register accesses (i e MMU, DMAC, ASCI Control Registers, etc) have 0 wait states inserted and thus require only three clock cycles

(2) For interrupt acknowledge cycles in which $\overline{\text{LIR}}$ is HIGH, such as interrupt vector table read and PC stacking cycle, memory access timing applies.

9.4 WAIT Input and RESET

During RESET, MWI1, MWI0, IWI1 and IWI0 are all set to 1, selecting the maximum number of wait states (Tw) (3 for memory accesses, 4 for external I/O accesses).

Also, note that the $\overline{\text{WAIT}}$ input is ignored during RESET. For example, if RESET is detected while the HD64180 is in a wait state (Tw), the wait stated cycle in progress will be aborted, and the RESET sequence initiated. Thus, RESET has higher priority than $\overline{\text{WAIT}}$.

10 DMA CONTROLLER (DMAC)

The HD64180 contains a two channel DMA (Direct Memory Access) controller which supports high speed data transfer. Both channels (channel 0 and channel 1) have the following capabilities.

Memory Address Space

Memory source and destination addresses can be directly specified anywhere within the 512k bytes physical address space using 19-bit source and destination memory addresses. In addition, memory transfers can arbitrarily cross 64k bytes physical address boundaries without CPU intervention.

I/O Address Space

I/O source and destination addresses can be directly specified anywhere within the 64k bytes I/O address space (16-bit source and destination I/O addresses).

Transfer Length

Up to 64k bytes can be transferred based on a 16-bit byte count register.

$\overline{\text{DREQ}}$ Input

Level and edge sense $\overline{\text{DREQ}}$ input detection are selectable.

$\overline{\text{TEND}}$ Output

Used to indicate DMA completion to external devices.

Transfer Rate

Each byte transfer can occur every six clock cycles. Wait states can be inserted in DMA cycles for slow memory or I/O devices. At the system clock (ϕ) = 6 MHz, the DMA transfer rate is as high as 1.0 megabytes/second (no wait states).

Additional feature disk for DMA interrupt request by DMA END.

Each channel has the following additional specific capabilities.

Channel 0

- Memory <—> memory, memory <—> I/O, memory <—> memory mapped I/O transfers
- Memory address increment, decrement, no-change
- Burst or cycle steal memory <—> memory transfers
- DMA to and from both ASCI channels
- Higher priority than DMAC channel 1

Channel 1

- Memory <—> I/O transfer
- Memory address increment, decrement

DMAC Registers

Each channel of the DMAC (channel 0, 1) has three registers specifically associated with that channel.

Channel 0

SAR0	–	Source Address Register
DAR0	–	Destination Address Register
BCR0	–	Byte Count Register

Channel 1

MAR1	–	Memory Address Register
IAR1	–	I/O Address Register
BCR1	–	Byte Count Register

The two channels share the following three additional registers in common.

DSTAT	–	DMA Status Register
DMODE	–	DMA Mode Register
DCNTL	–	DMA Control Register

10.1 DMAC Block Diagram

Fig. 40 shows the HD64180 DMAC Block Diagram.

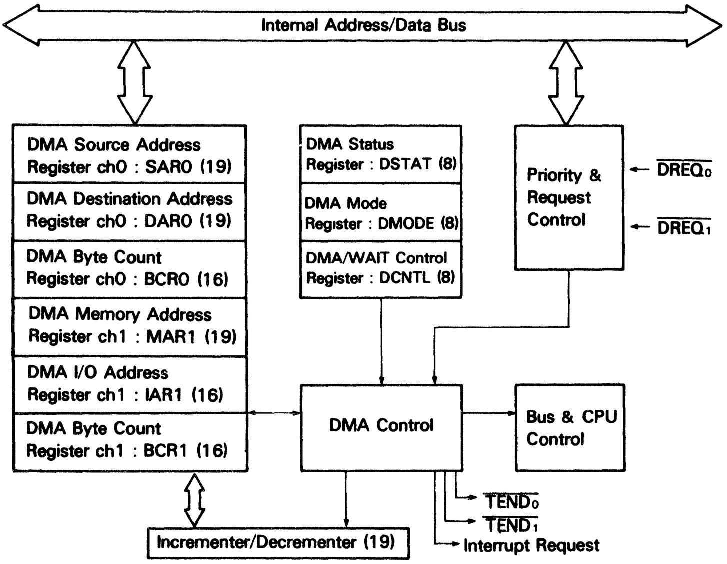

Internal Address/Data Bus
DMA Source Address Register ch0 : SAR0 (19)
DMA Destination Address Register ch0 : DAR0 (19)
DMA Byte Count Register ch0 : BCR0 (16)
DMA Memory Address Register ch1 : MAR1 (19)
DMA I/O Address Register ch1 : IAR1 (16)
DMA Byte Count Register ch1 : BCR1 (16)
DMA Status Register : DSTAT (8)
DMA Mode Register : DMODE (8)
DMA/WAIT Control Register : DCNTL (8)
Priority & Request Control
$\overline{\text{DREQ}_0}$
$\overline{\text{DREQ}_1}$
DMA Control
Bus & CPU Control
$\overline{\text{TEND}_0}$
$\overline{\text{TEND}_1}$
Interrupt Request
Incrementer/Decrementer (19)

Figure 40 DMAC Block Diagram

10.2 DMAC Register Description

(1) DMA Source Address Register Channel 0 (SAR0: I/O Address = 20H to 22H)

Specifies the physical source address for channel 0 transfers. The register contains 19 bits and may specify up to 512k bytes memory addresses or up to 64k bytes I/O addresses. Channel 0 source can be memory, I/O or memory mapped I/O

(2) DMA Destination Address Register Channel 0 (DAR0: I/O Address = 23H to 25H)

Specifies the physical destination address for channel 0 transfers. The register contains 19 bits and may specify up to 512k bytes memory addresses or up to 64k bytes I/O addresses. Channel 0 destination can be memory, I/O or memory mapped I/O.

(3) DMA Byte Count Register Channel 0 (BCR0: I/O Address = 26H to 27H)

Specifies the number of bytes to be transferred. This register contains 16 bits and may specify up to 64k bytes transfers. When one byte is transferred, the register is decremented by one. If "n" bytes should be transferred, "n" must be stored before the DMA operation.

(4) DMA Memory Address Register Channel 1 (MAR1: I/O Address = 28H to 2AH)

Specifies the physical memory address for channel 1 transfers. This may be destination or source memory address.
This register contains 19 bits and may specify up to 512k bytes memory addresses.

(5) DMA I/O Address Register Channel 1 (IAR1: I/O Address = 2BH to 2CH)

Specifies the I/O address for channel 1 transfers. This may be destination or source I/O address. This register contains 16 bits and may specify up to 64k bytes I/O addresses.

(6) DMA Byte Count Register Channel 1 (BCR1: I/O Address = 2EH to 2FH)

Specifies the number of bytes to be transferred. This register contains 16 bits and may specify up to 64k bytes transfers. When one byte is transferred, the register is decremented by one.

(7) DMA Status Register (DSTAT)

DSTAT is used to enable and disable DMA transfer and DMA termination interrupts. DSTAT also allows determining the status of a DMA transfer i.e. completed or in progress.

DMA Status Register (DSTAT I/O Address = 30H)

bit	7	6	5	4	3	2	1	0
	DE1	DE0	$\overline{DWE1}$	$\overline{DWE0}$	DIE1	DIE0	–	DME
	R/W	R/W	W	W	R/W	R/W		R

DE1: DMA Enable Channel 1 (bit 7)

When DE1 = 1 and DME = 1, channel 1 DMA is enabled. When a DMA transfer terminates (BCR1 = 0), DE1 is reset to 0 by the DMAC. When DE1 = 0 and the DMA interrupt is enabled (DIE1 = 1), a DMA interrupt request is made to the CPU.

To perform a software write to DE1, $\overline{DWE1}$ should be written with 0 during the same register write access. Writing DE1 to 0 disables channel 1 DMA, but DMA is restartable. Writing DE1 to 1 enables channel 1 DMA and automatically sets DME (DMA Main Enable) to 1. DE1 is cleared to 0 during RESET.

DE0: DMA Enable Channel 0 (bit 6)

When DE0 = 1 and DME = 1, channel 0 DMA is enabled. When a DMA transfer terminates (BCR0 = 0), DE0 is cleared to 0 by the DMAC. When DE0 = 0 and the DMA interrupt is enabled (DIE0 = 1), a DMA interrupt request is made to the CPU.

To perform a software write to DE0, $\overline{DWE0}$ should be written with 0 during the same register write access. Writing DE0 to 0 disables channel 0 DMA. Writing DE0 to 1 enables channel 0 DMA and automatically sets DME (DMA Main Enable) to 1. DE0 is cleared to 0 during RESET.

$\overline{DWE1}$: DE1 Bit Write Enable (bit 5)

When performing any software write to DE1, $\overline{DWE1}$ should be written with 0 during the same access. $\overline{DWE1}$ write value of 0 is not held and $\overline{DWE1}$ is always read as 1.

$\overline{DWE0}$: DE0 Bit Write Enable (bit 4)

When performing any software write to DE0, $\overline{DWE0}$ should be written with 0 during the same access. $\overline{DWE0}$ write value of 0 is not held and $\overline{DWE0}$ is always read as 1.

DIE1: DMA Interrupt Enable Channel 1 (bit 3)

When DIE1 is set to 1, the termination of channel 1 DMA transfer (indicated when DE1 = 0) causes a CPU interrupt request to be generated. When DIE1 = 0, the channel 1 DMA termination interrupt is disabled. DIE1 is cleared to 0 during RESET.

DIE0: DMA Interrupt Enable Channel 0 (bit 2)

When DIE0 is set to 1, the termination channel 0 of DMA transfer (indicated when DE0 = 0) causes a CPU interrupt request to be generated. When DIE0 = 0, the channel 0 DMA termination interrupt is disabled. DIE0 is cleared to 0 during RESET.

DME: DMA Main Enable (bit 0)

A DMA operation is only enabled when its DE bit (DE0 for channel 0, DE1 for channel 1) and the DME bit are set to 1.

When $\overline{NMI}$ occurs, DME is reset to 0, thus disabling DMA activity during the NMI interrupt service routine. To restart DMA, DE0 and/or DE1 should be written with 1 (even if the contents are already 1). This automatically sets DME to 1, allowing DMA operations to continue. Note that DME cannot be directly written. It is cleared to 0 by $\overline{NMI}$ or indirectly set to 1 by setting DE0 and/or DE1 to 1. DME is cleared to 0 during RESET.

(8) DMA Mode Register (DMODE)

DMODE is used to set the addressing and transfer mode for channel 0.

DMA Mode Register (DMODE I/O Address = 31H)

bit	7	6	5	4	3	2	1	0
	–	–	DM1	DM0	SM1	SM0	MMOD	–
			R/W	R/W	R/W	R/W	R/W	

DM1, DM0: Destination Mode Channel 0 (bits 5, 4)

Specifies whether the destination for channel 0 transfers is memory, I/O or memory mapped I/O and the corresponding address modifier. DM1 and DM0 are cleared to 0 during RESET.

Table 5 Destination

DM1	DM0	Memory/I/O	Address Increment/Decrement
0	0	Memory	+1
0	1	Memory	−1
1	0	Memory	fixed
1	1	I/O	fixed

SM1, SM0: Source Mode Channel 0 (bits 3, 2)

Specifies whether the source for channel 0 transfers is memory, I/O or memory mapped I/O and the corresponding address modifier. SM1 and SM0 are cleared to 0 during RESET.

Table 7 shows all DMA transfer mode combinations of DM0, DM1, SM0, SM1. Since I/O ⟷ I/O transfers are not implemented, twelve combinations are available.

Table 6 Source

SM1	SM0	Memory/I/O	Address Increment/Decrement
0	0	Memory	+1
0	1	Memory	−1
1	0	Memory	fixed
1	1	I/O	fixed

Table 7 Combination of Transfer Mode

DM1	DM0	SM1	SM0	Transfer Mode	Address Increment/Decrement
0	0	0	0	Memory→Memory	SAR0+1, DAR0+1
0	0	0	1	Memory→Memory	SAR0−1, DAR0+1
0	0	1	0	Memory*→Memory	SAR0 fixed, DAR0+1
0	0	1	1	I/O→Memory	SAR0 fixed, DAR0+1
0	1	0	0	Memory→Memory	SAR0+1, DAR0−1
0	1	0	1	Memory→Memory	SAR0−1, DAR0−1
0	1	1	0	Memory*→Memory	SAR0 fixed, DAR0−1
0	1	1	1	I/O→Memory	SAR0 fixed, DAR0−1
1	0	0	0	Memory→Memory*	SAR0+1, DAR0 fixed
1	0	0	1	Memory→Memory*	SAR0−1, DAR0 fixed
1	0	1	0	reserved	
1	0	1	1	reserved	
1	1	0	0	Memory→I/O	SAR0+1, DAR0 fixed
1	1	0	1	Memory→I/O	SAR0−1, DAR0 fixed
1	1	1	0	reserved	
1	1	1	1	reserved	

* : includes memory mapped I/O

MMOD: Memory Mode Channel 0 (bit 1)

When channel 0 is configured for memory ⟷ memory transfers, the external $\overline{DREQ}_0$ input is not used to control the transfer timing. Instead, two automatic transfer timing modes are selectable – burst (MMOD = 1) and cycle steal (MMOD = 0). For burst memory ⟷ memory transfers, the DMAC will sieze control of the bus continuously until the DMA transfer completes (as shown by the byte count register = 0). In cycle steal mode, the CPU is given a cycle for each DMA byte transfer cycle until the transfer is completed.

For channel 0 DMA with I/O source or destination, the $\overline{DREQ}_0$ input times the transfer and thus MMOD is ignored. MMOD is cleared to 0 during RESET.

DMA/WAIT Control Register (DCNTL)

DCNTL controls the insertion of wait states into DMAC (and CPU) accesses of memory or I/O. Also, the DMA request mode for each $\overline{DREQ}$ ($\overline{DREQ}_0$ and $\overline{DREQ}_1$) input is defined as level or edge sense. DCNTL also sets the DMA transfer mode for channel 1, which is limited to memory ⟷ I/O transfers.

DMA/WAIT Control Register (DCNTL I/O Address = 32H)

bit	7	6	5	4	3	2	1	0
	MWI1	MWI0	IWI1	IWI0	DMS1	DMS0	DIM1	DIM0
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

MWI1, MWI0: Memory Wait Insertion (bits 7-6)

Specifies the number of wait states introduced into CPU or DMAC memory access cycles. MWI1 and MWI0 are set to 1 during RESET. See section of Wait State Control for details.

IWI1, IWI0: I/O Wait Insertion (bits 5-4)

Specifies the number of wait states introduced into CPU or DMAC I/O access cycles. IWI1 and IWI0 are set to 1 during RESET. See section of Wait State Control for details.

DMS1, DMS0: DMA Request Sense (bits 3-2)

DMS1 and DMS0 specify the DMA request sense for channel 0 ($\overline{DREQ}_0$) and channel 1 ($\overline{DREQ}_1$) respectively. When reset to 0, the input is level sense. When set to 1, the input is edge sense. DMS1 and DMS0 are cleared to 0 during RESET.

DIM1, DIM0: DMA Channel 1 I/O and Memory Mode (bits 1-0)

Specifies the source/destination and address modifier for channel 1 memory ⟷ I/O transfer modes. IM1 and IM0 are cleared to 0 during RESET.

Table 8 Channel 1 Transfer Mode

DIM1	DIM0	Transfer Mode	Address Increment/Decrement
0	0	Memory→I/O	MAR1+1, IAR1 fixed
0	1	Memory→I/O	MAR1−1, IAR1 fixed
1	0	I/O→Memory	IAR1 fixed, MAR1+1
1	1	I/O→Memory	IAR1 fixed, MAR1−1

10.3 DMA Operation

This section discusses the three DMA operation modes for channel 0, memory ⟷ memory, memory ⟷ I/O and memory ⟷ memory mapped I/O. In addition, the operation of channel 0 DMA with the on-chip ASCI (Asynchronous Serial Communication Interface) as well as Channel 1 DMA are described.

(1) Memory <——> Memory – Channel 0

For memory <——> memory transfers, the external $\overline{DREQ_0}$ input is not used for DMA transfer timing. Rather, the DMA operation is timed in one of two programmable modes – burst or cycle steal. In both modes, the DMA operation will automatically proceed until termination as shown by byte count (BCR0) = 0.

In burst mode, the DMA operation will proceed until termination. In this case, the CPU cannot perform any program execution until the DMA operation is completed.

In cycle steal mode, the DMA and CPU operation are alternated after each DMA byte transfer until the DMA is completed. The sequence ...

1 CPU Machine Cycle
DMA Byte Transfer

. is repeated until DMA is completed. Fig. 41 shows cycle steal mode DMA timing.

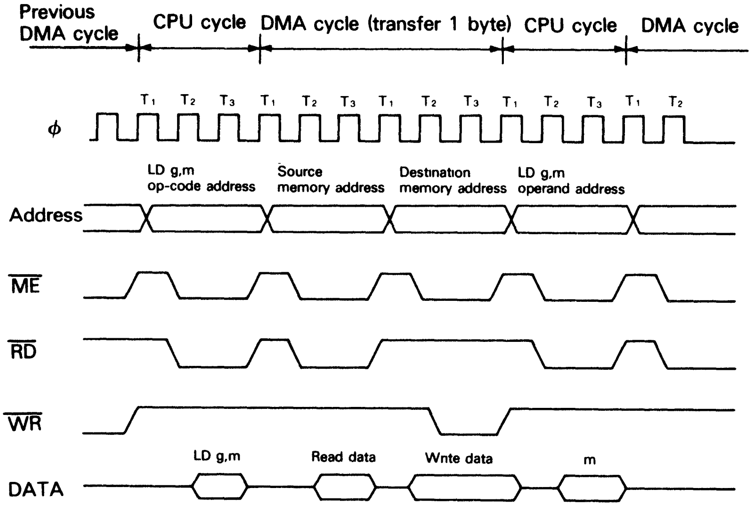


Figure 41 Cycle Steal Mode DMA Timing

To initiate memory <——> memory DMA transfer for channel 0, perform the following operations.

① Load the memory source and destination addresses into SAR0 and DAR0.
② Specify memory <——> memory mode and address increment/decrement in the SM0, SM1, DM0 and DM1 bits of DMODE.
③ Load the number of bytes to transfer in BCR0.
④ Specify burst or cycle steal mode in the MMOD bit of DCNTL.
⑤ Program DE0 = 1 (with $\overline{DWE0}$ = 0 in the same access) in DSTAT and the DMA operation will start 1 machine cycle later. If interrupt occurs at the same time, the DIE0 bit should be set to 1.

(2) Memory <——> I/O (Memory Mapped I/O) – Channel 0

For memory <——> I/O (and memory <——> memory mapped I/O) the $\overline{DREQ_0}$ input is used to time the DMA transfers. In addition, the $\overline{TEND_0}$ (Transfer End) output is used to indicate the last (byte count register BCR0 = 00H) transfer.

The $\overline{DREQ_0}$ input can be programmed as level or edge sensitive.

When level sense is programmed, the DMA operation begins when $\overline{DREQ_0}$ is sampled LOW. If $\overline{DREQ_0}$ is sampled HIGH, after the next DMA byte transfer, control is relinquished to the HD64180 CPU. As shown in Fig. 42. $\overline{DREQ_0}$ is sampled at the rising edge of the clock cycle prior to T_3 i.e. either T_2 or Tw.

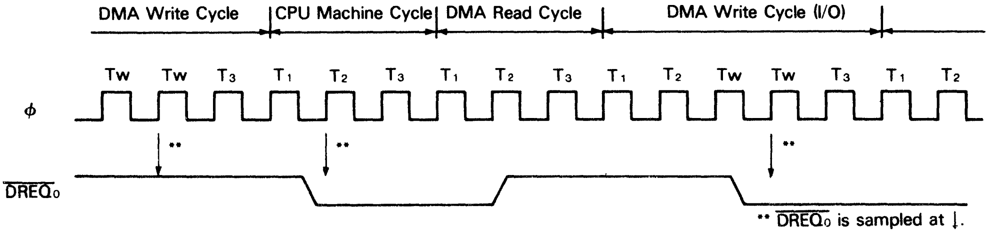


Figure 42 CPU Operation and DMA Operation
($\overline{DREQ_0}$ is programmed for level sense)

When edge sense is programmed, DMA operation begins at the falling edge of $\overline{DREQ_0}$. If another falling edge is detected before the rising edge of the clock prior to T_3 during DMA write cycle (i.e. T_2 or Tw), the DMAC continues operating. If an edge is not detected, the CPU is given control after the current byte DMA transfer completes. The CPU will continue operating until a $\overline{DREQ_0}$ falling edge is detected before the rising edge of the clock prior to T_3 at which time the DMA operation will (re)start. Fig. 43 shows the edge sense DMA timing.

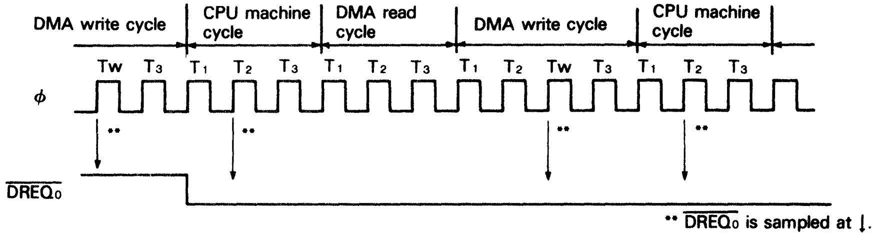


Figure 43 CPU Operation and DMA Operation ($\overline{DREQ_0}$ is programmed for edge sense)

During the transfers for channel 0, the $\overline{TEND_0}$ output will go LOW synchronous with the write cycle of the last (BCR0 = 00H) DMA transfer as shown in Fig. 44.

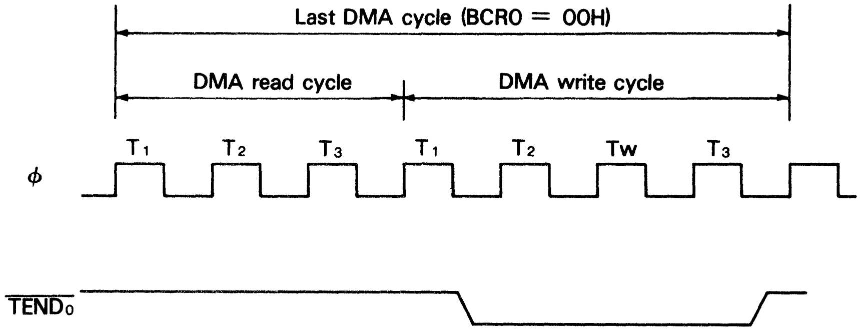


Figure 44 $\overline{TEND_0}$ Output Timing

The $\overline{DREQ_0}$ and $\overline{TEND_0}$ pins are programmably multiplexed with the CKA0 and CKA1 ASCI clock input/outputs. However, when DMA channel 0 is programmed for memory ⟷ I/O (and memory ⟷ memory mapped I/O) transfers, the CKA0/$\overline{DREQ_0}$ pin automatically functions as input pin even if it has been programmed as output pin for CKA0. And the CKA1/$\overline{TEND_0}$ pin functions as output pin for $\overline{TEND_0}$ by setting CKA1D to 1 in CNTLA1.

To initiate memory ⟷ I/O (and memory ⟷ memory mapped I/O) DMA transfer for channel 0, perform the following operations.

① Load the memory and I/O or memory mapped I/O source and destination addresses into SAR0 and DAR0. Note that I/O addresses (not memory mapped I/O) are limited to 16 bits (A_0-A_{15}). Make sure that bits A_{16} and A_{17} are 0 (A_{18} is a don't care) to correctly enable the external $\overline{DREQ_0}$ input.

② Specify memory ⟷ I/O or memory ⟷ memory mapped I/O mode and address increment/decrement in the SM0, SM1, DM0, and DM1 bits of DMODE.

③ Load the number of bytes to transfer in BCR0.

④ Specify whether $\overline{DREQ_0}$ is edge or level sense by programming the DMS0 bit of DCNTL.

⑤ Enable or disable DMA termination interrupt with the DIE0 bit in DSTAT.

⑥ Program DE0 = 1 (with $\overline{DWE0}$ = 0 in the same access) in DSTAT and the DMA operation will begin under the control of the $\overline{DREQ_0}$ input.

(3) Memory ⟷ ASCI – Channel 0

Channel 0 has extra capability to support DMA transfer to and from the on-chip two channel ASCI. In this case the external $\overline{DREQ_0}$ input is not used for DMA timing. Rather, the ASCI status bits are used to generate an internal $\overline{DREQ_0}$. The TDRE (Transmit Data Register Empty) bit and the RDRF (Receive Data Register Full) bit are used to generate an internal $\overline{DREQ_0}$ for ASCI transmission and reception respectively.

To initiate memory ⟷ ASCI DMA transfer, perform the following operations.

① Load the source and destination addresses into SAR0 and DAR0. Specify the I/O (ASCI) address as follows.
Bits A_0-A_7 should be contain the address of the ASCI channel transmitter or receiver (I/O addresses 06H-09H).
Bits A_8-A_{15} should equal 0.
Bits A_{17}-A_{16} should be set according to the following table to enable use of the appropriate ASCI status bit as an internal DMA request.

Table 9 DMA Request

SAR18	SAR17	SAR16	DMA Transfer Request
X	0	0	$\overline{\text{DREQ}}_0$
X	0	1	RDRF (ASCI channel 0)
X	1	0	RDRF (ASCI channel 1)
X	1	1	reserved

X Don't care

DAR18	DAR17	DAR16	DMA Transfer Request
X	0	0	$\overline{\text{DREQ}}_0$
X	0	1	TDRE (ASCI channel 0)
X	1	0	TDRE (ASCI channel 1)
X	1	1	reserved

X Don't care

② Specify memory ↔ I/O transfer mode and address increment/decrement in the SM0, SM1, DM0 and DM1 bits of DMODE.

③ Load the number of bytes to transfer in BCR0.

④ The DMA request sense mode (DMS0 bit in DCNTL) MUST be specified as 'edge sense'.

⑤ Enable or disable DMA termination interrupt with the DIE0 bit in DSTAT

⑥ Program DE0 = 1 (with $\overline{\text{DWE0}}$ = 0 in the same access) in DSTAT and the DMA operation with the ASCI will begin under control of the ASCI generated internal DMA request.

The ASCI receiver or transmitter being used for DMA must be initialized to allow the first DMA transfer to begin.

The ASCI receiver must be 'empty' as shown by RDRF = 0.

The ASCI transmitter must be 'full' as shown by TDRE = 0. Thus, the first byte should be written to the ASCI Transmit Data Register under program control. The remaining bytes will be transferred using DMA.

(4) Channel 1 DMA

DMAC Channel 1 can perform memory ↔ I/O transfers. Except for different registers and status/control bits, operation is exactly the same as described for channel 0 memory ↔ I/O DMA.

To initiate DMA channel 1 memory ↔ I/O transfer perform the following operations.

① Load the memory address (19 bits) into MAR1.

② Load the I/O address (16 bits) into IAR1.

③ Program the source/destination and address increment/decrement mode using the DIM1 and DIM0 bits in DCNTL.

④ Specify whether $\overline{\text{DREQ}}_1$ is level or edge sense in the DMS1 bit in DCNTL.

⑤ Enable or disable DMA termination interrupt with the DIE1 bit in DSTAT.

⑥ Program DE1 = 1 (with $\overline{\text{DWE1}}$ = 0 in the same access) in DSTAT and the DMA operation with the external I/O device will begin using the external $\overline{\text{DREQ}}_1$ input and $\overline{\text{TEND}}_1$ output.

10.4 DMA Bus Timing

When memory (and memory mapped I/O) is specified as a source or destination, $\overline{\text{ME}}$ goes LOW during the memory access. When I/O is specified as a source or destination, $\overline{\text{IOE}}$ goes LOW during the I/O access.

When I/O (and memory mapped I/O) is specified as a source or destination, the DMA timing is controlled by the external $\overline{\text{DREQ}}$ input and the $\overline{\text{TEND}}$ output indicates DMA termination. Note that external I/O devices may not overlap addresses with internal I/O and control registers, even using DMA.

For I/O accesses, 1 wait state is automatically inserted. Additional wait states can be inserted by programming the on-chip wait state generator or using the external $\overline{\text{WAIT}}$ input. Note that for memory mapped I/O accesses, this automatic I/O wait state is not inserted.

For memory to memory transfers (channel 0 only), the external $\overline{\text{DREQ}}_0$ input is ignored. Automatic DMA timing is programmed as either burst or cycle steal.

When a DMA memory address carry/borrow between bits A_{15} and A_{16} of the address bus occurs (when crossing 64k bytes boundaries), the minimum bus cycle is extended to four clocks by automatic insertion of one internal Ti state.

10.5 DMAC Channel Priority

For simultaneous $\overline{\text{DREQ}}_0$ and $\overline{\text{DREQ}}_1$ requests, channel 0 has priority over channel 1. When channel 0 is performing a memory ↔ memory transfer, channel 1 cannot operate until the channel 0 operation has terminated. If channel 1 is operating, channel 0 cannot operate until channel 1 releases control of the bus.

10.6 DMAC and $\overline{\text{BUSREQ}}$, $\overline{\text{BUSACK}}$

The $\overline{\text{BUSREQ}}$ and $\overline{\text{BUSACK}}$ inputs allow another bus master to take control of the HD64180 bus. $\overline{\text{BUSREQ}}$ and $\overline{\text{BUSACK}}$ have priority over the on-chip DMAC and will suspend DMAC operation. The DMAC releases the bus to the external bus master at the breakpoint of the DMAC memory or I/O access. Since a single byte DMAC transfer requires a read and a write cycle, it is possible for the DMAC to be suspended after the DMAC read, but before the DMAC write. Even in this case, when the external master releases the HD64180 bus ($\overline{\text{BUSREQ}}$ HIGH), the on-chip DMAC will correctly continue the suspended DMA operation.

10.7 DMAC Internal Interrupts

Fig. 45 illustrates the internal DMA interrupt request generation circuit.

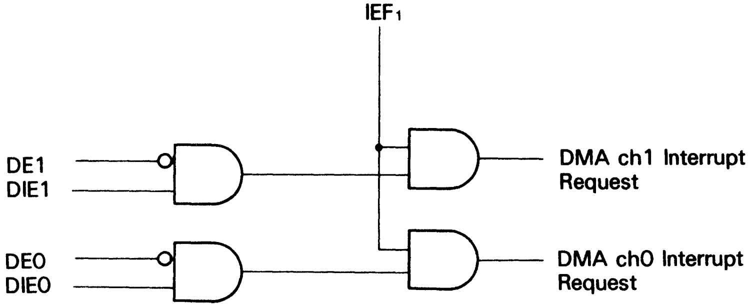


Figure 45 DMAC Interrupt Request Circuit Diagram

DE0 and DE1 are automatically cleared to 0 by the HD64180 at the completion (byte count = 0) of a DMA operation for channel 0 and channel 1 respectively. They remain 0 until a 1 is written. Since DE0 and DE1 use level sense, an interrupt will occur if the CPU IEF_1 flag is set to 1. Therefore, the DMA termination interrupt service routine should disable further DMA interrupts (by programming the channel DIE bit = 0) before enabling CPU interrupts (i.e. IEF_1 is set to 1). After reloading the DMAC address and count registers, the DIE bit can be set to 1 to reenable the channel interrupt, and at the same time DMA can resume by programming the channel DE bit = 1.

10.8 DMAC and $\overline{\text{NMI}}$

$\overline{\text{NMI}}$, unlike all other interrupts, automatically disables DMAC operation by clearing the DME bit of DSTAT. Thus, the $\overline{\text{NMI}}$ interrupt service routine may respond to time critical events without delay due to DMAC bus usage. Also, $\overline{\text{NMI}}$ can be effectively used as an external DMA abort input, recognizing that both channels are suspended by the clearing of DME.

If the falling edge of $\overline{\text{NMI}}$ occurs before the falling clock of the state prior to T_3 (T_2 or Tw) of the DMA write cycle, the DMAC will be suspended and the CPU will start the $\overline{\text{NMI}}$ response at the end of the current cycle.

By setting a channels DE bit to 1, that channels operation can be restarted, and DMA will correctly resume from the point at which it was suspended by NMI. See Fig. 46 for details.

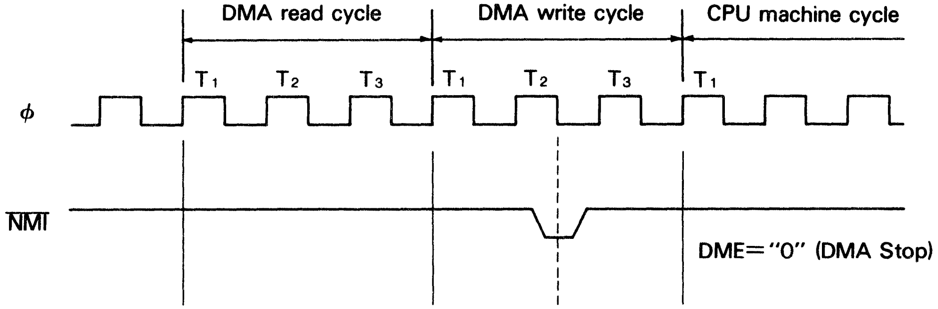


Figure 46 $\overline{\text{NMI}}$ and DMA Operation

10.9 DMAC and RESET

During RESET the bits in DSTAT, DMODE, and DCNTL are initialized as stated in their individual register descriptions. Any DMA operation in progress is stopped allowing the CPU to use the bus to perform the RESET sequence. However, the address register (SAR0, DAR0, MAR1, IAR1) and byte count register (BCR0, BCR1) contents are not changed during RESET.

11 ASYNCHRONOUS SERIAL COMMUNICATION INTERFACE (ASCI)

The HD64180 on-chip ASCI has two independent full duplex channels. Based on full programmability of the following functions, the ASCI can directly communicate with a wide variety of standard UARTs (Universal Asynchronous Receiver/Transmitter) including the HD6350 CMOS ACIA and the Serial Communication Interface (SCI) contained on the HD6301 series CMOS single chip controllers.

The key functions for ASCI are shown below. Each channel is independently programmable.

- Full duplex communication
- 7- or 8-bit data length
- Program controlled 9th data bit for multiprocessor communication
- 1 or 2 stop bits
- Odd, even, no parity
- Parity, overrun, framing error detection
- Programmable baud rate generator, /16 and /64 modes Speed to 38.4k bits per second (CPU $f_C = 6.144$ MHz)
- Modem control signals — Channel 0 has $\overline{DCD_0}$, $\overline{CTS_0}$ and $\overline{RTS_0}$ Channel 1 has $\overline{CTS_1}$
- Programmable interrupt condition enable and disable
- Operation with on-chip DMAC

11.1 ASCI Block Diagram

Fig. 47 shows the ASCI Block Diagram.

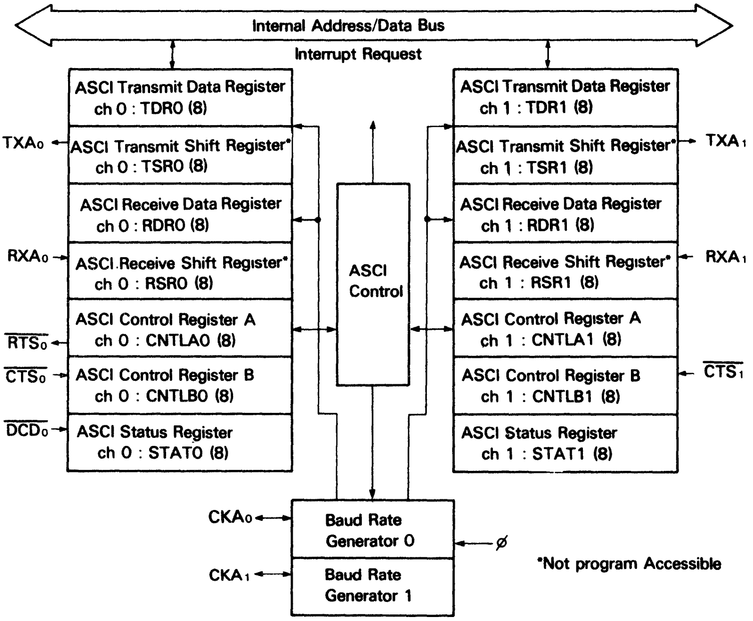

Internal Address/Data Bus
Interrupt Request
ASCI Transmit Data Register ch 0 : TDR0 (8)
ASCI Transmit Shift Register* ch 0 : TSR0 (8)
ASCI Receive Data Register ch 0 : RDR0 (8)
ASCI Receive Shift Register* ch 0 : RSR0 (8)
ASCI Control Register A ch 0 : CNTLA0 (8)
ASCI Control Register B ch 0 : CNTLB0 (8)
ASCI Status Register ch 0 : STAT0 (8)
ASCI Transmit Data Register ch 1 : TDR1 (8)
ASCI Transmit Shift Register* ch 1 : TSR1 (8)
ASCI Receive Data Register ch 1 : RDR1 (8)
ASCI Receive Shift Register* ch 1 : RSR1 (8)
ASCI Control Register A ch 1 : CNTLA1 (8)
ASCI Control Register B ch 1 : CNTLB1 (8)
ASCI Status Register ch 1 : STAT1 (8)
ASCI Control
TXA0
RXA0
RTS0
CTS0
DCD0
TXA1
RXA1
CTS1
CKA0
CKA1
Baud Rate Generator 0
Baud Rate Generator 1
ø
*Not program Accessible

Figure 47 ASCI Block Diagram

11.2 ASCI Register Description

(1) ASCI Transmit Shift Register 0, 1 (TSR0, 1)

When the ASCI Transmit Shift Register receives data from the ASCI Transmit Data Register (TDR), the data is shifted out to the TXA pin. When transmission is completed, the next byte (if available) is automatically loaded from TDR into TSR and the next transmission starts. If no data is available for transmission, TSR idles by outputting a continuous HIGH level. This register is not program accessible.

(2) ASCI Transmit Data Register 0, 1 (TDR0, 1: I/O Address = 06H, 07H)

Data written to the ASCI Transmit Data Register is transferred to the TSR as soon as TSR is empty. Data can be written to while TSR is shifting out the previous byte of data. Thus, the ASCI transmitter is double bufferred.

Data can be written into and read from the ASCI Transmit Data Register.

If data is read from the ASCI Transmit Data Register, the ASCI data transmit operation won't be affected by this read operation.

(3) ASCI Receive Shift Register 0, 1 (RSR0, 1)

This register receives data shifted in on the RXA pin. When full, data is automatically transferred to the ASCI Receive Data Register (RDR) if it is empty. If RSR is not empty when the next incoming data byte is shifted in, an overrun error occurs. This register is not program accessible.

(4) ASCI Receive Data Register 0, 1 (RDR0, 1: I/O Address = 08H, 09H)

When a complete incoming data byte is assembled in RSR, it is automatically transferred to the RDR if RDR is empty. The next incoming data byte can be shifted into RSR while RDR contains the previous received data byte. Thus, the ASCI receiver is double buffered.

The ASCI Receive Data Register is read-only-register.

However, if RDRF = 0, data can be written into the ASCI Receive Data Register, and the data can be read.

(5) ASCI Status Register 0, 1 (STAT0, 1)

Each channel status register allows interrogation of ASCI communication, error and modem control signal status as well as enabling and disabling of ASCI interrupts.

ASCI Status Register 0 (STAT0 I/O Address = 04H)

bit	7	6	5	4	3	2	1	0
	RDRF	OVRN	PE	FE	RIE	$\overline{DCD_0}$	TDRE	TIE
	R	R	R	R	R/W	R	R	R/W

ASCI Status Register 1 (STAT1 I/O Address = 05H)

bit	7	6	5	4	3	2	1	0
	RDRF	OVRN	PE	FE	RIE	CTS1E	TDRE	TIE
	R	R	R	R	R/W	R/W	R	R/W

RDRF: Receive Data Register Full (bit 7)

RDRF is set to 1 when an incoming data byte is loaded into RDR. Note that if a framing or parity error occurs, RDRF is still set and the receive data (which generated the error) is still loaded into RDR. RDRF is cleared to 0 by reading RDR, when the $\overline{DCD_0}$ input is HIGH, in IOSTOP mode and during RESET.

OVRN: Overrun Error (bit 6)

OVRN is set to 1 when RDR is full and RSR becomes full. OVRN is cleared to 0 when the EFR bit (Error Flag Reset) of CNTLA is written to 0, when $\overline{DCD_0}$ is HIGH, in IOSTOP mode and during RESET.

PE: Parity Error (bit 5)

PE is set to 1 when a parity error is detected on an incoming data byte and ASCI parity detection is enabled (the MOD1 bit of CNTLA is set to 1). PE is cleared to 0 when the EFR bit (Error Flag Reset) of CNTLA is written to 0, when $\overline{DCD_0}$ is HIGH, in IOSTOP mode and during RESET.

FE: Framing Error (bit 4)

If a receive data byte frame is delimited by an invalid stop bit (i.e. 0, should be 1), FE is set to 1. FE is cleared to 0 when the EFR bit (Error Flag Reset) of CNTLA is written to 0, when $\overline{DCD_0}$ is HIGH, in IOSTOP mode and during RESET.

RIE: Receive Interrupt Enable (bit 3)

RIE should be set to 1 to enable ASCI receive interrupt requests. When RIE to 1, if any of the flags RDRF, OVRN, PE, FE become set to 1 an interrupt request is generated. For channel 0, an interrupt will also be generated by the transition of the external $\overline{DCD_0}$ input from LOW to HIGH. RIE is cleared to 0 during RESET.

$\overline{DCD_0}$: Data Carrier Detect (bit 2 STAT0)

Channel 0 has an external $\overline{DCD_0}$ input pin. The $\overline{DCD_0}$ bit is set to 1 when the $\overline{DCD_0}$ input is HIGH. It is cleared to 0 on the first read of STAT0 following the HIGH to LOW transition of $\overline{DCD_0}$ input and during RESET. When $\overline{DCD_0}$ = 1, receiver unit is reset and receiver operation is inhibited.

CTS1E: Channel 1 $\overline{CTS}$ Enable (bit 2 STAT1)

Channel 1 has an external $\overline{CTS_1}$ input which is multiplexed with the receive data pin (RXS) for the CSI/O (Clocked Serial I/O Port). Setting CTS1E to 1 selects the $\overline{CTS_1}$ function and clearing CTS1E to 0 selects the RXS function.

TDRE: Transmit Data Register Empty (bit 1)

TDRE = 1 indicates that the TDR is empty and the next transmit data byte can be written to TDR. After the byte is written to TDR, TDRE is cleared to 0 until the ASCI transfers the byte from the TDR to the TSR, at which time TDRE is again set to 1. TDRE is set to 1 in IOSTOP mode and during RESET. When the external $\overline{CTS}$ input is HIGH, TDRE is reset to 0.

TIE: Transmit Interrupt Enable (bit 0)

TIE should be set to 1 to enable ASCI transmit interrupt requests. If TIE = 1, an interrupt will be requested when TDRE = 1. TIE is cleared to 0 during RESET.

· ASCI Control Register A0, 1 (CNTLA0, 1)

Each ASCI channel Control Register A configures the major operating modes such as receiver/transmitter enable and disable, data format, and multiprocessor communication mode.

ASCI Control Register A 0 (CNTLA0 I/O Address = 00H)

bit	7	6	5	4	3	2	1	0
	MPE	RE	TE	$\overline{RTS_0}$	MPBR/ EFR	MOD2	MOD1	MOD0
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

ASCI Control Register A 1 (CNTLA1 I/O Address = 01H)

bit	7	6	5	4	3	2	1	0
	MPE	RE	TE	CKA1D	MPBR/ EFR	MOD2	MOD1	MOD0
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

MPE: Multi Processor Mode Enable (bit 7)

The ASCI has a multiprocessor communication mode which utilizes an extra data bit for selective communication when a number of processors share a common serial bus. Multiprocessor data format is selected when the MP bit in CNTLB is set to 1. If multiprocessor mode is not selected (MP bit in CNTLB = 0), MPE has no effect. If multiprocessor mode is selected, MPE enables or disables the 'wake-up' feature as follows. If MPE is set to 1, only received bytes in which the MPB (multiprocessor bit) = 1 can affect the RDRF and error flags. Effectively, other bytes (with MPB = 0) are 'ignored' by the ASCI. If MPE is reset to 0, all bytes, regardless of the state of the MPB data bit, affect the RDRF and error flags. MPE is cleared to 0 during RESET.

RE: Receiver Enable (bit 6)

When RE is set to 1, the ASCI receiver is enabled. When RE is cleared to 0, the receiver is disabled and any receive operation in progress is interrupted. However, the RDRF and error flags are not reset and the previous contents of RDRF and error flags are held. RE is cleared to 0 in IOSTOP mode and during RESET.

TE: Transmitter Enable (bit 5)

When TE is set to 1, the ASCI transmitter is enabled. When TE is cleared to 0, the transmitter is disabled and any transmit operation in progress is interrupted. However, the TDRE flag is not reset and the previous contents of TDRE are held. TE is cleared to 0 in IOSTOP mode and during RESET.

$\overline{RTS_0}$ – Request to Send Channel 0 (bit 4 in CNTLA0)

When $\overline{RTS_0}$ is cleared to 0, the $\overline{RTS_0}$ output pin will go LOW. When $\overline{RTS_0}$ is set to 1, the $\overline{RTS_0}$ output immediately goes HIGH. $\overline{RTS_0}$ is set to 1 during RESET.

CKA1D: CKA1 Clock Disable (bit 4 in CNTLA1)

When CKA1D is set to 1, the multiplexed CKA1/$\overline{TEND_0}$ pin is used for the $\overline{TEND_0}$ function. When CKA1D = 0, the pin is used as CKA1, an external data clock input/output for channel 1. CKA1D is cleared to 0 during RESET.

MPBR/EFR: Multiprocessor Bit Receive/Error Flag Reset (bit 3)

When multiprocessor mode is enabled (MP in CNTLB = 1), MPBR, when read, contains the value of the MPB bit for the last receive operation. When written to 0, the EFR function is selected to reset all error flags (OVRN, FE and PE) to 0. MPBR/EFR is undefined during RESET.

MOD2, 1, 0: ASCI Data Format Mode 2, 1, 0 (bits 2-0)

These bits program the ASCI data format as follows.

MOD2
= 0 → 7 bit data
= 1 → 8 bit data
MOD1
= 0 → No parity
= 1 → Parity enabled
MOD0
= 0 → 1 stop bit
= 1 → 2 stop bits

The data formats available based on all combinations of MOD2, MOD1 and MOD0 are shown in Table 10.

Table 10 Combination of Data Format

MOD2	MOD1	MOD0	Data Format
0	0	0	Start+7 bit data+1 stop
0	0	1	Start+7 bit data+2 stop
0	1	0	Start+7 bit data+parity+1 stop
0	1	1	Start+7 bit data+parity+2 stop
1	0	0	Start+8 bit data+1 stop
1	0	1	Start+8 bit data+2 stop
1	1	0	Start+8 bit data+parity+1 stop
1	1	1	Start+8 bit data+parity+2 stop

(6) ASCI Control Register B0, 1 (CNTLB0, 1)

Each ASCI channel control register B configures multiprocessor mode, parity and baud rate selection.

ASCI Control Register B 0 (CNTLB0 : I/O Address = 02H)
ASCI Control Register B 1 (CNTLB1 . I/O Address = 03H)

bit	7	6	5	4	3	2	1	0
	MPBT	MP	$\overline{\text{CTS}}$/ PS	PEO	DR	SS2	SS1	SS0
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

MPBT: Multiprocessor Bit Transmit (bit 7)

When multiprocessor communication format is selected (MP bit = 1), MPBT is used to specify the MPB data bit for transmission. If MPBT = 1, then MPB = 1 is transmitted. If MPBT = 0, then MPB = 0 is transmitted. MPBT state is undefined during and after RESET.

MP: Multiprocessor Mode (bit 6)

When MP is set to 1, the data format is configured for multiprocessor mode based on the MOD2 (number of data bits) and MOD0 (number of stop bits) bits in CNTLA. The format is as follows.

Start bit + 7 or 8 data bits + MPB bit + 1 or 2 stop bits

Note that multiprocessor (MP = 1) format has no provision for parity. If MP = 0, the data format is based on MOD0, MOD1 and MOD2 and may include parity. The MP bit is cleared to 0 during RESET.

$\overline{\text{CTS}}$/PS: Clear to Send/Prescale (bit 5)

When read, $\overline{\text{CTS}}$/PS reflects the state of the external $\overline{\text{CTS}}$ input. If the $\overline{\text{CTS}}$ input pin is HIGH, $\overline{\text{CTS}}$/PS will be read as 1. Note that when the $\overline{\text{CTS}}$ input pin is HIGH, the TDRE bit is inhibited (i.e. held at 0). For channel 1, the $\overline{\text{CTS}}_1$ input is multiplexed with RXS pin (Clocked Serial Receive Data). Thus, $\overline{\text{CTS}}$/PS is only valid when read if the channel 1 CTS1E bit = 1 and the $\overline{\text{CTS}}_1$ input pin function is selected. The read data of $\overline{\text{CTS}}$/PS is not affected by RESET.

When written, $\overline{\text{CTS}}$/PS specifies the baud rate generator prescale factor. If $\overline{\text{CTS}}$/PS is set to 1, the system clock (ϕ) is prescaled by 30 while if $\overline{\text{CTS}}$/PS is cleared to 0, the system clock is prescaled by 10. $\overline{\text{CTS}}$/PS is cleared to 0 during RESET.

PEO: Parity Even Odd (bit 4)

PEO selects even or odd parity. PEO does not affect the enabling/disabling of parity (MOD1 bit of CNTLA). If PEO is cleared to 0, even parity is selected. If PEO is set to 1, odd parity is selected. PEO is cleared to 0 during RESET.

DR: Divide Ratio (bit 3)

DR specifies the divider used to obtain baud rate from the data sampling clock. If DR is cleared to 0, divide by 16 is used while if DR is set to 1, divide by 64 is used. DR is cleared to 0 during RESET.

SS2, 1, 0: Source/Speed Select 2, 1, 0 (bits 2-0)

Specify the data clock source (internal or external) and baud rate prescale factor. SS2, SS1, and SS0 are all set to 1 during RESET. Table 11 shows the divide ratio corresponding to SS2, SS1, and SS0.

Table 11 Divide Ratio

SS2	SS1	SS0	Divide Ratio
0	0	0	÷1
0	0	1	÷2
0	1	0	÷4
0	1	1	÷8
1	0	0	÷16
1	0	1	÷32
1	1	0	÷64
1	1	1	external clock

The external ASCI channel 0 data clock pins are multiplexed with DMA control lines ($\text{CKA}_0/\overline{\text{DREQ}}_0$ and $\text{CKA}_1/\overline{\text{TEND}}_0$). During RESET, these pins are initialized as ASCI data clock inputs. If SS2, SS1, and SS0 are reprogrammed (any other value than SS2, SS1, SS0 = 1) these pins become ASCI data clock outputs. However, if DMAC channel 0 is configured to perform memory ←→ I/O (and memory mapped I/O) transfers the $\text{CKA}_0/\overline{\text{DREQ}}_0$ pin revert to DMA control signals regardless of SS2, SS1, and SS0 programming. Also, if the CKA1D bit in the CNTLA register is set to 1, then the $\text{CKA}_1/\overline{\text{TEND}}_0$ reverts to the DMA Control output function regardless of SS2, SS1, and SS0 programming.

Final data clock rates are based on $\overline{\text{CTS}}$/PS (prescale), DR, SS2, SS1, SS0, and the HD64180 system clock (ϕ) frequency as shown in Table 12.

Table 12 Baud Rate List

Prescaler		Sampling Rate		Baud Rate				General Divide Ratio	Baud Rate (Example) (BPS)			CKA	
PS	Divide Ratio	DR	Rate	SS2	SS1	SS0	Divide Ratio		ϕ=6.144 MHz	ϕ=4.608 MHz	ϕ=3.072 MHz	I/O	Clock Frequency
0	ϕ÷10	0	16	0	0	0	÷1	ϕ÷160	38400		19200	O	ϕ÷10
				0	0	1	2	320	19200		9600		20
				0	1	0	4	640	9600		4800		40
				0	1	1	8	1280	4800		2400		80
				1	0	0	16	2560	2400		1200		160
				1	0	1	32	5120	1200		600		320
				1	1	0	64	10240	600		300		640
				1	1	1	–	fc÷16	–	–	–	I	fc
		1	64	0	0	0	÷1	ϕ÷640	9600		4800	O	ϕ÷10
				0	0	1	2	1280	4800		2400		20
				0	1	0	4	2560	2400		1200		40
				0	1	1	8	5120	1200		600		80
				1	0	0	16	10240	600		300		160
				1	0	1	32	20480	300		150		320
				1	1	0	64	40960	150		75		640
				1	1	1	–	fc÷64	–	–	–	I	fc
1	ϕ÷30	0	16	0	0	0	÷1	ϕ÷480		9600		O	ϕ÷30
				0	0	1	2	960		4800			60
				0	1	0	4	1920		2400			120
				0	1	1	8	3840		1200			240
				1	0	0	16	7680		600			480
				1	0	1	32	15360		300			960
				1	1	0	64	30720		150			1920
				1	1	1	–	fc÷16	–	–	–	I	fc
		1	64	0	0	0	÷1	ϕ÷1920		2400		O	ϕ÷30
				0	0	1	2	3840		1200			60
				0	1	0	4	7680		600			120
				0	1	1	8	15360		300			240
				1	0	0	16	30720		150			480
				1	0	1	32	61440		75			960
				1	1	0	64	122880		37 5			1920
				1	1	1	–	fc÷64	–	–	–	I	fc

11.3 MODEM Control Signals

ASCI channel 0 has $\overline{\mathrm{CTS}}_0$, $\overline{\mathrm{DCD}}_0$, and $\overline{\mathrm{RTS}}_0$ external modem control signals. ASCI channel 1 has a $\overline{\mathrm{CTS}}_1$ modem control signal which is multiplexed with RXS pin (Clocked Serial Receive Data).

(1) $\overline{\mathrm{CTS}}_0$: Clear to Send 0 (input)

The CTS_0 input allows external control (start/stop) of ASCI channel 0 transmit operations. When $\overline{\mathrm{CTS}}_0$ is HIGH, channel 0 TDRE bit is held at 0 regardless of whether the TDR0 (Transmit Data Register) is full or empty. When $\overline{\mathrm{CTS}}_0$ is LOW, TDRE will reflect the state of TDR0. Note that the actual transmit operation is not disabled by $\overline{\mathrm{CTS}}_0$ HIGH, only TDRE is inhibited.

(2) $\overline{\mathrm{DCD}}_0$: Data Carrier Detect 0 (input)

The $\overline{\mathrm{DCD}}_0$ input allows external control (start/stop) of ASCI channel 0 receive operations. When $\overline{\mathrm{DCD}}_0$ is HIGH, channel 0 RDRF bit is held at 0 regardless of whether the RDR0 (Receive Data Register) is full or empty. The error flags (PE, FE and OVRN bits) are also held at 0. Even after the $\overline{\mathrm{DCD}}_0$ input goes LOW, these bits will not resume normal operation until the status register (STAT0) is read. Note that this first read of STAT0, while enabling normal operation, will still indicate the $\overline{\mathrm{DCD}}_0$ input is HIGH ($\overline{\mathrm{DCD0}}$ bit = 1) even though it has gone LOW. Thus, the STAT0 register should be read twice to insure the $\overline{\mathrm{DCD0}}$ bit is cleared to 0.

(3) $\overline{\mathrm{RTS}}_0$: Request to Send 0 (output)

$\overline{\mathrm{RTS}}_0$ allows the ASCI to control (start/stop) another communication devices transmission (for example, by connection to that devices $\overline{\mathrm{CTS}}$ input). $\overline{\mathrm{RTS}}_0$ is essentially a 1 bit output port, having no side effects on other ASCI registers or flags.

(4) $\overline{\mathrm{CTS}}_1$: Clear to Send 1 (input)

Channel 1 CTS_1 input is multiplexed with the RXS pin (Clocked Serial Receive Data). The $\overline{\mathrm{CTS}}_1$ function is selected when the CTS1E bit in STAT1 is set to 1. When enabled, the $\overline{\mathrm{CTS}}_1$ operation is equivalent to $\overline{\mathrm{CTS}}_0$.

Modem control signal timing is shown in Fig. 48 (a) and Fig. 48 (b).

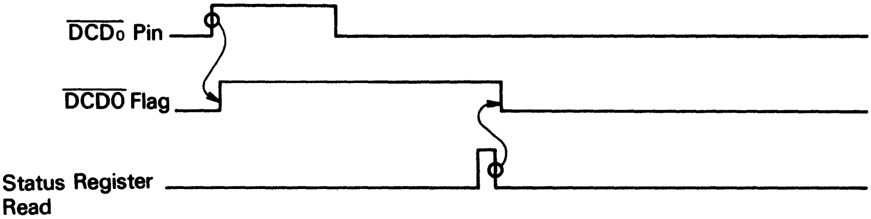


Figure 48 (a) $\overline{DCD_0}$ Timing

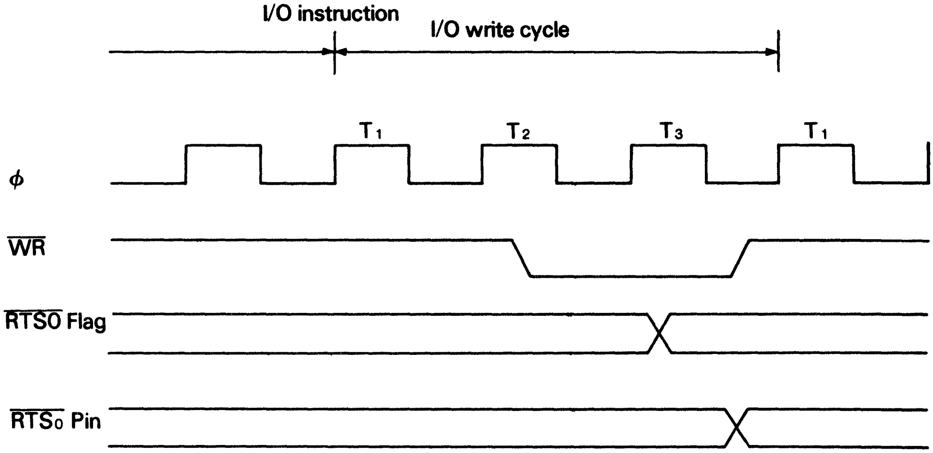


Figure 48 (b) $\overline{RTS_0}$ Timing

11.4 ASCI Interrupts

Fig. 49 shows the ASCI interrupt request generation circuit.

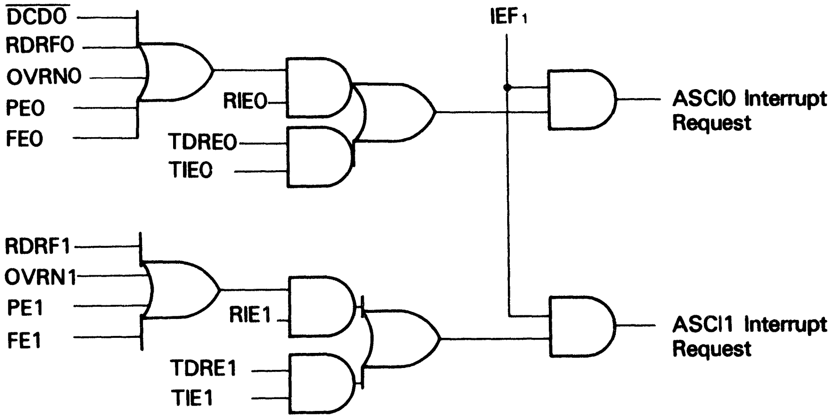


Figure 49 ASCI Interrupt Request Circuit Diagram

11.5 ASCI <——> DMAC operation

Operation of the ASCI with the on-chip DMAC channel 0 requires the DMAC be correctly configured to utilize the ASCI flags as DMA request signals.

11.6 ASCI and RESET

During RESET, the ASCI status and control registers are initialized as defined in the individual register descriptions.

Receive and Transmit operations are stopped durng RESET. However, the contents of the transmit and receive data registers (TDR and RDR) are not changed by RESET

11.7 ASCI Clock

In external clock input mode, the external clock is drrectly input to the sampling rate (÷16/÷64) as shown in Fig. 50.

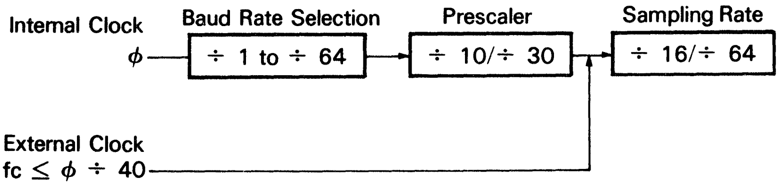


Figure 50 ASCI Clock Block Dıagram

12 CLOCKED SERIAL I/O PORT (CSI/O)

The HD64180 includes a simple, high speed clock synchronous serial I/O port. The CSI/O includes transmit/receive (half duplex), fixed 8-bit data and internal or external data clock selection. High speed operation (baud rate as high as 200k bits/second at $f_C = 4$ MHz) is provided. The CSI/O is ideal for implementing a multiprocessor communication link between the HD64180 and the HMCS400 series (4-bit) and the HD6301 series (8-bit) single chip controllers as well as additıonal HD64180s. These secondary devices may typically perform a portıon of the system I/O processing such as keyboard scan/decode, LDC interface etc

12.1 CSI/O Block Diagram

The CSI/O block diagram is shown in Fig 51. The CSI/O consists of two registers – the Transmit/Receive Data Register (TRDR) and Control Register (CNTR).

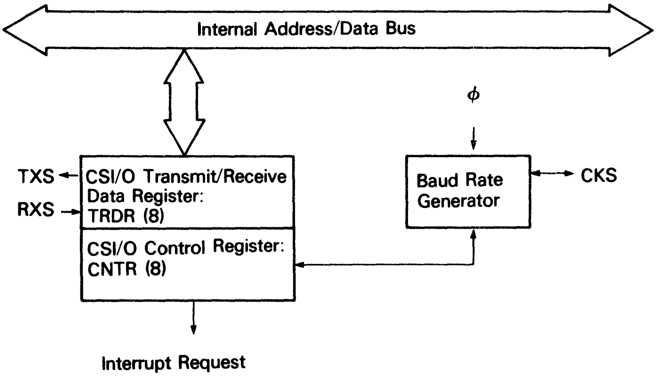

Internal Address/Data Bus
φ
TXS
RXS
CSI/O Transmit/Receive Data Register: TRDR (8)
CSI/O Control Register: CNTR (8)
Baud Rate Generator
CKS
Interrupt Request

Figure 51 CSI/O Block Dıagram

12.2 CSI/O Register Description

(1) CSI/O Transmit/Receive Data Register (TRDR: I/O Address = OBH)

TRDR is used for both CSI/O transmission and reception. Thus, the system design must insure that the constraints of half-duplex operation are met (Transmit and receive operation can't occur simultaneously). For example, if a CSI/O transmission is attempted at the same time that the CSI/O is receiving data, a CSI/O will not work. Also note that TRDR is not buffered. Therefore, attempting to perform a CSI/O transmit while the previous transmit data is still being shifted out causes the shift data to be immediately updated, thereby corrupting the transmit operation in progress. Similarly, reading TRDR while a transmit or receive is in progress should be avoided.

(2) CSI/O Control/Status Register (CNTR: I/O Address = OAH)

CNTR is used to monitor CSI/O status, enable and disable the CSI/O, enable and disable interrupt generation and select the data clock speed and source.

CSI/O Control Regıster (CNTR · I/O Address = OAH)

bıt	7	6	5	4	3	2	1	0
	EF	EIE	RE	TE	–	SS2	SS1	SSO
	R	R/W	R/W	R/W		R/W	R/W	R/W

EF: End Flag (bit 7)

EF is set to 1 by the CSI/O to indicate completion of an 8-bit data transmit or receive operation. If EIE (End Interrupt Enable) bit = 1 during the time EF = 1, a CPU interrupt request will be generated. Program access of TRDR should only occur if EF = 1. The CSI/O clears EF to 0 when TRDR is read or written. EF is cleared to 0 duıing RESET and IOSTOP mode.

EIE: End Interrupt Enable (bit 6)

EIE should be set to 1 to enable EF = 1 to generate a CPU interrupt request. The interrupt request is inhibited ıf EIE is cleared to 0. EIE is cleared to 0 during RESET.

RE: Receive Enable (bit 5)

A CSI/O receive operation is started by setting RE to 1. When RE is set to 1, the data clock is enabled. In internal clock mode, the data clock is output from the CKS pin. In external clock mode, the clock is input on the CKS pin. In either case, data is shifted in on the RXS pin in synchronization with the (internal or external) data clock. After receiving 8 bits of data, the CSI/O automatically clears RE to 0, EF is set to 1 and an interrupt (if enabled by EIE = 1) will be generated. Note that RE and TE should never both be set to 1 at the same time. RE is cleared to 0 during RESET and IOSTOP mode.

Note that the RXS pin (pin 52) is multiplexed with $\overline{CTS_1}$ modem control input of ASCI channel 1. In order to enable the RXS function, the CTS1E bit in CNTA1 should be reset to 0.

TE: Transmit Enable (bit 4)

A CSI/O transmit operation is started by setting TE to 1. When TE is set to 1, the data clock is enabled. In internal clock mode, the data clock is output from the CKS pin. In external clock mode, the clock is input on the CKS pin. In either case, data is shifted out on the TXS pin synchronous with the (internal or external) data clock. After transmitting 8 bits of data, the CSI/O automatically clears TE to 0, EF is set to 1 and an interrupt (if enabled by EIE = 1) will be generated. Note that TE and RE should never both be set to 1 at the same time. TE is cleared to 0 during RESET and IOSTOP mode.

SS2, 1, 0: Speed Select 2, 1, 0 (bits 2-0)

SS2, SS1 and SS0 select the CSI/O transmit/receive clock source and speed. SS2, SS1 and SS0 are all set to 1 during RESET. Table 13 shows CSI/O Baud Rate Selection.

Table 13 CSI/O Baud Rate Selection

SS2	SS1	SSO	Divide Ratio	Baud Rate
0	0	0	÷20	(200000)
0	0	1	÷40	(100000)
0	1	0	÷80	(50000)
0	1	1	÷160	(25000)
1	0	0	÷320	(12500)
1	0	1	÷640	(6250)
1	1	0	÷1280	(3125)
1	1	1	external Clock input (less than ÷20)	

( ) shows the baud rate (BPS) at ϕ = 4 MHz.

After RESET, the CKS pin is configured as an external clock input (SS2, SS1, SS0 = 1). Changing these values causes CKS to become an output pin and the selected clock will be output when transmit or receive operations are enabled.

12.3 CSI/O Interrupts

The CSI/O interrupt request circuit is shown in Fig. 52.

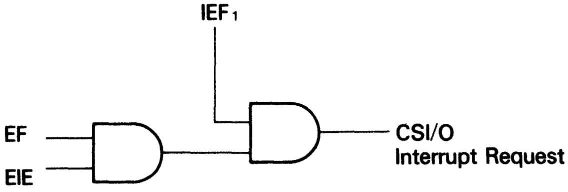


Figure 52 CSI/O Interrupt Circuit Diagram

12.4 CSI/O Operation

The CSI/O can be operated using status polling or interrupt driven algorithms.

(1) Transmit — Polling

① Poll the TE bit in CNTR until TE = 0.
② Write the transmit data into TRDR.
③ Set the TE bit in CNTR to 1.
④ Repeat 1 to 3 for each transmit data byte.

(2) Transmit — Interrupts

① Poll the TE bit in CNTR until TE = 0.
② Write the first transmit data byte into TRDR.
③ Set the TE and EIE bits in CNTR to 1.
④ When the transmit interrupt occurs, write the next transmit data byte into TRDR.
⑤ Set the TE bit in CNTR to 1.
⑥ Repeat 4 to 5 for each transmit data byte.

(3) Receive — Polling

① Poll the RE bit in CNTR until RE = 0.
② Set the RE bit in CNTR to 1.
③ Poll the RE bit in CNTR until RE = 0.
④ Read the receive data from TRDR.
⑤ Repeat 2 to 4 for each receive data byte.

(4) Receive — Interrupts

① Poll the RE bit in CNTR until RE = 0.
② Set the RE and EIE bits in CNTR to 1.
③ When the receive interrupt occurs read the receive data from TRDR.
④ Set the RE bit in CNTR to 1.
⑤ Repeat 3 to 4 for each receive data byte.

12.5 CSI/O Operation Timing Notes

(1) Note that transmitter clocking and receiver sampling timings are different from internal and external clocking modes. Fig. 53 to Fig. 54 shows CSI/O Transmit/Receive Timing.
(2) The transmitter and receiver should be disabled (TE and RE = 0) when initializing or changing the baud rate.

12.6 CSI/O Operation Notes

(1) Disable the transmitter and receiver (TE and RE = 0) before initializing or changing the baud rate. When changing the baud rate after completion of transmission or reception, a delay of at least one bit time is required before baud rate modification.

(2) When RE or TE is cleared to 0 by software, a corresponding receive or transmit operation is immediately terminated. Normally, TE or RE should only be cleared to 0 when EF = 1.

(3) Simultaneous transmission and reception is not possible. Thus, TE and RE should not both be 1 at the same time.

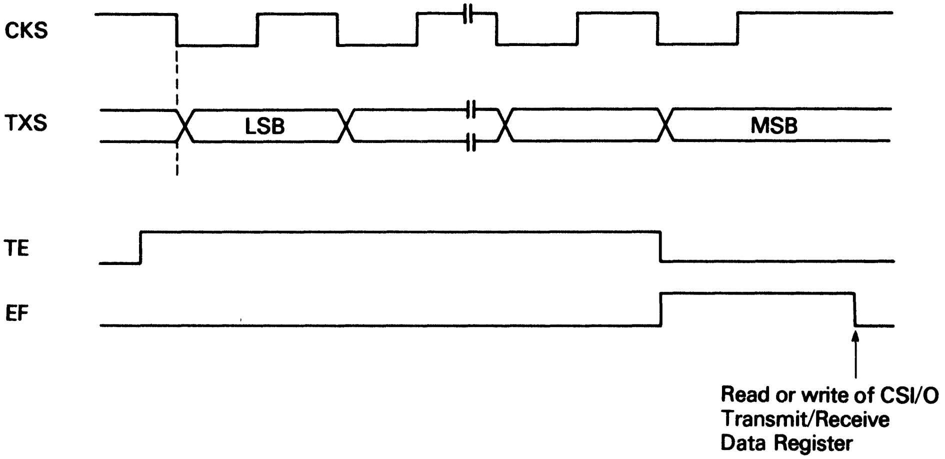


Figure 53 Transmit Timing — Internal Clock

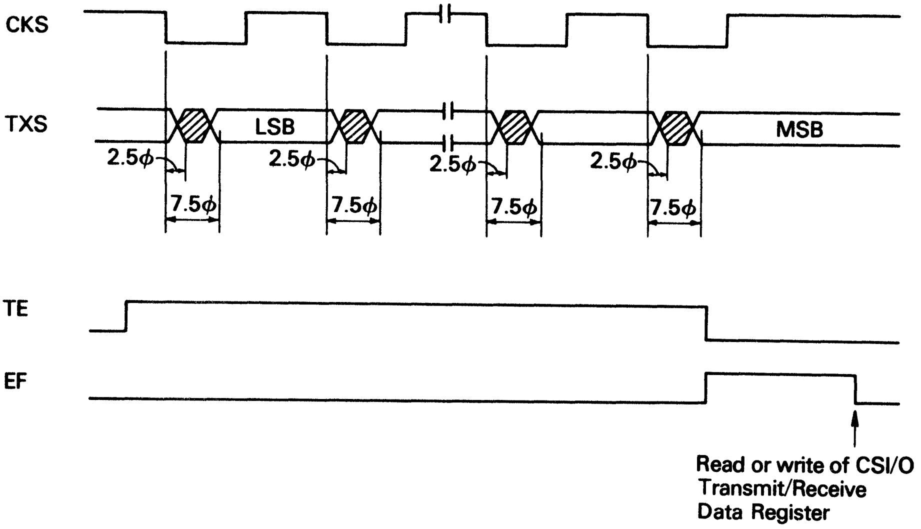


Figure 54 Transmit Timing — External Clock

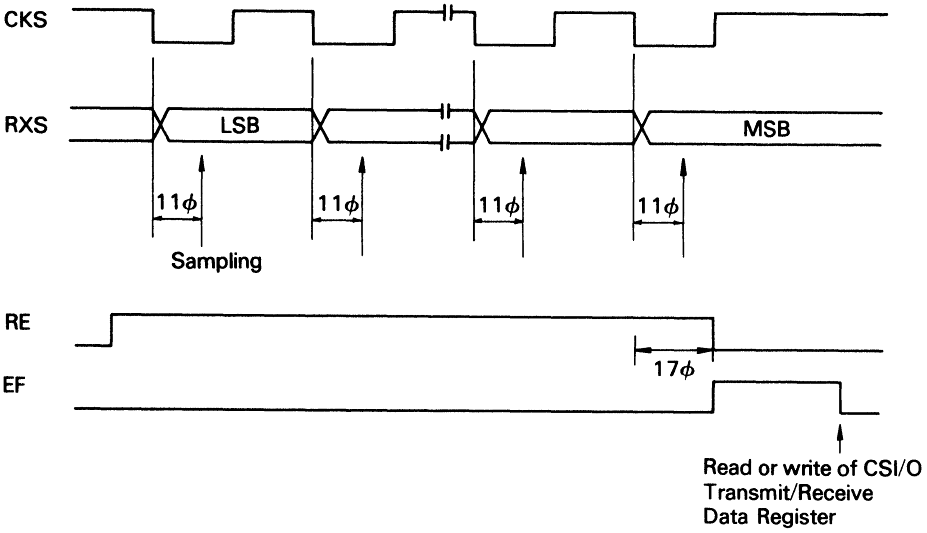


Figure 55 Receive Timing — Internal Clock

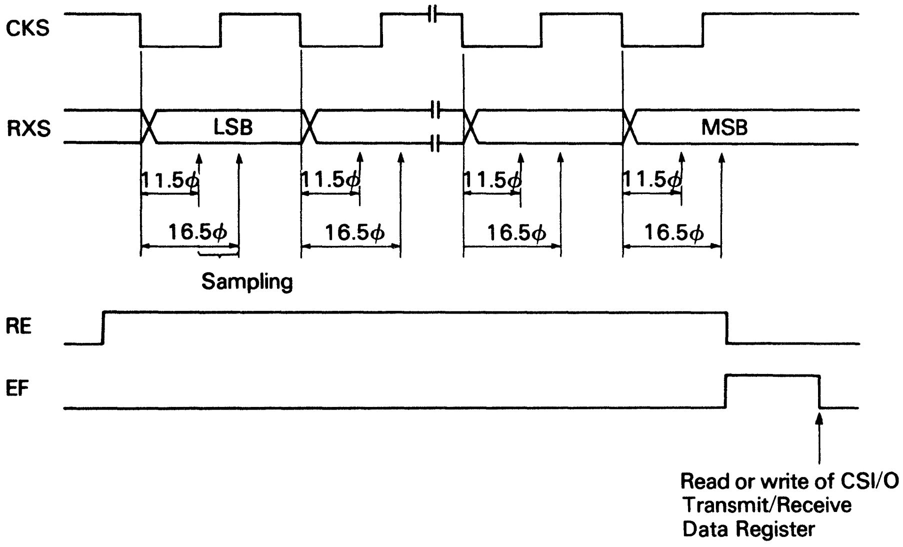


Figure 56 Receive Timing — External Clock

12.7 CSI/O and RESET

During RESET each bit in the CNTR is initialized as defined in the CNTR register description.

CSI/O transmit and receive operations in progress are aborted during RESET. However, the contents of TRDR are not changed.

13 PROGRAMMABLE RELOAD TIMER (PRT)

The HD64180 contains a two channel 16-bit Programmable Reload Timer. Each PRT channel contains a 16-bit down counter and a 16-bit reload register. The down counter can be directly read and written and a down counter overflow interrupt can be programmably enabled or disabled. In addition, PRT channel 1 has a TOUT output pin (multiplexed with A_{18}) which can be set HIGH, LOW, or toggled. Thus PRT1 can perform programmable output waveform generation.

13.1 PRT Block Diagram

The PRT block diagram is shown in Fig. 57. The two channels have separate timer data and reload registers and a common status/control register. The PRT input clock for both channels is equal to the system clock (ϕ) divided by 20.

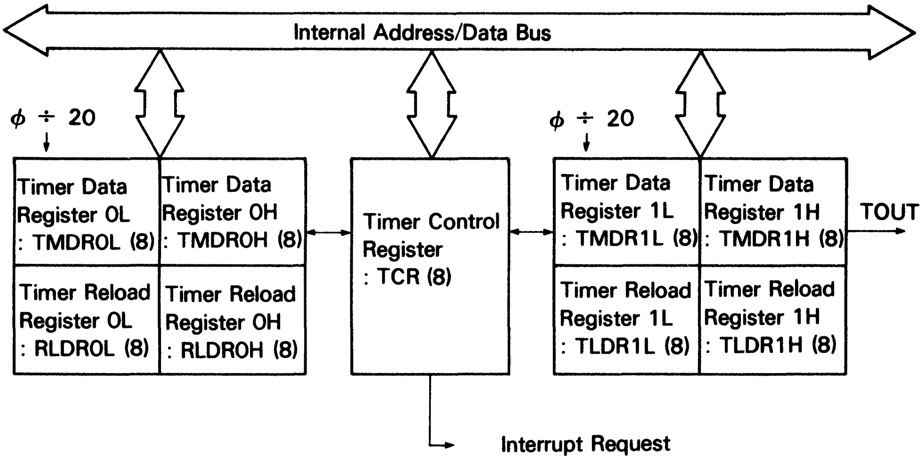

Internal Address/Data Bus
$\phi \div 20$
Timer Data Register 0L : TMDR0L (8)
Timer Data Register 0H : TMDR0H (8)
Timer Reload Register 0L : RLDR0L (8)
Timer Reload Register 0H : RLDR0H (8)
Timer Control Register : TCR (8)
$\phi \div 20$
Timer Data Register 1L : TMDR1L (8)
Timer Data Register 1H : TMDR1H (8)
Timer Reload Register 1L : TLDR1L (8)
Timer Reload Register 1H : TLDR1H (8)
TOUT
Interrupt Request

Figure 57 PRT Block Diagram

13.2 PRT Register Description

(1) Timer Data Register (TMDR: I/O Address = CH0: 0DH, 0CH CH1: 15H, 14H)

PRT0 and PRT1 each have 16-bit Timer Data Registers (TMDR). TMDR0 and TMDR1 are each accessed as low and high byte registers (TMDR0H, TMDR0L and TMDR1H, TMDR1L). During RESET, TMDR0 and TMDR1 are set to FFFFH.

TMDR is decremented once every twenty ϕ clocks. When TMDR counts down to 0, it is automatically reloaded with the value contained in the Reload Register (RLDR).

TMDR can be read and written by software using the following procedures. The read procedure uses a PRT internal temporary storage register to return accurate data without requiring the timer to be stopped. The write procedure requires the PRT to be stopped.

For reading (without stopping the timer), TMDR must be read in the order of lower byte — higher byte (TMDRnL, TMDRnH). The lower byte read (TMDRnL) will store the higher byte value in an internal register. The following higher byte read (TMDRnH) will access this internal register. This procedure insures timer data validity by eliminating the problem of potential 16-bit timer updating between each 8-bit read. Specifically, reading TMDR in higher byte — lower byte order may result in invalid data. Note the implications of TMDR higher byte internal storage for applications which may read only the lower and/or higher bytes. In normal operation all TMDR read routines should access both the lower and higher bytes, in that order.

For writing, the TMDR down counting must be inhibited using the TDE (Timer Down Count Enable) bits in the TCR (Timer Control Register), following which any or both higher and lower bytes of TMDR can be freely written (and read) in any order.

(2) Timer Reload Register (RLDR: I/O Address = CH0: 0EH, 0FH CH1: 16H, 17H)

PRT0 and PRT1 each have 16-bit Timer Reload Registers (RLDR). RLDR0 and RLDR1 are each accessed as low and high byte registers (RLDR0H, RLDR0L and RLDR1H, RLDR1L). During RESET RLDR0 and RLDR1 are set to FFFFH.

When the TMDR counts down to 0, it is automatically reloaded with the contents of RLDR.

(3) Timer Control Register (TCR)

TCR monitors both channels (PRT0, PRT1) TMDR status and controls enabling and disabling of down counting and interrupts as well as controlling the output pin (A_{18}/TOUT) for PRT 1.

Timer Control Register (TCR : I/O Address = 10H)

bit	7	6	5	4	3	2	1	0
	TIF1	TIF0	TIE1	TIE0	TOC1	TOC0	TDE1	TDE0
	R	R	R/W	R/W	R/W	R/W	R/W	R/W

TIF1: Timer Interrupt Flag 1 (bit 7)

When TMDR1 decrements to 0, TIF1 is set to 1. This can generate an interrupt request if enabled by TIE1 = 1. TIF1 is reset to 0 when TCR is read and the higher or lower byte of TMDR1 are read. During RESET, TIF1 is cleared to 0.

TIF0: Timer Interrupt Flag 0 (bit 6)

When TMDR0 decrements to 0, TIF0 is set to 1. This can generate an interrupt request if enabled by TIE0 = 1. TIF0 is reset to 0 when TCR is read and the higher or lower byte of TMDR0 are read. During RESET, TIF0 is cleared to 0.

TIE1: Timer Interrupt Enable 1 (bit 5)

When TIE1 is set to 1, TIF1 = 1 will generate a CPU interrupt request When TIE1 is reset to 0, the interrupt request is inhibited During RESET, TIE1 is cleared to 0.

TIE0: Timer Interrupt Enable 0 (bit 4)

When TIE0 is set to 1, TIF0 = 1 will generate a CPU interrupt request When TIE0 is reset to 0, the interrupt request is inhibited. During RESET, TIE0 is cleared to 0

TOC1, 0: Timer Output Control (bits 3, 2)

TOC1 and TOC0 control the output of PRT1 using the multiplexed A_{18}/TOUT pin as shown below. During RESET, TOC1 and TOC0 are cleared to 0. This selects the address function for A_{18}/ TOUT By programming TOC1 and TOC0, the A_{18}/TOUT pin can be forced HIGH, LOW or toggled when TMDR1 decrements to 0

TOC1	TOC0	OUTPUT	
0	0	Inhibited	(A_{18}/TOUT pin is selected as an address output function.)
0	1	toggled*	(A_{18}/TOUT pin is selected as a PRT1 output function)
1	0	0	
1	1	1	

- When TMDR1 decrements to 0, TOUT level is reversed This can provide square wave with 50% duty to external devices without any software support

TDE1, 0: Timer Down Count Enable (bits 1, 0)

TDE1 and TDE0 enable and disable down counting for TMDR1 and TMDR0 respectively. When TDEn (n = 0, 1) is set to 1, down counting is executed for TMDRn. When TDEn is reset to 0, down counting is stopped and TMDRn can be freely read or written. TDE1 and TDE0 are cleared to 0 during RESET and TMDRn will not decrement until TDEn is set to 1.

Fig. 58 shows timer initialization, count down and reload timing. Fig. 59 shows timer output (A_{18}/TOUT) timing.

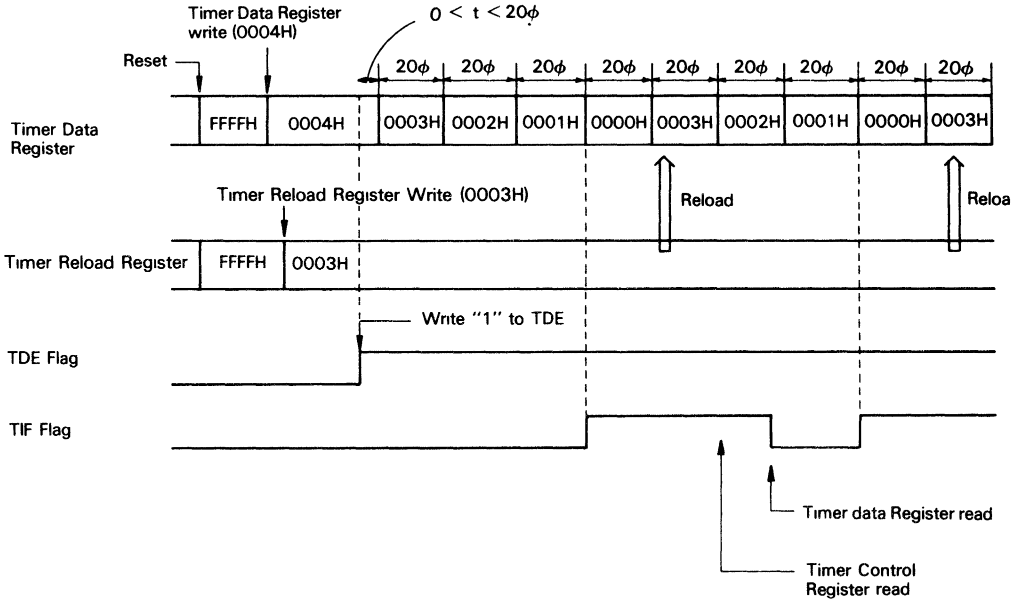


Figure 58 PRT Operation Timing

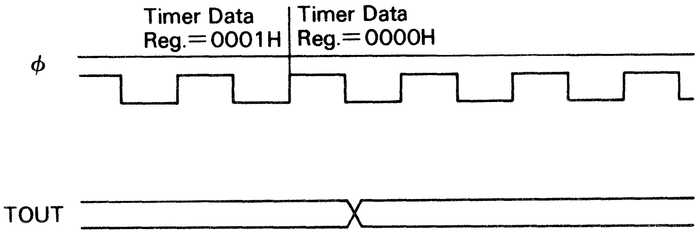


Figure 59 PRT Output Timing

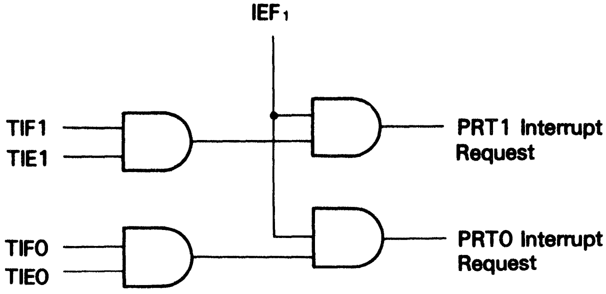


Figure 60 PRT Interrupt Request Circuit Diagram

13.3 PRT Interrupts

The PRT interrupt request circuit is shown in Fig. 60.

13.4 PRT and RESET

During RESET the bits in TCR are initialized as defined in the TCR register description. Down counting is stopped and the TMDR and RLDR registers are initialized to FFFFH. The A_{18}/TOUT pin reverts to the address output function.

13.5 PRT Operation Notes

(1) TMDR data can be accurately read without stopping down counting by reading the lower (TMDRnL*) and higher (TMDRnH*) bytes in that order. Or, TMDR can be freely read or written by stopping the down counting.

(2) Care should be taken to insure that a timer reload does not occur during or between lower (RLDRnL*) and higher (RLDRnH*) byte writes. This may be guaranteed by system design/timing or by stopping down counting (with TMDR containing a non-zero value) during the RLDR updating.
Similarly, in applications in which TMDR is written at each TMDR overflow, the system/software design should guarantee that RLDR can be updated before the next overflow occurs. Otherwise, time base inaccuracy will occur.
NOTE: * n = 0, 1

(3) During RESET, the multiplexed A_{18}/TOUT pin reverts to the address output.
By reprogramming the TOC1 and TOC0 bits, the timer output function for PRT channel 1 can be selected. The following shows the initial state of the TOUT pin after TOC1 and TOC0 are programmed to select the PRT channel 1 timer output function.

(i) PRT (channel 1) has not counted down to 0.

If the PRT has not counted down to 0 (timed out), the initial state of TOUT depends on the programmed value in TOC1 and TOC0.

TOC1	TOCO	TOUT State After Programming TOC1/TOCO	TOUT State After Next Timeout
0	1	HIGH (1)	LOW (0)
1	0	HIGH (1)	LOW (0)
1	1	HIGH (1)	HIGH (1)

(ii)PRT (channel 1) has counted down to 0 at least once.

If the PRT has counted down to 0 (timed out) at least once, the initial state of TOUT depends on the number of time outs (even or odd) that have occurred.

Numbers of Timeouts (even or odd)	TOUT State After Programming TOC1/TOCO
Even (2, 4, 6 ...)	HIGH (1)
Odd (1, 3, 5 ...)	LOW (0)

HITACHI

14 INTERNAL I/O REGISTERS

The HD64180 internal I/O Registers occupy 64 I/O addresses (including reserved addresses). These registers access the internal I/O modules (ASCI, CSI/O, PRT) and control functions (DMAC, DRAM refresh, interrupts, wait state generator, MMU and I/O relocation).

To avoid address conflicts with external I/O, the HD64180 internal I/O addresses can be relocated on 64 bytes boundaries within the bottom 256 bytes of the 64k bytes I/O address space.

14.1 I/O Control Register (ICR)

ICR allows relocating of the internal I/O addresses. ICR also controls enabling/disabling of the IOSTOP mode.

I/O Control Register (ICR I/O Address = 3FH)

bit	7	6	5	4	3	2	1	0
	IOA7	IOA6	IOSTP	–	–	–	–	–
	R/W	R/W	R/W					

IOA7,6: I/O Address Relocation (bits 7-6)

IOA7 and IOA6 relocate internal I/O as shown in Fig. 61. Note that the high-order 8 bits of 16-bit internal I/O addresses are always 0. IOA7 and IOA6 are cleared to 0 during RESET.

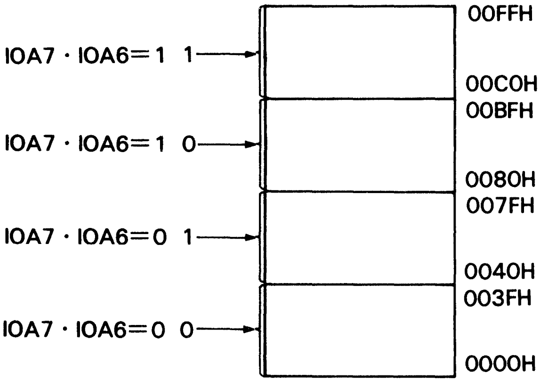


Figure 61 Internal I/O Address Relocation

IOSTP: IOSTOP Mode (bit 5)

IOSTOP mode is enabled when IOSTP is set to 1. Normal I/O operation resumes when IOSTP is reset to 0. IOSTP is cleared to 0 during RESET.

14.2 Internal I/O Registers Address Map

The internal I/O register addresses are shown in Table 14. These addresses are relative to the 64 bytes boundary base address specified in ICR.

Table 14 Internal I/O Register Address Map (1)

	Register	Mnemonic	Address	
			Binary	Hexadecimal
ASCI	ASCI Control Register A Ch 0	CNTLA0	XX000000	00H
	ASCI Control Register A Ch 1	CNTLA1	XX000001	01H
	ASCI Control Register B Ch 0	CNTLB0	XX000010	02H
	ASCI Control Register B Ch 1	CNTLB1	XX000011	03H
	ASCI Status Register Ch 0	STAT0	XX000100	04H
	ASCI Status Register Ch 1	STAT1	XX000101	05H
	ASCI Transmit Data Register Ch 0	TDR0	XX000110	06H
	ASCI Transmit Data Register Ch 1	TDR1	XX000111	07H
	ASCI Receive Data Register Ch 0	RDR0	XX001000	08H
	ASCI Receive Data Register Ch 1	RDR1	XX001001	09H
CSI/O	CSI/O Control Register	CNTR	XX001010	0AH
	CSI/O Transmit/Receive Data Register	TRDR	XX001011	0BH
Timer	Timer Data Register Ch 0L	TMDR0L	XX001100	0CH
	Timer Data Register Ch 0H	TMDR0H	XX001101	0DH
	Reload Register Ch 0L	RLDR0L	XX001110	0EH
	Reload Register Ch 0H	RLDR0H	XX001111	0FH
	Timer Control Register	TCR	XX010000	10H
	Reserved		XX010001	11H
			≀	≀
			XX010011	13H
	Timer Data Register Ch 1L	TMDR1L	XX010100	14H
	Timer Data Register Ch 1H	TMDR1H	XX010101	15H
	Reload Register Ch 1L	RLDR1L	XX010110	16H
	Reload Register Ch 1H	RLDR1H	XX010111	17H
Others	Free Running Counter	FRC	XX011000	18H
	Reserved		XX011001	19H
			≀	≀
			XX011111	1FH

Table 14 Internal I/O Register Address Map (2)

	Register	Mnemonic	Address	
			Binary	Hexadecimal
DMA	DMA Source Address Register Ch 0L	SAR0L	XX100000	20H
	DMA Source Address Register Ch 0H	SAR0H	XX100001	21H
	DMA Source Address Register Ch 0B	SAR0B	XX100010	22H
	DMA Destination Address Register Ch 0L	DAR0L	XX100011	23H
	DMA Destination Address Register Ch 0H	DAR0H	XX100100	24H
	DMA Destination Address Register Ch 0B	DAR0B	XX100101	25H
	DMA Byte Count Register Ch 0L	BCR0L	XX100110	26H
	DMA Byte Count Register Ch 0H	BCR0H	XX100111	27H
	DMA Memory Address Register Ch 1L	MAR1L	XX101000	28H
	DMA Memory Address Register Ch 1H	MAR1H	XX101001	29H
	DMA Memory Address Register Ch 1B	MAR1B	XX101010	2AH
	DMA I/O Address Register Ch 1L	IAR1L	XX101011	2BH
	DMA I/O Address Register Ch 1H	IAR1H	XX101100	2CH
	Reserved		XX101101	2DH
	DMA Byte Count Register Ch 1L	BCR1L	XX101110	2EH
	DMA Byte Count Register Ch 1H	BCR1H	XX101111	2FH
	DMA Status Register	DSTAT	XX110000	30H
	DMA Mode Register	DMODE	XX110001	31H
	DMA/WAIT Control Register	DCNTL	XX110010	32H
INT	IL Register (Interrupt Vector Low Register)	IL	XX110011	33H
	INT/TRAP Control Register	ITC	XX110100	34H
	Reserved		XX110101	35H
Refresh	Refresh Control Register	RCR	XX110110	36H
	Reserved		XX110111	37H
MMU	MMU Common Base Register	CBR	XX111000	38H
	MMU Bank Base Register	BBR	XX111001	39H
	MMU Common/Bank Area Register	CBAR	XX111010	3AH
I/O	Reserved		XX111011 ∫ XX111110	3BH ∫ 3EH
	I/O Control Register	ICR	XX111111	3FH

14.3 I/O Addressing Notes

The internal I/O register addresses are located in the I/O address space from 0000H to 00FFH (16-bit I/O addresses). Thus, to access the internal I/O registers (using I/O instructions), the high-order 8 bits of the 16-bit I/O address must be 0.

The conventional I/O instructions (OUT (m),A/ IN A,(m) / OUTI / INI/ etc.) place the contents of a CPU register on the high-order 8 bits of the address bus, and thus may be difficult to use for accessing internal I/O registers.

For efficient internal I/O register access, a number of new instructions have been added, which force the high-order 8 bits of the 16-bit I/O address to 0. These instructions are IN0, OUT0, OTIM, OTIMR, OTDM, OTDMR and TSTIO (See section 19 Instruction Set).

Note that when writing to an internal I/O register, the same I/O write occurs on the external bus. However, the duplicate external I/O write cycle will exhibit internal I/O write cycle timing. For example, the $\overline{\text{WAIT}}$ input and programmable wait state generator are ignored. Similarly, internal I/O read cycles also cause a duplicate external I/O read cycle — however, the external read data is ignored by the HD64180.

Normally, external I/O addresses should be chosen to avoid overlap with internal I/O addresses to avoid duplicate I/O accesses.

15 E CLOCK OUTPUT TIMING – 6800 TYPE BUS INTERFACE

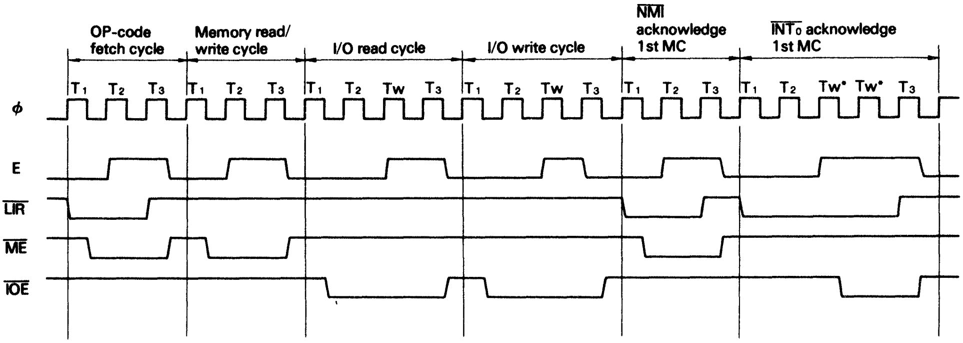
A large selection of 6800 type peripheral devices can be connected to the HD64180, including the Hitachi 6300 CMOS series (6321 PIA, 6350 ACIA, etc.) as well as 6500 family devices.

These devices require connection with the HD64180 synchronous E clock output. The speed (access time) required for the peripheral device are determined by the HD64180 clock rate. Table 15, Fig. 62 and Fig. 63 define E clock output timing.

Table 15 E Clock Timing in Each Condition

Condition	Duration of E Clock Output "High"	
Op-code Fetch Cycle Memory Read/Write Cycle	$T_2\uparrow - T_3\downarrow$	$(1.5\phi + n_w \cdot \phi)$
I/O read Cycle	1st Tw↑ – $T_3\downarrow$	$(0.5\phi + n_w \cdot \phi)$
I/O Write Cycle	1st Tw↑ – $T_3\uparrow$	$(n_w \cdot \phi)$
$\overline{\text{NMI}}$ Acknowledge 1st MC	$T_2\uparrow - T_3\downarrow$	(1.5ϕ)
$\overline{\text{INT}_0}$ Acknowledge 1st MC	1st Tw↑ – $T_3\downarrow$	$(0.5\phi + n_w \cdot \phi)$
BUS RELEASE mode SLEEP mode SYSTEM STOP mode	$\phi\downarrow - \phi\downarrow$	$(2\phi$ or $1\phi)$

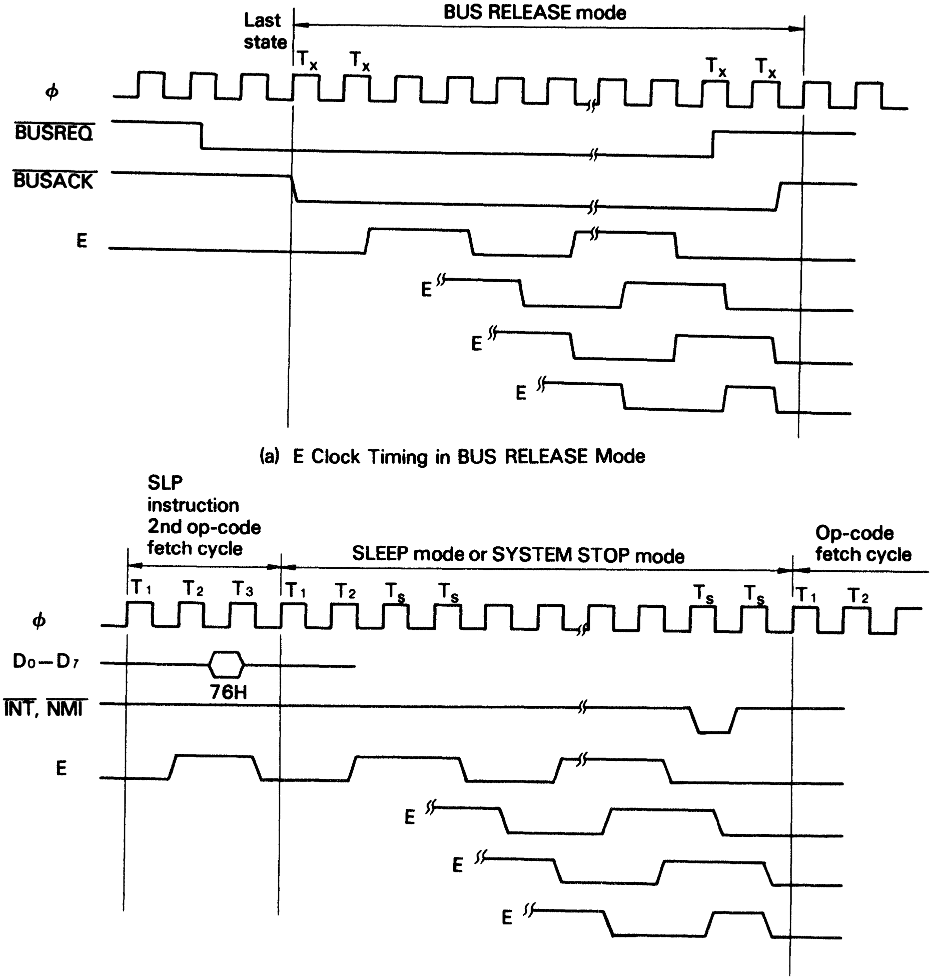
NOTE) n_w . the number of wait states
MC Machine Cycle



* Two wait states are automatically inserted.

NOTE) MC: Machine Cycle

Figure 62 E Clock Timing (During Read/Write Cycle and Interrupt Acknowledge Cycle)



(a) E Clock Timing in BUS RELEASE Mode

(b) E Clock Timing in SLEEP Mode and SYSTEM STOP Mode

Figure 63 E Clock Timing
(in BUS RELEASE mode, SLEEP mode, SYSTEM STOP mode)

Wait states inserted in op-code fetch, memory read/write and I/O read/write cycles extend the duration of E clock output HIGH. Note that during I/O read/write cycles with no wait states (only occurs during on-chip I/O register accesses), E will not go HIGH.

The correspondence between the duration of E clock output HIGH and standard peripheral device speed selections is as follows.

Device Speed Selection	Required duration of E clock output HIGH
1.0 MHz (ex: HD6321P)	500 ns min
1.5 MHz (ex: HD63A21P)	333 ns min.
2.0 MHz (ex: HD63B21P)	230 ns min.

15.1 6800 Type Bus Interfacing Note

When the HD64180 is connected to 6800 type peripheral LSIs with E clock, the 6800 type peripheral LSIs should be located in I/O address space.

If the 6800 type peripheral LSIs are located in memory address space, WR set-up time and WR hold time for E clock won't be guaranteed during memory read/write cycles and 6800 type peripheral LSIs can't be connected correctly.

16 ON-CHIP CLOCK GENERATOR

The HD64180 contains a crystal oscillator and system clock (ϕ) generator. A crystal can be directly connected or an external clock input can be provided. In either case, the system clock (ϕ) is equal to one-half the input clock. For example, a crystal or external clock input of 8 MHz corresponds with a system clock rate of ϕ = 4 MHz.

The following table shows the AT cut crystal characteristics (Co, Rs) and the load capacitance (CL1, CL2) required for various frequencies of HD64180 operation.

Table 16 Crystal Characteristics

Clock Frequency / Item	4MHz	4MHz < f ≦ 12MHz	12MHz < f ≦ 16MHz
Co	<7 pF	<7 pF	<7 pF
Rs	<60Ω	<60Ω	<60Ω
CL_1, CL_2	10 to 22 pF ±10%	10 to 22 pF ±10%	10 to 22 pF ±10%

If an external clock input is used instead of a crystal, the waveform (twice the ϕ clock rate) should exhibit a 50% ± 5% duty cycle. Note that the minimum clock input HIGH voltage level is $V_{CC}-0.6V$. The external clock input is connected to the EXTAL pin, while the XTAL pin is left open. Fig. 64 shows external clock interface.

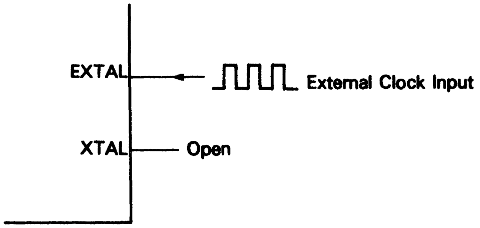


Figure 64 External Clock Interface

Fig. 65 shows the HD64180 clock generator circuit while Fig. 66 and Fig. 67 specify circuit board design rules.

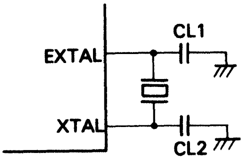


Figure 65 Crystal Interface

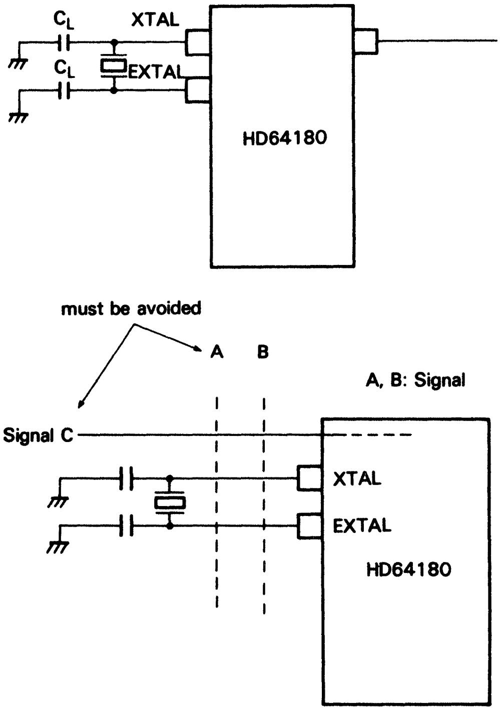


Figure 66 Note for Board Design of the Oscillation Circuit

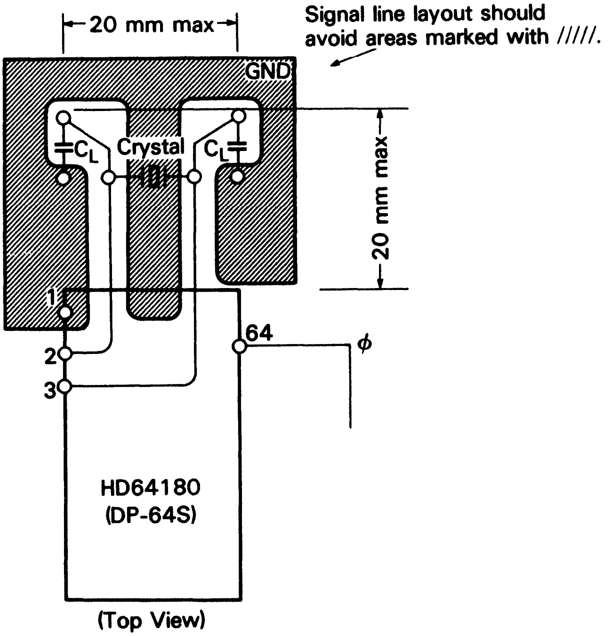


Figure 67 Example of Board Design

Circuit Board design should observe the followings.

(1) To prevent induced noise, the crystal and load capacitors should be physically located as close to the LSI as possible.

(2) Signal lines should not run parallel to the clock oscillator inputs. In particular, the clock input circuitry and the system clock ϕ output should be separated as much as possible.

(3) Similar to (2), V_{CC} power lines should be separated from the clock oscillator input circuitry.

(4) Resistivity between XTAL or EXTAL and the other pins should be greater than 10M ohms.
Signal line layout should avoid areas marked with /////.

17 MISCELLANEOUS

Free Running Counter (I/O Address = 18H)

Read only 8-bit free running counter without control registers and status registers. The contents of the 8-bit free running counter is counted down by 1 with an interval of 10 ϕ clock cycles. The free running counter continues counting down without being affected by the read operation.

If data is written into the free running counter, we can't guarantee the interval of DRAM refresh cycle and baud rates of ASCI and CSI/O.

In IOSTOP mode, the free running counter continues counting down. It is initialized to FFH during RESET.

18 OPERATION NOTES

18.1 Precaution on Interfacing the Z80* Family Peripheral LSIs to the HD64180

(1) Problem

In daisy chain, the Z80* family peripheral LSI (PIO, DMA, CTC, SIO, or DART) resets interrupt circuit (i.e. IEO changes from LOW to HIGH) by fetching the RETI op-code on the data bus concurrently during the CPU fetches the RETI. Therefore, the followings should be noted for the RETI op-code (EDH, 4DH) fetch timing in the Z80* peripheral LSI.

When the peripheral LSI fetches the first op-code of RETI (EDH), $\overline{\text{LIR}}$ should be negated HIGH at the rising edge of system clock ϕ as shown in Fig. 71, A. (This isn't referred in the manuals for the Z80* peripheral LSI.) So, $\overline{\text{LIR}}$ hold time ($\overline{\text{LIR}}$ = HIGH) should be required as shown in Fig. 71.

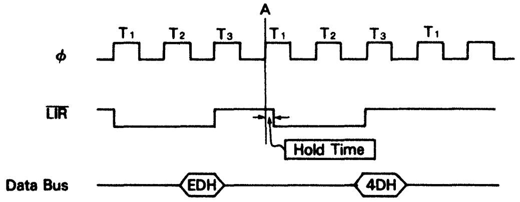


Figure 71 $\overline{\text{LIR}}$ Hold Time

Because $\overline{\text{LIR}}$ changes synchronously with the rising edge of system clock ϕ, $\overline{\text{LIR}}$ delay time is equal to $\overline{\text{LIR}}$ hold time of the Z80* peripheral LSI. However, this $\overline{\text{LIR}}$ hold time may not be sufficient for the Z80* peripheral LSI in some case and IEO line may not be reset.

(2) An example of countermeasure

Fig. 72 shows an example of circuit, while Fig. 73 shows the $\overline{\text{LIR}}$ and $\overline{\text{LIR}}$' timing in the circuit.

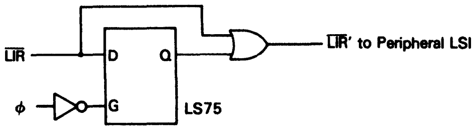


Figure 72 Circuit Example

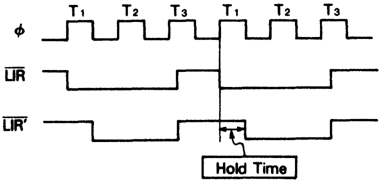


Figure 73 $\overline{\text{LIR}}$ and $\overline{\text{LIR}}$' Timing in the Circuit

$\overline{\text{LIR}}$', which is synchronized with the falling edge of system clock ϕ, is provided to the peripheral LSI. In this case, one-half clock cycle duration is confirmed as the hold time.

Please carefully examine the circuit before you use it on your application.

* Z80 is a registered trademark of Zilog, Inc.

18.5 Precaution on Interfacing HD64180 with Z80* CTC

(1) Problem

The following problem may happen when interfacing HD64180 with Z80* CTC (Z8430). Therefore, countermeasure shown in section 2 should be taken. Fig. 81 illustrates Z80* CTC write timing specified in Z80* CTC Data Sheet. Fig. 82 and Fig. 83 show Z80* I/O write timing and HD64180 I/O write timing respectively.

As shown above, $\overline{IOE}$ in HD64180 goes LOW by a half ϕ clock cycle faster than $\overline{IORQ}$ in Z80. When interfacing Z80 with Z80* CTC, data is written into Z80* CTC at the rising edge of Tw. By contrast, when interfacing HD64180 with Z80* CTC, data is written into Z80* CTC at the rising edge of T_2. In the latter case, data may not be written into Z80* CTC if $\overline{IOE}$ set-up time for the rising edge of T_2 is less than the set-up time specified in Z80* CTC.

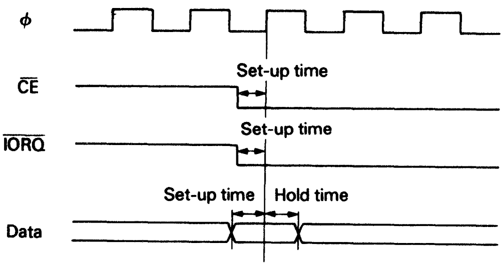


Figure 81 Z80* CTC Write Timing**

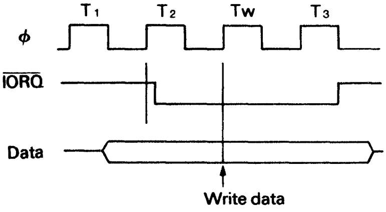


Figure 82 Z80* I/O Write Timing

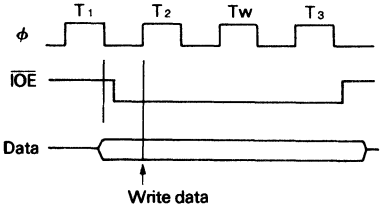
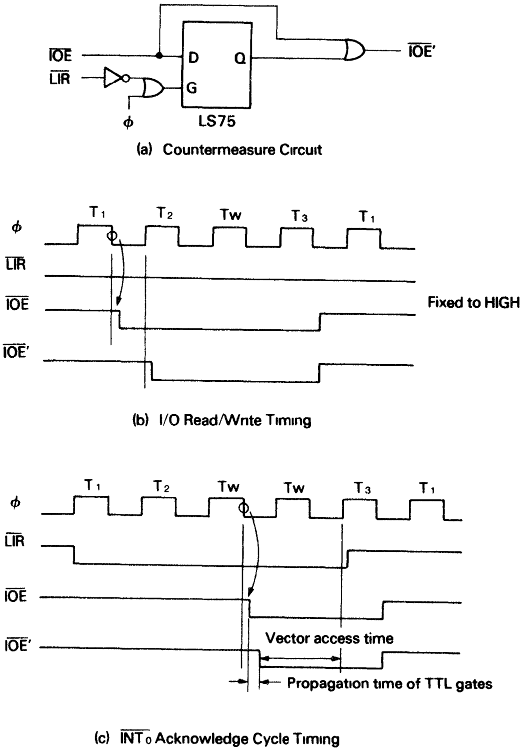


Figure 83 HD64180 I/O Write Timing

(2) Countermeasure

To Avoid the problem, $\overline{IOE}$ in HD64180 should be asserted LOW at the rising edge of T_2 to assure the set-up time specified in Z80* CTC. Fig. 84 (a) shows a circuit for delaying $\overline{IOE}$ by a half ϕ clock cycle.

If this circuit is externally connected between HD64180 and Z80* CTC, $\overline{IOE}$' will be pulled LOW at the rising edge of T_2 only in I/O read/write cycle as shown in Fig. 84 (b). While in $\overline{INT_0}$ acknowledge cycle, $\overline{IOE}$ and $\overline{IOE}$' are asserted LOW at the timing shown in Fig. 84 (c). In INT_0 acknowledge cycle, $\overline{IOE}$' delays because of propagation time of TTL gates of the countermeasure circuit and the vector access time is shortened. If vector access time for HD64180 is not assured during $\overline{INT_0}$ acknowledge cycle, wait states should be inserted by programming IWI0 and IWI1 bits of DMA/WAIT Control Register. However, note that wait states insertion by software should be inhibited during Z80* CTC read/write cycles, because more than one wait state can not be allowed in the case of Z80* CTC. (Please see Z80* CTC Data Sheet. One wait state is automatically inserted during the cycles.) Refer to "Fig. 85 Z80* CTC Access Flow" for details.



(a) Countermeasure Circuit

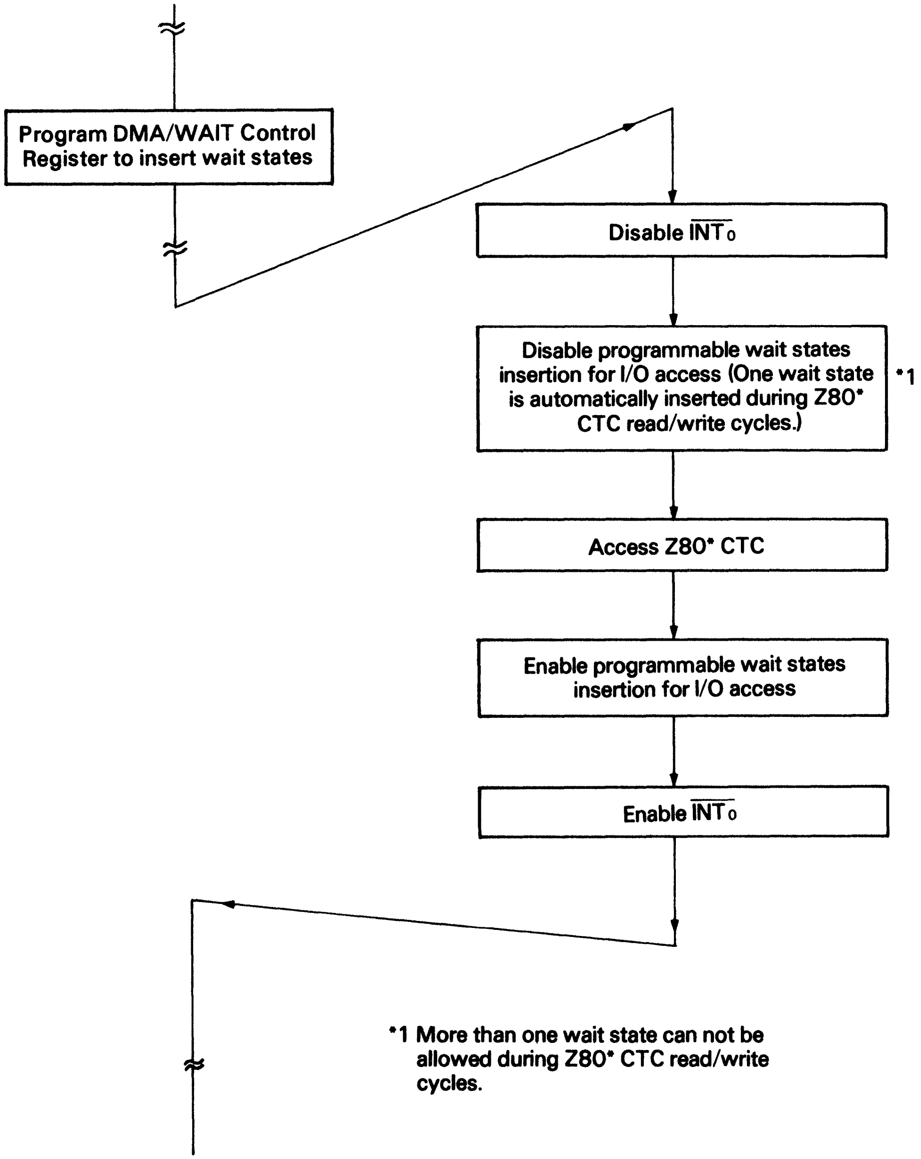
(b) I/O Read/Write Timing

(c) $\overline{INT_0}$ Acknowledge Cycle Timing

Figure 84 Countermeasure Circuit and Timings in the Circuit

* Z80 is a registered trademark of Zilog, Inc.

** Copied from Z80* CTC Data Sheet (April, 1985)


Program DMA/WAIT Control Register to insert wait states
Disable $\overline{INT}_0$
Disable programmable wait states insertion for I/O access (One wait state is automatically inserted during Z80* CTC read/write cycles.)
*1
Access Z80* CTC
Enable programmable wait states insertion for I/O access
Enable $\overline{INT}_0$

1 More than one wait state can not be allowed during Z80 CTC read/write cycles.

Figure 85 Z80* CTC Access Flow

18.6 Notes on HD64180 $\overline{\mathrm{INT}}_0$ Mode 0

(1) Problem

In $\overline{\mathrm{INT}}_0$ Mode 0, the CPU executes an instruction which is placed on the data bus during the interrupt acknowledge cycle. Usually, RST (1-byte instruction) or CALL (3-byte instruction) is placed on the data bus. Then, the CPU pushes the Program Counter (PC) onto the stack and jumps to the interrupt service routine. In the case of RST instruction, the correct return address is pushed onto the stack. However, in the case of CALL instruction, the pushed return address is equal to the correct return address + 2.

(2) Explanation of operation

During the 1st op-code fetch cycle in the interrupt acknowledge cycle, the CPU stops incrementing the PC. At this time, the PC contains the return address. After the 1st op-code is fetched, the CPU restarts incrementing the PC. Therefore, is RST (1-byte instruction) is executed in the interrupt acknowledge cycle, the correct return address is pushed onto the stack and the CPU can return from the interrupt service routine correctly. While, if CALL (3-byte instruction) is executed in the interrupt acknowledge cycle, the PC is incremented twice during the operand read cycle of the 2 bytes after the 1st op-code is fetched. Therefore, the return address + 2 in the PC is pushed onto the stack. So, when RETI is executed at the end of the interrupt service routine, the CPU can not return from the interrupt correctly.

Fig 86 shows the CALL execution timing in $\overline{\mathrm{INT}}_0$ Mode 0.

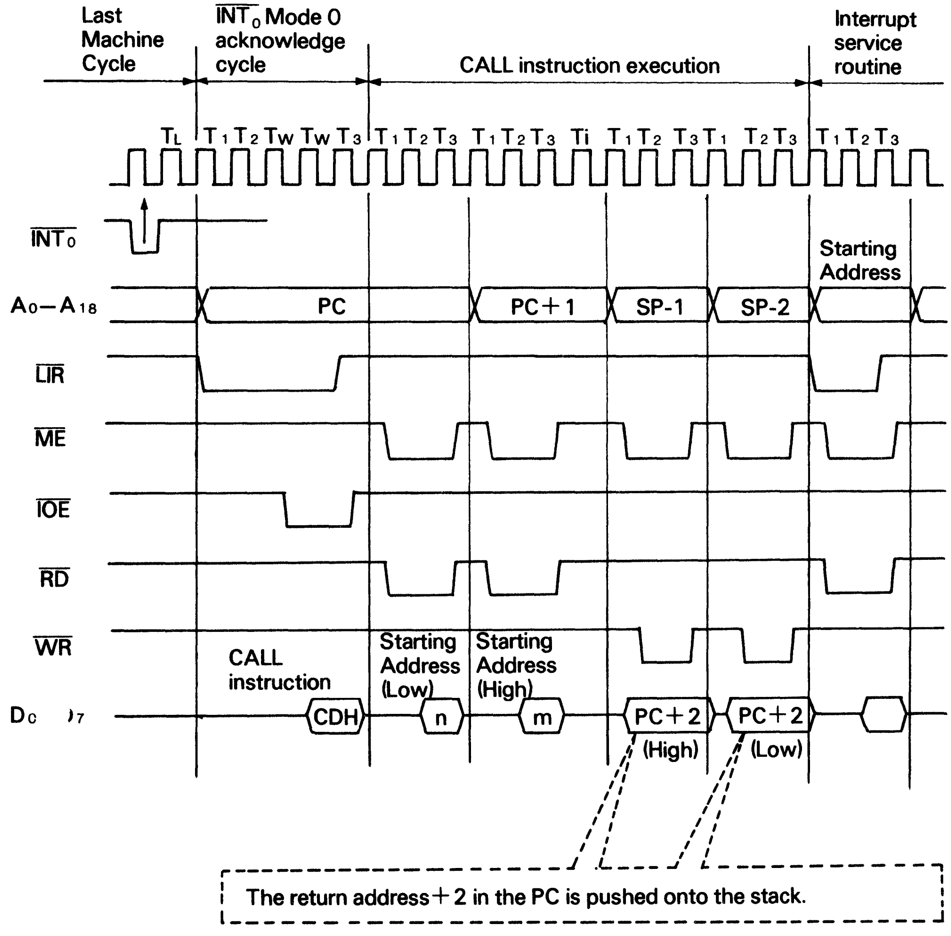

Last Machine Cycle
INT0 Mode 0 acknowledge cycle
CALL instruction execution
Interrupt service routine
TL T1 T2 Tw Tw T3 T1 T2 T3 T1 T2 T3 Ti T1 T2 T3 T1 T2 T3 T1 T2 T3
INT0
A0—A18
PC
PC+1
SP-1
SP-2
Starting Address
LIR
ME
IOE
RD
WR
D0 D7
CALL instruction
CDH
Starting Address (Low)
n
Starting Address (High)
m
PC+2 (High)
PC+2 (Low)
The return address+2 in the PC is pushed onto the stack.

Figure 86 The CALL Execution Timing in $\overline{\mathrm{INT}}_0$ Mode 0

(3) Countermeasure

The following explains the countermeasure of the problem in $\overline{INT_0}$ Mode 0.

① RST

When RST is executed, the correct return address in the PC is pushed onto the stack.

② CALL

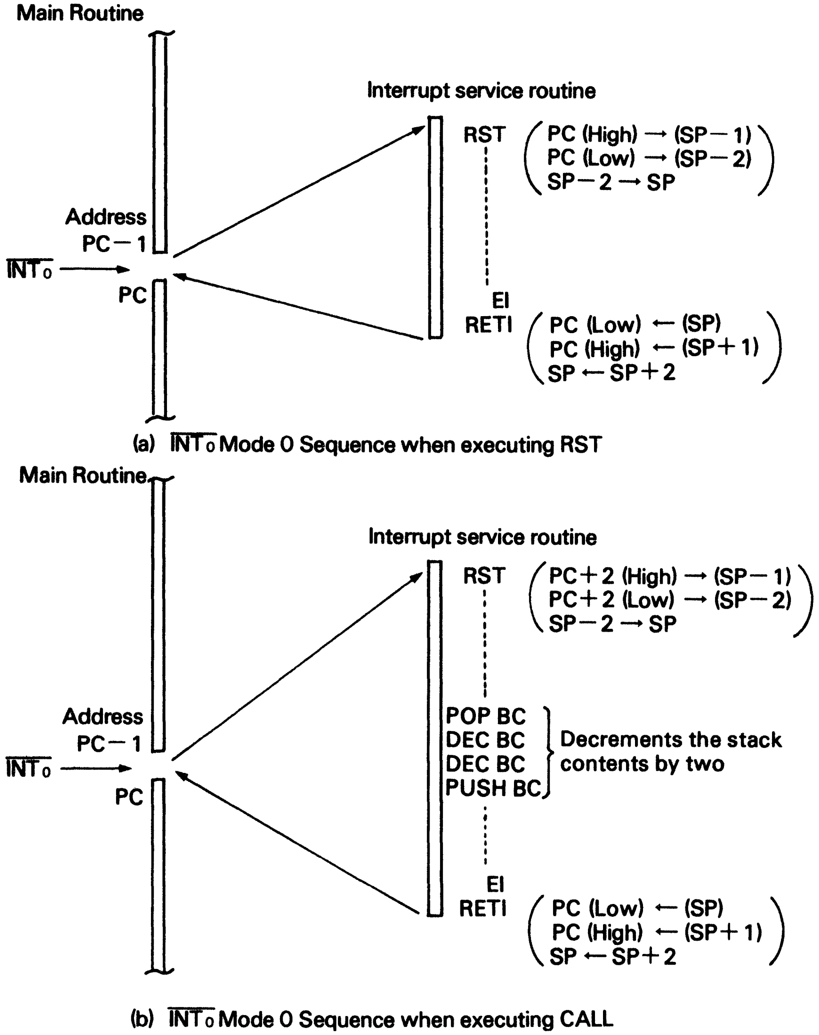
When CALL is executed, the stack contents must be decremented by two in the interrupt service routine to return from the interrupt correctly.

Table 18 summarizes how to adjust the stack contents depending on the instruction to be executed.

Table 18 Stack Contents Adjustment

Instruction	Stack Contents Adjustment
RST	No
CALL	Decrement the stack contents by two
Other instructions	No (The PC is not stacked.)

The $\overline{INT_0}$ Mode 0 sequences when executing RST and CALL are shown in Fig. 87.



(a) $\overline{INT_0}$ Mode 0 Sequence when executing RST

(b) $\overline{INT_0}$ Mode 0 Sequence when executing CALL

NOTE) PC: PC indicates the return address

Figure 87 $\overline{INT_0}$ Mode 0 Sequence

19 INSTRUCTION SET

19.1 Instruction set overview

The HD64180 is object code compatible with standard 8-bit operating system and application software. The instruction set also contains a number of new instructions to improve system and software performance, reliability and efficiency.

New Instructions	Operation
SLP	Enter SLEEP mode
MLT	8-bit multiply with 16-bit result
IN0 g, (m)	Input contents of immediate I/O address into register
OUT0 (m), g	Output register contents to immediate I/O address
OTIM	Block output — increment
OTIMR	Block output — increment and repeat
OTDM	Block output — decrement
OTDMR	Block output — decrement and repeat
TSTIO m	Non-destructive AND, I/O port and accumulator
TST g	Non-destructive AND, register and accumulator
TST m	Non-destructive AND, immediate data and accumulator
TST (HL)	Non-destructive AND, memory data and accumulator

(1) SLP — Sleep

The SLP instruction causes the HD64180 to enter SLEEP low power consumption mode. See section 5 for a complete description of the SLEEP state.

(2) MLT — Multiply

The MLT performs unsigned multiplication on two 8 bit numbers yielding a 16 bit result. MLT may specify BC, DE, HL or SP registers. In all cases, the 8-bit operands are loaded inio each half of the 16-bit register and the 16-bit result is returned in that register.

(3) IN0 g, (m) — Input, Immediate I/O address

The contents of immediately specified 8-bit I/O address are input into the specified register. When I/O is accessed, 00H is output in high-order bits of address automatically.

(4) OUT0 (m), g — Output, immediate I/O address

The contents of the specified register are output to the immediately specified 8-bit I/O address. When I/O is accessed, 00H is output in high-order bits of address automatically.

(5) OTIM, OTIMR, OTDM, OTDMR — Block I/O

The contents of memory pointed to by HL is output to the I/O address in (C). The memory address (HL) and I/O address (C) are incremented in OTIM and OTIMR and decremented in OTDM and OTDMR respectively. B register is decremented. The OTIMR and OTDMR variants repeat the above sequence until register B is decremented to 0. Since the I/O address (C) is automatically incremented or decremented, these instructions are useful for block I/O (such as HD64180 on-chip I/O) initialization. When I/O is accessed, 00H is output in high-order bits of address automatically.

(6) TSTIO m — Test I/O Port

The contents of the I/O port addressed by C are ANDed with 8-bit immediate data and the status flags are updated. The I/O port contents are not written (non-destructive AND). When I/O is accessed, 00H is output in higher bits of address automatically.

(7) TST g — Test Register

The contents of the specified register are ANDed with the accumulator (A) and the status flags are updated. The accumulator and specified register are not changed (non-destructive AND).

(8) TST m — Test Immediate

The 8-bit immediate data is ANDed with the accumulator (A) and the status flags are updated. The accumulator is not changed (non-destructive AND).

(9) TST (HL) — Test Memory

The contents of memory pointed to by HL are ANDed with the accumulator (A) and the status flags are updated. The memory contents and accumulator are not changed (non-destructive AND).

19.2 Instruction set summary

The followings explain the symbols in instruction set, and the following tables summarize the operation of each instruction.

(1) Register

g, g', ww, xx, yy, and zz specify a register to be used. g and g' specify an 8-bit register. ww, xx, yy, and zz specify a pair of 16-bit registers. The following tables show the correspondence between symbols and registers.

g,g'	Reg.
000	B
001	C
010	D
011	E
100	H
101	L
111	A

ww	Reg.
00	BC
01	DE
10	HL
11	SP

xx	Reg.
00	BC
01	DE
10	IX
11	SP

yy	Reg.
00	BC
01	DE
10	IY
11	SP

zz	Reg.
00	BC
01	DE
10	HL
11	AF

NOTE: Suffixed H and L to ww,xx,yy,zz (ex.wwH,IXL) indicate upper and lower 8-bit of the 16-bit register respectively.

(2) Bit

b specifies a bit to be manipulated in the bit manipulation instruction. The following table shows the correspondençe between b and bits.

b	Bit
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

(3) Condition

f specifies the condition in program control instructions. The following shows the correspondence between f and conditions.

f	Condition
000	NZ non zero
001	Z zero
010	NC non carry
011	C carry
100	PO parity odd
101	PE parity even
110	P sign plus
111	M sign minus

(4) Restart Address

v specifies a restart address. The following table shows the correspondence between v and restart addresses.

v	Address
000	00H
001	08H
010	10H
011	18H
100	20H
101	28H
110	30H
111	38H

(5) Flag

The following symbols show the flag conditions.

·	:	not affected
↑	:	affected
×	:	undefined
S	:	set to 1
R	:	reset to 0
P	:	parity
V	:	overflow

(6) Miscellaneous

$(\quad)_M$	:	data in the memory address
$(\quad)_I$	:	data in the I/O address
m or n	:	8-bit data
mn	:	16-bit data
r	:	8-bit register
R	:	16-bit register
$b\cdot(\quad)_M$	:	a content of bit b in the memory address
b·gr	:	a content of bit b in the register gr
d or j	:	8-bit signed displacement
S	:	source addressing mode
D	:	destination addressing mode
·	:	AND operation
+	:	OR operation
$\oplus$	:	EXCLUSIVE OR operation

Data Manipulation Instructions

Arithmetic and Logical Instructions (8-bit)

Operation name	MNEMONICS	OP-code	Addressing							Bytes	States	Operation	Flag					
													7	6	4	2	1	0
			IMMED	EXT	IND	REG	REGI	IMP	REL				S	Z	H	P/V	N	C
ADD	ADD A,g	10 000 g				S		D		1	4	$Ar+gr \rightarrow Ar$	↕	↕	↕	V	R	↕
	ADD A, (HL)	10 000 110					S	D		1	6	$Ar+(HL)_M \rightarrow Ar$	↕	↕	↕	V	R	↕
	ADD A,m	11 000 110	S					D		2	6	$Ar+m \rightarrow Ar$	↕	↕	↕	V	R	↕
		< m >																
	ADD A, (IX+d)	11 011 101			S			D		3	14	$Ar+(IX+d)_M \rightarrow Ar$	↕	↕	↕	V	R	↕
		10 000 110																
		< d >																
	ADD A, (IY+d)	11 111 101			S			D		3	14	$Ar+(IY+d)_M \rightarrow Ar$	↕	↕	↕	V	R	↕
		10 000 110																
		< d >																
ADC	ADC A,g	10 001 g				S		D		1	4	$Ar+gr+c \rightarrow Ar$	↕	↕	↕	V	R	↕
	ADC A, (HL)	10 001 110					S	D		1	6	$Ar+(HL)_M+c \rightarrow Ar$	↕	↕	↕	V	R	↕
	ADC A,m	11 001 110	S					D		2	6	$Ar+m+c \rightarrow Ar$	↕	↕	↕	V	R	↕
		< m >																
	ADC A, (IX+d)	11 011 101			S			D		3	14	$Ar+(IX+d)_M+c \rightarrow Ar$	↕	↕	↕	V	R	↕
		10 001 110																
		< d >																
	ADC A, (IY+d)	11 111 101			S			D		3	14	$Ar+(IY+d)_M+c \rightarrow Ar$	↕	↕	↕	V	R	↕
		10 001 110																
		< d >																
AND	AND g	10 100 g				S		D		1	4	$Ar \cdot gr \rightarrow Ar$	↕	↕	S	P	R	R
	AND (HL)	10 100 110					S	D		1	6	$Ar \cdot (HL)_M \rightarrow Ar$	↕	↕	S	P	R	R
	AND m	11 100 110	S					D		2	6	$Ar \quad m \rightarrow Ar$	↕	↕	S	P	R	R
		< m >																
	AND (IX+d)	11 011 101			S			D		3	14	$Ar \cdot (IX+d)_M \rightarrow Ar$	↕	↕	S	P	R	R
		10 100 110																
		< d >																
	AND (IY+d)	11 111 101			S			D		3	14	$Ar \quad (IY+d)_M \rightarrow Ar$	↕	↕	S	P	R	R
		10 100 110																
		< d >																
Compare	CP g	10 111 g				S		D		1	4	$Ar-gr$	↕	↕	↕	V	S	↕
	CP (HL)	10 111 110					S	D		1	6	$Ar-(HL)_M$	↕	↕	↕	V	S	↕
	CP m	11 111 110	S					D		2	6	$Ar-m$	↕	↕	↕	V	S	↕
		< m >																
	CP (IX+d)	11 011 101			S			D		3	14	$Ar-(IX+d)_M$	↕	↕	↕	V	S	↕
		10 111 110																
		< d >																
	CP (IY+d)	11 111 101			S			D		3	14	$Ar-(IY+d)_M$	↕	↕	↕	V	S	↕
		10 111 110																
		< d >																
COMPLEMENT	CPL	00 101 111						S/D		1	3	$\overline{Ar} \rightarrow Ar$		·	S	·	S	·
DEC	DEC g	00 g 101				S/D				1	4	$gr-1 \rightarrow gr$	↕	↕	↕	V	S	·
	DEC (HL)	00 110 101					S/D			1	10	$(HL)_M-1 \rightarrow (HL)_M$	↕	↕	↕	V	S	·
	DEC (IX+d)	11 011 101			S/D					3	18	$(IX+d)_M-1 \rightarrow$	↕	↕	↕	V	S	
		00 110 101										$(IX+d)_M$						
		< d >																
	DEC (IY+d)	11 111 101			S/D					3	18	$(IY+d)_M-1 \rightarrow$	↕	↕	↕	V	S	·
		00 110 101										$(IY+d)_M$						
		< d >																
INC	INC g	00 g 100				S/D				1	4	$gr+1 \rightarrow gr$	↕	↕	↕	V	R	·
	INC (HL)	00 110 100					S/D			1	10	$(HL)_M+1 \rightarrow (HL)_M$	↕	↕	↕	V	R	
	INC (IX+d)	11 011 101			S/D					3	18	$(IX+d)_M+1 \rightarrow$	↕	↕	↕	V	R	·
		00 110 100										$(IX+d)_M$						

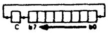
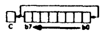
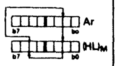
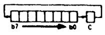
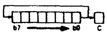
(to be continued)

3

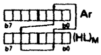
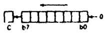
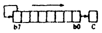
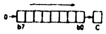
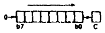
Operation name	MNEMONICS	OP-code	Addressing							Bytes	States	Operation	Flag					
													7	6	4	2	1	0
			IMMED	EXT	IND	REG	REGI	IMP	REL				S	Z	H	P/V	N	C
INC	INC (IY+d)	< d > 11 111 101 00 110 100 < d >			S/D					3	18	$(IY+d)_M+1 \rightarrow$ $(IY+d)_M$	↕	↕	↕	V	R	·
MULT	MLT ww	11 101 101 01 ww1 100				S/D				2	17	$wwHr \times wwLr \rightarrow ww_R$	·		·		·	·
NEGATE	NEG	11 101 101 01 000 100						S/D		2	6	$0-Ar \rightarrow Ar$	↕	↕	↕	V	S	↕
OR	OR g	10 110 g				S		D		1	4	$Ar+gr \rightarrow Ar$	↕	↕	R	P	R	R
	OR (HL)	10 110 110					S	D		1	6	$Ar+(HL)_M \rightarrow Ar$	↕	↕	R	P	R	R
	OR m	11 110 110 < m >	S					D		2	6	$Ar+m \rightarrow Ar$	↕	↕	R	P	R	R
	OR (IX+d)	11 011 101 10 110 110 < d >			S			D		3	14	$Ar+(IX+d)_M \rightarrow Ar$	↕	↕	R	P	R	R
	OR (IY+d)	11 111 101 10 110 110 < d >			S			D		3	14	$Ar+(IY+d)_M \rightarrow Ar$	↕	↕	R	P	R	R
SUB	SUB g	10 010 g				S		D		1	4	$Ar-gr \rightarrow Ar$	↕	↕	↕	V	S	↕
	SUB (HL)	10 010 110					S	D		1	6	$Ar-(HL)_M \rightarrow Ar$	↕	↕	↕	V	S	↕
	SUB m	11 010 110 < m >	S					D		2	6	$Ar-m \rightarrow Ar$	↕	↕	↕	V	S	↕
	SUB (IX+d)	11 011 101 10 010 110 < d >			S			D		3	14	$Ar-(IX+d)_M \rightarrow Ar$	↕	↕	↕	V	S	↕
	SUB (IY+d)	11 111 101 10 010 110 < d >			S			D		3	14	$Ar-(IY+d)_M \rightarrow Ar$	↕	↕	↕	V	S	↕
SUBC	SBC A,g	10 011 g				S		D		1	4	$Ar-gr-c \rightarrow Ar$	↕	↕	↕	V	S	↕
	SBC A, (HL)	10 011 110					S	D		1	6	$Ar-(HL)_M-c \rightarrow Ar$	↕	↕	↕	V	S	↕
	SBC A,m	11 011 110 < m >	S					D		2	6	$Ar-m-c \rightarrow Ar$	↕	↕	↕	V	S	↕
	SBC A, (IX+d)	11 011 101 10 011 110 < d >			S			D		3	14	$Ar-(IX+d)_M-c \rightarrow Ar$	↕	↕	↕	V	S	↕
	SBC A, (IY+d)	11 111 101 10 011 110 < d >			S			D		3	14	$Ar-(IY+d)_M-c \rightarrow Ar$	↕	↕	↕	V	S	↕
TEST	TST g	11 101 101 00 g 100				S				2	7	$Ar \cdot gr$	↕	↕	S	P	R	R
	TST (HL)	11 101 101 00 110 100					S			2	10	$Ar \cdot (HL)_M$	↕	↕	S	P	R	R
	TST m	11 101 101 01 100 100 < m >	S							3	9	Ar m	↕	↕	S	P	R	R
XOR	XOR g	10 101 g				S		D		1	4	$Ar \oplus gr \rightarrow Ar$	↕	↕	R	P	R	R
	XOR (HL)	10 101 110					S	D		1	6	$Ar \oplus (HL)_M \rightarrow Ar$	↕	↕	R	P	R	R
	XOR m	11 101 110 < m >	S					D		2	6	$Ar \oplus m \rightarrow Ar$	↕	↕	R	P	R	R
	XOR (IX+d)	11 011 101 10 101 110 < d >			S			D		3	14	$Ar \oplus (IX+d)_M \rightarrow Ar$	↕	↕	R	P	R	R
	XOR (IY+d)	11 111 101 10 101 110 < d >			S			D		3	14	$Ar \oplus (IY+d)_M \rightarrow Ar$	↕	↕	R	P	R	R

(to be continued)

Rotate and Shift Instructions

Operation name	MNEMONICS	OP-code	Addressing							Bytes	States	Operation	Flag					
													7	6	4	2	1	0
			IMMED	EXT	IND	REG	REGI	IMP	REL				S	Z	H	P/V	N	C
Rotate and Shift Data	RLA	00 010 111						S/D		1	3			·	R	·	R	↕
	RL g	11 001 011 00 010 g				S/D				2	7		↕	↕	R	P	R	↕
	RL (HL)	11 001 011 00 010 110					S/D			2	13		↕	↕	R	P	R	↕
	RL (IX+d)	11 011 101 11 001 011 < d > 00 010 110			S/D					4	19		↕	↕	R	P	R	↕
	RL (IY+d)	11 111 101 11 001 011 < d > 00 010 110			S/D					4	19		↕	↕	R	P	R	↕
	RLCA	00 000 111						S/D		1	3		·	·	R		R	↕
	RLC g	11 001 011 00 000 g				S/D				2	7		↕	↕	R	P	R	↕
	RLC (HL)	11 001 011 00 000 110					S/D			2	13		↕	↕	R	P	R	↕
	RLC (IX+d)	11 011 101 11 001 011 < d > 00 000 110			S/D					4	19		↕	↕	R	P	R	↕
	RLC (IY+d)	11 111 101 11 001 011 < d > 00 000 110			S/D					4	19		↕	↕	R	P	R	↕
	RLD	11 101 101 01 101 111						S/D		2	16		↕	↕	R	P	R	·
	RRA	00 011 111						S/D		1	3		·		R		R	↕
	RR g	11 001 011 00 011 g				S/D				2	7		↕	↕	R	P	R	↕
	RR (HL)	11 001 011 00 011 110					S/D			2	13		↕	↕	R	P	R	↕
	RR (IX+d)	11 011 101 11 001 011 < d > 00 011 110			S/D					4	19		↕	↕	R	P	R	↕
	RR (IY+d)	11 111 101 11 001 011 < d > 00 011 110			S/D					4	19		↕	↕	R	P	R	↕
	RRCA	00 001 111						S/D		1	3		·	·	R		R	↕
	RRC g	11 001 011 00 001 g				S/D				2	7		↕	↕	R	P	R	↕
	RRC (HL)	11 001 011 00 001 110					S/D			2	13		↕	↕	R	P	R	↕
	RRC (IX+d)	11 011 101 11 001 011 < d > 00 001 110			S/D					4	19		↕	↕	R	P	R	↕
	RRC (IY+d)	11 111 101 11 001 011 < d > 00 001 110			S/D					4	19		↕	↕	R	P	R	↕

(to be continued)

Operation name	MNEMONICS	OP-code	Addressing							Bytes	States	Operation	Flag					
													7	6	4	2	1	0
			IMMED	EXT	IND	REG	REGI	IMP	REL				S	Z	H	P/V	N	C
Rotate	RRD	11 101 101						S/D		2	16		↕	↕	R	P	R	
and		01 100 111																
Shift	SLA g	11 001 011				S/D				2	7		↕	↕	R	P	R	↕
Data		00 100 g																
	SLA (HL)	11 001 011					S/D			2	13		↕	↕	R	P	R	↕
		00 100 110																
	SLA (IX+d)	11 011 101			S/D					4	19		↕	↕	R	P	R	↕
		11 001 011																
		< d >																
		00 100 110																
	SLA (IY+d)	11 111 101			S/D					4	19		↕	↕	R	P	R	↕
		11 001 011																
		< d >																
		00 100 110																
	SRA g	11 001 011				S/D				2	7		↕	↕	R	P	R	↕
		00 101 g																
	SRA (HL)	11 001 011					S/D			2	13		↕	↕	R	P	R	↕
		00 101 110																
	SRA (IX+d)	11 011 101			S/D					4	19		↕	↕	R	P	R	↕
		11 001 011																
		< d >																
		00 101 110																
	SRA (IY+d)	11 111 101			S/D					4	19		↕	↕	R	P	R	↕
		11 001 011																
		< d >																
		00 101 110																
	SRL g	11 001 011				S/D				2	7		↕	↕	R	P	R	↕
		00 111 g																
	SRL (HL)	11 001 011					S/D			2	13		↕	↕	R	P	R	↕
		00 111 110																
	SRL (IX+d)	11 011 101			S/D					4	19		↕	↕	R	P	R	↕
		11 001 011																
		< d >																
		00 111 110																
	SRL (IY+d)	11 111 101			S/D					4	19		↕	↕	R	P	R	↕
		11 001 011																
		< d >																
		00 111 110																

Bit Manipulation Instructions

Operation name	MNEMONICS	OP-code	Addressing							Bytes	States	Operation	Flag					
													7	6	4	2	1	0
			IMMED	EXT	IND	REG	REGI	IMP	REL				S	Z	H	P/V	N	C
Bit Set	SET b,g	11 001 011 11 b g				S/D				2	7	$1 \to b \cdot gr$	·	·	·	·	·	·
	SET b, (HL)	11 001 011 11 b 110					S/D			2	13	$1 \to b \cdot (HL)_M$	·	·	·	·	·	·
	SET b, (IX+d)	11 011 101 11 001 011 < d > 11 b 110			S/D					4	19	$1 \to b \cdot (IX+d)_M$	·	·	·	·	·	·
	SET b, (IY+d)	11 111 101 11 001 011 < d > 11 b 110			S/D					4	19	$1 \to b \cdot (IY+d)_M$	·	·	·	·	·	·
Bit Reset	RES b,g	11 001 011 10 b g				S/D				2	7	$0 \to b \cdot gr$	·	·	·	·	·	·
	RES b, (HL)	11 001 011 10 b 110					S/D			2	13	$0 \to b \cdot (HL)_M$	·	·	·	·	·	·
	RES b, (IX+d)	11 011 101 11 001 011 < d > 10 b 110			S/D					4	19	$0 \to b \cdot (IX+d)_M$	·	·	·	·	·	·
	RES b, (IY+d)	11 111 101 11 001 011 < d > 10 b 110			S/D					4	19	$0 \to b \ (IY+d)_M$	·	·	·	·	·	·
Bit Test	BIT b,g	11 001 011 01 b g				S				2	6	$\overline{b \cdot gr} \to z$	X	↕	S	X	R	·
	BIT b, (HL)	11 001 011 01 b 110					S			2	9	$\overline{b \cdot (HL)_M} \to z$	X	↕	S	X	R	·
	BIT b, (IX+d)	11 011 101 11 001 011 < d > 01 b 110			S					4	15	$\overline{b \cdot (IX+d)_M} \to z$	X	↕	S	X	R	·
	BIT b, (IY+d)	11 111 101 11 001 011 < d > 01 b 110			S					4	15	$\overline{b \cdot (IY+d)_M} \to z$	X	↕	S	X	R	·

3

Arithmetic Instructions (16-bit)

Operation name	MNEMONICS	OP-code	Addressing							Bytes	States	Operation	Flag					
													7	6	4	2	1	0
			IMMED	EXT	IND	REG	REGI	IMP	REL				S	Z	H	P/V	N	C
ADD	ADD HL,ww	00 ww1 001				S		D		1	7	$HL_R + ww_R \rightarrow HL_R$	·	·	X	·	R	↕
	ADD IX,xx	11 011 101				S		D		2	10	$IX_R + xx_R \rightarrow IX_R$	·	·	X	·	R	↕
		00 xx1 001																
	ADD IY,yy	11 111 101				S		D		2	10	$IY_R + yy_R \rightarrow IY_R$	·	·	X	·	R	↕
		00 yy1 001																
ADC	ADC HL,ww	11 101 101				S		D		2	10	$HL_R + ww_R + c \rightarrow HL_R$	↕	↕	X	V	R	↕
		01 ww1 010																
DEC	DEC ww	00 ww1 011				S/D				1	4	$ww_R - 1 \rightarrow ww_R$	·	·	·	·	·	·
	DEC IX	11 011 101						S/D		2	7	$IX_R - 1 \rightarrow IX_R$	·	·	·	·		·
		00 101 011																
	DEC IY	11 111 101						S/D		2	7	$IY_R - 1 \rightarrow IY_R$	·	·	·	·	·	
		00 101 011																
INC	INC ww	00 ww0 011				S/D				1	4	$ww_R + 1 \rightarrow ww_R$		·	·	·		·
	INC IX	11 011 101						S/D		2	7	$IX_R + 1 \rightarrow IX_R$	·	·	·	·	·	·
		00 100 011																
	INC IY	11 111 101						S/D		2	7	$IY_R + 1 \rightarrow IY_R$	·		·			·
		00 100 011																
SBC	SBC HL,ww	11 101 101				S		D		2	10	$HL_R - ww_R - c \rightarrow HL_R$	↕	↕	X	V	S	↕
		01 ww0 010																

(to be continued)

Data Transfer Instructions

8-Bit Load

Operation name	MNEMONICS	OP-code	Addressing							Bytes	States	Operation	Flag					
													7	6	4	2	1	0
			IMMED	EXT	IND	REG	REGI	IMP	REL				S	Z	H	P/V	N	C
Load	LD A,I	11 101 101						S/D		2	6	Ir→Ar	↕	↕	R	IEF$_2$	R	·
8-bit		01 010 111																
Data	LD A,R	11 101 101						S/D		2	6	Rr→Ar	↕	↕	R	IEF$_2$	R	·
		01 011 111																
	LD A, (BC)	00 001 010					S	D		1	6	(BC)$_M$→Ar	·	·	·	·	·	·
	LD A, (DE)	00 011 010					S	D		1	6	(DE)$_M$→Ar	·	·	·	·	·	
	LD A, (mn)	00 111 010		S				D		3	12	(mn)$_M$→Ar	·	·	·	·	·	·
		< n >																
		< m >																
	LD I,A	11 101 101						S/D		2	6	Ar→Ir		·	·		·	·
		01 000 111																
	LD R,A	11 101 101						S/D		2	6	Ar→Rr	·	·		·		·
		01 001 111																
	LD (BC),A	00 000 010					D	S		1	7	Ar→(BC)$_M$	·	·	·	·	·	·
	LD (DE),A	00 010 010					D	S		1	7	Ar→(DE)$_M$	·		·	·		
	LD (mn),A	00 110 010		D				S		3	13	Ar→(mn)$_M$	·			·		·
		< n >																
		< m >																
	LD g,g'	01 g g'				S/D				1	4	gr'→gr	·		·	·	·	·
	LD g, (HL)	01 g 110				D	S			1	6	(HL)$_M$→gr	·	·	·	·	·	
	LD g,m	00 g 110	S			D				2	6	m→gr	·	·	·	·		·
		< m >																
	LD g, (IX+d)	11 011 101			S	D				3	14	(IX+d)$_M$→gr	·	·		·		·
		01 g 110																
		< d >																
	LD g, (IY+d)	11 111 101			S	D				3	14	(IY+d)$_M$→gr		·	·	·	·	·
		01 g 110																
		< d >																
	LD (HL),m	00 110 110	S				D			2	9	m→(HL)$_M$		·	·		·	·
		< m >																
	LD (IX+d),m	11 011 101	S		D					4	15	m→(IX+d)$_M$	·	·	·		·	·
		00 110 110																
		< d >																
		< m >																
	LD (IY+d),m	11 111 101	S		D					4	15	m→(IY+d)$_M$		·	·	·	·	·
		00 110 110																
		< d >																
		< m >																
	LD (HL),g	01 110 g				S	D			1	7	gr→(HL)$_M$	·	·	·	·	·	·
	LD (IX+d),g	11 011 101			D	S				3	15	gr→(IX+d)$_M$	·	·	·	·	·	·
		01 110 g																
		< d >																
	LD (IY+d),g	11 111 101			D	S				3	15	gr→(IY+d)$_M$	·	·	·	·	·	·
		01 110 g																
		< d >																

16-Bit Load

Operation name	MNEMONICS	OP-code	Addressing							Bytes	States	Operation	Flag					
													7	6	4	2	1	0
			IMMED	EXT	IND	REG	REGI	IMP	REL				S	Z	H	P/V	N	C
Load 16-bit Data	LD ww, mn	00 ww0 001 < n > < m >	S			D				3	9	mn→ww$_R$		·	·	·	·	
	LD IX, mn	11 011 101 00 100 001 < n > < m >	S					D		4	12	mn→IX$_R$			·		·	·
	LD IY, mn	11 111 101 00 100 001 < n > < m >	S					D		4	12	mn→IY$_R$		·				·
	LD SP, HL	11 111 001						S/D		1	4	HL$_R$→SP$_R$		·	·	·		·
	LD SP, IX	11 011 101 11 111 001						S/D		2	7	IX$_R$→SP$_R$		·		·		·
	LD SP, IY	11 111 101 11 111 001						S/D		2	7	IY$_R$→SP$_R$	·		·			
	LD ww, (mn)	11 101 101 01 ww1 011 < n > < m >		S		D				4	18	(mn+1)$_M$→wwHr (mn)$_M$→wwLr	·	·		·		
	LD HL, (mn)	00 101 010 < n > < m >		S				D		3	15	(mn+1)$_M$→Hr (mn)$_M$→Lr		·	·	·	·	
	LD IX, (mn)	11 011 101 00 101 010 < n > < m >		S				D		4	18	(mn+1)$_M$→IXHr (mn)$_M$→IXLr			·		·	·
	LD IY, (mn)	11 111 101 00 101 010 < n > < m >		S				D		4	18	(mn+1)$_M$→IYHr (mn)$_M$→IYLr	·					·
	LD (mn),ww	11 101 101 01 ww0 011 < n > < m >		D		S				4	19	wwHr→(mn+1)$_M$ wwLr→(mn)$_M$	·	·	·	·	·	·
	LD (mn),HL	00 100 010 < n > < m >		D				S		3	16	Hr→(mn+1)$_M$ Lr→(mn)$_M$	·		·	·		
	LD (mn), IX	11 011 101 00 100 010 < n > < m >		D				S		4	19	IXHr→(mn+1)$_M$ IXLr→(mn)$_M$	·	·	·		·	
	LD (mn), IY	11 111 101 00 100 010 < n > < m >		D				S		4	19	IYHr→(mn+1)$_M$ IYLr→(mn)$_M$	·			·		·

(to be continued)

Block Transfer

Operation name	MNEMONICS	OP-code	Addressing							Bytes	States	Operation	Flag					
													7	6	4	2	1	0
			IMMED	EXT	IND	REG	REGI	IMP	REL				S	Z	H	P/V	N	C
Block Transfer Search Data	CPD	11 101 101 10 101 001					S	S		2	12	$Ar-(HL)_M$ $BC_R-1\rightarrow BC_R$ $HL_R-1\rightarrow HL_R$	↕	② ↕	↕	① ↕	S	·
	CPDR	11 101 101 10 111 001					S	S		2	14 12	$BC_R\neq 0$ $Ar\neq(HL)_M$ $BC_R=0$ or $Ar=(HL)_M$ $Q\left[\begin{array}{l}Ar-(HL)_M\\ BC_R-1\rightarrow BC_R\\ HL_R-1\rightarrow HL_R\end{array}\right.$ Repeat Q until $Ar=(HL)_M$ or $BC_R=0$	↕	② ↕	↕	① ↕	S	·
	CPI	11 101 101 10 100 001					S	S		2	12	$Ar-(HL)_M$ $BC_R-1\rightarrow BC_R$ $HL_R+1\rightarrow HL_R$	↕	② ↕	↕	① ↕	S	·
	CPIR	11 101 101 10 110 001					S	S		2	14 12	$BC_R\neq 0$ $Ar\neq(HL)_M$ $BC_R=0$ or $Ar=(HL)_M$ $Q\left[\begin{array}{l}Ar-(HL)_M\\ BC_R-1\rightarrow BC_R\\ HL_R+1\rightarrow HL_R\end{array}\right.$ Repeat Q until $Ar=(HL)_M$ or $BC_R=0$	↕	② ↕	↕	① ↕	S	·
	LDD	11 101 101 10 101 000					S/D			2	12	$(HL)_M\rightarrow(DE)_M$ $BC_R-1\rightarrow BC_R$ $DE_R-1\rightarrow DE_R$ $HL_R-1\rightarrow HL_R$	·	·	R	① ↕	R	·
	LDDR	11 101 101 10 111 000					S/D			2	$14(BC_R\neq 0)$ $12(BC_R=0)$	$Q\left[\begin{array}{l}(HL)_M\rightarrow(DE)_M\\ BC_R-1\rightarrow BC_R\\ DE_R-1\rightarrow DE_R\\ HL_R-1\rightarrow HL_R\end{array}\right.$ Repeat Q until $BC_R=0$	·	·	R	R	R	·
	LDI	11 101 101 10 100 000					S/D			2	12	$(HL)_M\rightarrow(DE)_M$ $BC_R-1\rightarrow BC_R$ $DE_R+1\rightarrow DE_R$ $HL_R+1\rightarrow HL_R$	·	·	R	① ↕	R	·
	LDIR	11 101 101 10 110 000					S/D			2	$14(BC_R\neq 0)$ $12(BC_R=0)$	$Q\left[\begin{array}{l}(HL)_M\rightarrow(DE)_M\\ BC_R-1\rightarrow BC_R\\ DE_R+1\rightarrow DE_R\\ HL_R+1\rightarrow HL_R\end{array}\right.$ Repeat Q until $BC_R=0$	·	·	R	R	R	·

① $P/V=0$ $BC_R-1=0$
$P/V=1$ $BC_R-1\neq 0$

② $Z=1$ $Ar=(HL)_M$
$Z=0$ $Ar\neq(HL)_M$

Stack and Exchange

Operation name	MNEMONICS	OP-code	Addressing							Bytes	States	Operation	Flag					
													7	6	4	2	1	0
			IMMED	EXT	IND	REG	REGI	IMP	REL				S	Z	H	P/V	N	C
PUSH	PUSH zz	11 zz0 101				S		D		1	11	$zzLr \rightarrow (SP-2)_M$	·	·		·		·
												$zzHr \rightarrow (SP-1)_M$						
												$SP_R-2 \rightarrow SP_R$						
	PUSH IX	11 011 101						S/D		2	14	$IXLr \rightarrow (SP-2)_M$		·	·	·	·	·
		11 100 101										$IXHr \rightarrow (SP-1)_M$						
												$SP_R-2 \rightarrow SP_R$						
	PUSH IY	11 111 101						S/D		2	14	$IYLr \rightarrow (SP-2)_M$	·	·	·		·	·
		11 100 101										$IYHr \rightarrow (SP-1)_M$						
												$SP_R-2 \rightarrow SP_R$						
POP	POP zz	11 zz0 001				D		S		1	9	$(SP+1)_M \rightarrow zzHr$	·	·		·	·	·
												$(SP)_M \rightarrow zzLr$						
												$SP_R+2 \rightarrow SP_R$						
	POP IX	11 011 101						S/D		2	12	$(SP+1)_M \rightarrow IXHr$	·	·	·	·		·
		11 100 001										$(SP)_M \rightarrow IXLr$						
												$SP_R+2 \rightarrow SP_R$						
	POP IY	11 111 101						S/D		2	12	$(SP+1)_M \rightarrow IYHr$	·	·	·	·	·	·
		11 100 001										$(SP)_M \rightarrow IYLr$						
												$SP_R+2 \rightarrow SP_R$						
Exchange	EX AF,AF'	00 001 000						S/D		1	4	$AF_R \leftrightarrow AF_R'$	·		·	·	·	·
	EX DE,HL	11 101 011						S/D		1	3	$DE_R \leftrightarrow HL_R$		·		·	·	·
	EXX	11 011 001						S/D		1	3	$BC_R \leftrightarrow BC_R'$	·	·	·	·	·	·
												$DE_R \leftrightarrow DE_R'$						
												$HL_R \leftrightarrow HL_R'$						
	EX (SP),HL	11 100 011						S/D		1	16	$Hr \leftrightarrow (SP+1)_M$	·	·	·	·	·	·
												$Lr \leftrightarrow (SP)_M$						
	EX (SP),IX	11 011 101						S/D		2	19	$IXHr \leftrightarrow (SP+1)_M$	·	·	·	·	·	·
		11 100 011										$IXLr \leftrightarrow (SP)_M$						
	EX (SP),IY	11 111 101						S/D		2	19	$IYHr \leftrightarrow (SP+1)_M$	·	·	·	·	·	·
		11 100 011										$IYLr \leftrightarrow (SP)_M$						

Program Control Instructions

Operation name	MNEMONICS	OP-code	Addressing							Bytes	States	Operation	Flag					
													7	6	4	2	1	0
			IMMED	EXT	IND	REG	REGI	IMP	REL				S	Z	H	P/V	N	C
Call	CALL mn	11 001 101		D						3	16	$PCHr \rightarrow (SP-1)_M$		·	·	·	·	·
		< n >										$PCLr \rightarrow (SP-2)_M$						
		< m >										$mn \rightarrow PC_R$						
												$SP_R - 2 \rightarrow SP_R$						
	CALL f, mn	11 f 100		D						3	6(f false)	continue f is false	·	·	·	·	·	·
		< n >									16(f true)	CALL mn f is true						
		< m >																
Jump	DJNZ j	00 010 000							D	2	9 (Br≠0)		·	·	·	·	·	·
		< j-2 >								2	7 (Br=0)							
												$Br - 1 \rightarrow Br$						
												continue Br=0						
												$PC_R + j \rightarrow PC_R$ Br≠0						
	JP f, mn	11 f 010		D						3	6 (f false)		·	·	·	·	·	·
		< n >								3	9 (f true)							
		< m >										$mn \rightarrow PC_R$.f is true						
												continue f is false						
	JP mn	11 000 011		D						3	9	$mn \rightarrow PC_R$	·	·		·	·	·
		< n >																
		< m >																
	JP (HL)	11 101 001					D			1	3	$HL_R \rightarrow PC_R$	·	·	·	·	·	·
	JP (IX)	11 011 101					D			2	6	$IX_R \rightarrow PC_R$	·	·	·	·	·	·
		11 101 001																
	JP (IY)	11 111 101					D			2	6	$IY_R \rightarrow PC_R$	·	·	·	·	·	·
		11 101 001																
	JR j	00 011 000							D	2	8	$PC_R + j \rightarrow PC_R$	·	·	·	·	·	·
		< j-2 >																
	JR C,j	00 111 000							D	2	6	continue C=0	·	·	·	·	·	·
		< j-2 >								2	8	$PC_R + j \rightarrow PC_R$ C=1						
	JR NC,j	00 110 000							D	2	6	continue C=1	·	·	·	·	·	·
		< j-2 >								2	8	$PC_R + j \rightarrow PC_R$ C=0						
	JR Z,j	00 101 000							D	2	6	continue Z=0	·	·	·	·	·	
		< j-2 >								2	8	$PC_R + j \rightarrow PC_R$ Z=1						
	JR NZ,j	00 100 000							D	2	6	continue Z=1	·	·	·	·	·	·
		< j-2 >								2	8	$PC_R + j \rightarrow PC_R$ Z=0						
Return	RET	11 001 001						D		1	9	$(SP)_M \rightarrow PCLr$	·	·	·	·	·	·
												$(SP+1)_M \rightarrow PCHr$						
												$SP_R + 2 \rightarrow SP_R$						
	RET f	11 f 000						D		1	5(f false)	continue f is false	·	·	·	·	·	·
										1	10(f true)	RET f is true						
	RETI	11 101 101						D		2	12	$(SP)_M \rightarrow PCLr$	·	·	·	·	·	·
		01 001 101										$(SP+1)_M \rightarrow PCHr$						
												$SP_R + 2 \rightarrow SP_R$						
	RETN	11 101 101						D		2	12	$(SP)_M \rightarrow PCLr$	·	·	·	·	·	·
		01 000 101										$(SP+1)_M \rightarrow PCHr$						
												$SP_R + 2 \rightarrow SP_R$						
												$IEF_2 \rightarrow IEF_1$						

(to be continued)

Operation name	MNEMONICS	OP-code	Addressing							Bytes	States	Operation	Flag					
													7	6	4	2	1	0
			IMMED	EXT	IND	REG	REGI	IMP	REL				S	Z	H	P/V	N	C
Restart	RST v	11 v 111						D		1	11	PCHr→$(SP-1)_M$ PCLr→$(SP-2)_M$ 0→PCHr v→PCLr SP_R-2→SP_R	·	·	·	·	·	·

I/O Instructions

Operation name	MNEMONICS	OP-code	Addressing							Bytes	States	Operation	Flag					
													7	6	4	2	1	0
			IMMED	EXT	IND	REG	REGI	IMP	IO				S	Z	H	P/V	N	C
INPUT	IN A,(m)	11 011 011 < m >						D	S	2	9	$(Am)_I$→Ar m→A_0~A_7 Ar→A_8~A_{15}	·	·		·	·	·
	IN g,(C)	11 101 101 01 g 000				D			S	2	9	$(BC)_I$→gr g=110 Only the flags will change Cr→A_0~A_7 Br→A_8~A_{15}	↕	↕	R	P	R	·
	INO g,(m)	11 101 101 00 g 000 < m >				D			S	3	12	$(00m)_I$→gr g=110 Only the flags will change m→A_0~A_7 00→A_8~A_{15}	↕	↕	R	P	R	
	IND	11 101 101 10 101 010					D		S	2	12	$(BC)_I$→$(HL)_M$ HL_R-1→HL_R Br−1→Br Cr→A_0~A_7 Br→A_8~A_{15}	X	③ ↕	X	X	④ ↕	X
	INDR	11 101 101 10 111 010					D		S	2	14(Br≠0) 12(Br=0)	Q [$(BC)_I$→$(HL)_M$ HL_R-1→HL_R Br−1→Br] Repeat Q until Br=0 Cr→A_0~A_7 Br→A_8~A_{15}	X	S	X	X	④ ↕	X
	INI	11 101 101 10 100 010					D		S	2	12	$(BC)_I$→$(HL)_M$ HL_R+1→HL_R Br−1→Br Cr→A_0~A_7 Br→A_8~A_{15}	X	③ ↕	X	X	④ ↕	X
	INIR	11 101 101 10 110 010					D		S	2	14(Br≠0) 12(Br=0)	Q [$(BC)_I$→$(HL)_M$ HL_R+1→HL_R Br−1→Br] Repeat Q until Br=0 Cr→A_0~A_7 Br→A_8~A_{15}	X	S	X	X	④ ↕	X

(to be continued)

③ Z=1 Br−1=0
Z=0 Br−1≠0
④ N=1 MSB of Data=1
N=0 MSB of Data=0

Operation name	MNEMONICS	OP-code	Addressing							Bytes	States	Operation	Flag					
													7	6	4	2	1	0
			IMMED	EXT	IND	REG	REGI	IMP	IO				S	Z	H	P/V	N	C
OUTPUT	OUT (m),A	11 010 011 < m >						S	D	2	10	$Ar \rightarrow (Am)_I$ $m \rightarrow A_0 \sim A_7$ $Ar \rightarrow A_8 \sim A_{15}$	·	·	·	·	·	·
	OUT (C),g	11 101 101 01 g 001				S			D	2	10	$gr \rightarrow (BC)_I$ $Cr \rightarrow A_0 \sim A_7$ $Br \rightarrow A_8 \sim A_{15}$		·	·	·	·	·
	OUT0 (m),g	11 101 101 00 g 001 < m >				S			D	3	13	$gr \rightarrow (00m)_I$ $m \rightarrow A_0 \sim A_7$ $00 \rightarrow A_8 \sim A_{15}$	·	·	·	·	·	·
	OTDM	11 101 101 10 001 011					S		D	2	14	$(HL)_M \rightarrow (00C)_I$ $HL_R - 1 \rightarrow HL_R$ $Cr - 1 \rightarrow Cr$ $Br - 1 \rightarrow Br$ $Cr \rightarrow A_0 \sim A_7$ $00 \rightarrow A_8 \sim A_{15}$	↕	③ ↕	↕	P	④ ↕	↕
	OTDMR	11 101 101 10 011 011					S		D	2	16(Br≠0) 14(Br=0)	$(HL)_M \rightarrow (00C)_I$ Q $HL_R - 1 \rightarrow HL_R$ $Cr - 1 \rightarrow Cr$ $Br - 1 \rightarrow Br$ Repeat Q until $Br = 0$ $Cr \rightarrow A_0 \sim A_7$ $00 \rightarrow A_8 \sim A_{15}$	R	S	R	S	④ ↕	R
	OTDR	11 101 101 10 111 011					S		D	2	14(Br≠0) 12(Br=0)	$(HL)_M \rightarrow (BC)_I$ Q $HL_R - 1 \rightarrow HL_R$ $Br - 1 \rightarrow Br$ Repeat Q until $Br = 0$ $Cr \rightarrow A_0 \sim A_7$ $Br \rightarrow A_8 \sim A_{15}$	X	S	X	X	④ ↕	X
	OUTI	11 101 101 10 100 011					S		D	2	12	$(HL)_M \rightarrow (BC)_I$ $HL_R + 1 \rightarrow HL_R$ $Br - 1 \rightarrow Br$ $Cr \rightarrow A_0 \sim A_7$ $Br \rightarrow A_8 \sim A_{15}$	X	③ ↕	X	X	④ ↕	X
	OTIR	11 101 101 10 110 011					S		D	2	14(Br≠0) 12(Br=0)	$(HL)_M \rightarrow (BC)_I$ Q $HL_R + 1 \rightarrow HL_R$ $Br - 1 \rightarrow Br$ Repeat Q until $Br = 0$ $Cr \rightarrow A_0 \sim A_7$ $Br \rightarrow A_8 \sim A_{15}$	X	S	X	X	④ ↕	X
	TSTIO m	11 101 101 01 110 100 < m >	S						S	3	12	$(00C)_I$ m $Cr \rightarrow A_0 \sim A_7$ $00 \rightarrow A_8 \sim A_{15}$	↕	↕	S	P	R	R

(to be continued)

③ Z=1 Br−1=0
Z=0 Br−1≠0
④ N=1 MSB of Data=1
N=0 MSB of Data=0

Operation name	MNEMONICS	OP-code	Addressing							Bytes	States	Operation	Flag					
													7	6	4	2	1	0
			IMMED	EXT	IND	REG	REGI	IMP	IO				S	Z	H	P/V	N	C
OUTPUT	OTIM	11 101 101 10 000 011					S		D	2	14	$(HL)_M \rightarrow (00C)_I$ $HL_R + 1 \rightarrow HL_R$ $Cr + 1 \rightarrow Cr$ $Br - 1 \rightarrow Br$ $Cr \rightarrow A_0 \sim A_7$ $00 \rightarrow A_8 \sim A_{15}$	↕	$\overline{③}$ ↕	↕	P	④ ↕	↕
	OTIMR	11 101 101 10 010 011					S		D	2	16(Br≠0) 14(Br=0)	Q [$(HL)_M \rightarrow (00C)_I$ $HL_R + 1 \rightarrow HL_R$ $Cr + 1 \rightarrow Cr$ $Br - 1 \rightarrow Br$] Repeat Q until Br=0 $Cr \rightarrow A_0 \sim A_7$ $00 \rightarrow A_8 \sim A_{15}$	R	S	R	S	④ ↕	R
	OUTD	11 101 101 10 101 011					S		D	2	12	$(HL)_M \rightarrow (BC)_I$ $HL_R - 1 \rightarrow HL_R$ $Br - 1 \rightarrow Br$ $Cr \rightarrow A_0 \sim A_7$ $Br \rightarrow A_8 \sim A_{15}$	X	③ ↕	X	X	④ ↕	X

③ Z=1 Br−1=0
Z=0 Br−1≠0
④ N=1 MSB of Data=1
N=0 MSB of Data=0

Special Control Instructions

Operation name	MNEMONICS	OP-code	Addressing							Bytes	States	Operation	Flag					
													7	6	4	2	1	0
			IMMED	EXT	IND	REG	REGI	IMP	REL				S	Z	H	P/V	N	C
Special Function	DAA	00 100 111						S/D		1	4	Decimal Adjust Accumulator	↕	↕	↕	P	·	↕
Carry Control	CCF	00 111 111								1	3	$\overline{c} \rightarrow c$	·	·	R	·	R	↕
	SCF	00 110 111								1	3	1→c	·	·	R	·	R	S
CPU Control	DI	11 110 011								1	3	$0 \rightarrow IEF_1, 0 \rightarrow IEF_2$ ⑤	·	·	·		·	·
	EI	11 111 011								1	3	$1 \rightarrow IEF_1, 1 \rightarrow IEF_2$ ⑤	·	·	·	·	·	·
	HALT	01 110 110								1	3	CPU halted	·	·	·		·	·
	IM 0	11 101 101 01 000 110								2	6	Interrupt mode 0	·	·	·		·	·
	IM 1	11 101 101 01 010 110								2	6	Interrupt mode 1		·	·	·		·
	IM 2	11 101 101 01 011 110								2	6	Interrupt mode 2	·	·	·	·	·	·
	NOP	00 000 000								1	3	No operation	·	·	·	·	·	·
	SLP	11 101 101 01 110 110								2	8	Sleep	·	·	·	·	·	·

⑤ Interrupts are not sampled at the end of DI or EI.

20 INSTRUCTION SUMMARY IN ALPHABETICAL ORDER

MNEMONICS	Bytes	Machine Cycles	States
ADC A,m	2	2	6
ADC A,g	1	2	4
ADC A, (HL)	1	2	6
ADC A, (IX+d)	3	6	14
ADC A, (IY+d)	3	6	14
ADD A,m	2	2	6
ADD A,g	1	2	4
ADD A, (HL)	1	2	6
ADD A, (IX+d)	3	6	14
ADD A, (IY+d)	3	6	14
ADC HL,ww	2	6	10
ADD HL,ww	1	5	7
ADD IX,xx	2	6	10
ADD IY,yy	2	6	10
AND m	2	2	6
AND g	1	2	4
AND (HL)	1	2	6
AND (IX+d)	3	6	14
AND (IY+d)	3	6	14
BIT b, (HL)	2	3	9
BIT b, (IX+d)	4	5	15
BIT b, (IY+d)	4	5	15
BIT b,g	2	2	6
CALL f,mn	3	2	6 (If condition is false)
	3	6	16 (If condition is true)

(to be continued)

MNEMONICS	Bytes	Machine Cycles	States
CALL mn	3	6	16
CCF	1	1	3
CPD	2	6	12
CPDR	2	8	14
			(If $BC_R \neq 0$ and $Ar \neq (HL)_M$)
	2	6	12
			(If $BC_R = 0$ or $Ar = (HL)_M$)
CP (HL)	1	2	6
CPI	2	6	12
CPIR	2	8	14
			(If $BC_R \neq 0$ and $Ar \neq (HL)_M$)
	2	6	12
			(If $BC_R = 0$ or $Ar = (HL)_M$)
CP (IX+d)	3	6	14
CP (IY+d)	3	6	14
CPL	1	1	3
CP m	2	2	6
CP g	1	2	4
DAA	1	2	4
DEC (HL)	1	4	10
DEC IX	2	3	7
DEC IY	2	3	7
DEC (IX+d)	3	8	18
DEC (IY+d)	3	8	18
DEC g	1	2	4
DEC ww	1	2	4
DI	1	1	3

(to be continued)

MNEMONICS	Bytes	Machine Cycles	States
DJNZ j	2	5	9 (If Br≠0)
	2	3	7 (If Br=0)
EI	1	1	3
EX AF,AF'	1	2	4
EX DE,HL	1	1	3
EX (SP),HL	1	6	16
EX (SP),IX	2	7	19
EX (SP),IY	2	7	19
EXX	1	1	3
HALT	1	1	3
IM 0	2	2	6
IM 1	2	2	6
IM 2	2	2	6
INC g	1	2	4
INC (HL)	1	4	10
INC (IX+d)	3	8	18
INC (IY+d)	3	8	18
INC ww	1	2	4
INC IX	2	3	7
INC IY	2	3	7
IN A,(m)	2	3	9
IN g,(C)	2	3	9
INI	2	4	12
INIR	2	6	14 (If Br≠0)
	2	4	12 (If Br=0)
IND	2	4	12
INDR	2	6	14 (If Br≠0)

(to be continued)

MNEMONICS	Bytes	Machine Cycles	States
INDR	2	4	12 (If Br=0)
INO g,(m)	3	4	12
JP f,mn	3	2	6
			(If f is false)
	3	3	9
			(If f is true)
JP (HL)	1	1	3
JP (IX)	2	2	6
JP (IY)	2	2	6
JP mn	3	3	9
JR j	2	4	8
JR C,j	2	2	6
			(If condition is false)
	2	4	8
			(If condition is true)
JR NC,j	2	2	6
			(If condition is false)
	2	4	8
			(If condition is true)
JR Z,j	2	2	6
			(If condition is false)
	2	4	8
			(If condition is true)
JR NZ,j	2	2	6
			(If condition is false)
	2	4	8
			(If condition is true)

(to be continued)

MNEMONICS	Bytes	Machine Cycles	States
LD A, (BC)	1	2	6
LD A, (DE)	1	2	6
LD A,I	2	2	6
LD A, (mn)	3	4	12
LD A,R	2	2	6
LD (BC),A	1	3	7
LDD	2	4	12
LD (DE),A	1	3	7
LD ww,mn	3	3	9
LD ww,(mn)	4	6	18
LDDR	2	6	14 (If $BC_R \neq 0$)
	2	4	12 (If $BC_R = 0$)
LD (HL),m	2	3	9
LD HL,(mn)	3	5	15
LD (HL),g	1	3	7
LDI	2	4	12
LD I,A	2	2	6
LDIR	2	6	14 (If $BC_R \neq 0$)
	2	4	12 (If $BC_R = 0$)
LD IX,mn	4	4	12
LD IX,(mn)	4	6	18
LD (IX+d),m	4	5	15
LD (IX+d),g	3	7	15
LD IY,mn	4	4	12
LD IY,(mn)	4	6	18
LD (IY+d),m	4	5	15
LD (IY+d),g	3	7	15

(to be continued)

MNEMONICS	Bytes	Machine Cycles	States
LD (mn),A	3	5	13
LD (mn),ww	4	7	19
LD (mn),HL	3	6	16
LD (mn),IX	4	7	19
LD (mn),IY	4	7	19
LD R,A	2	2	6
LD g,(HL)	1	2	6
LD g,(IX+d)	3	6	14
LD g,(IY+d)	3	6	14
LD g,m	2	2	6
LD g,g'	1	2	4
LD SP,HL	1	2	4
LD SP,IX	2	3	7
LD SP,IY	2	3	7
MLT ww	2	13	17
NEG	2	2	6
NOP	1	1	3
OR (HL)	1	2	6
OR (IX+d)	3	6	14
OR (IY+d)	3	6	14
OR m	2	2	6
OR g	1	2	4
OTDM	2	6	14
OTDMR	2	8	16 (If Br≠0)
	2	6	14 (If Br=0)
OTDR	2	6	14 (If Br≠0)
	2	4	12 (If Br=0)

(to be continued)

MNEMONICS	Bytes	Machine Cycles	States
OTIM	2	6	14
OTIMR	2	8	16 (If Br≠0)
	2	6	14 (If Br=0)
OTIR	2	6	14 (If Br≠0)
	2	4	12 (If Br=0)
OUTD	2	4	12
OUTI	2	4	12
OUT (m),A	2	4	10
OUT (C),g	2	4	10
OUT0 (m),g	3	5	13
POP IX	2	4	12
POP IY	2	4	12
POP zz	1	3	9
PUSH IX	2	6	14
PUSH IY	2	6	14
PUSH zz	1	5	11
RES b,(HL)	2	5	13
RES b,(IX+d)	4	7	19
RES b,(IY+d)	4	7	19
RES b,g	2	3	7
RET	1	3	9
RET f	1	3	5 (If condition is false)
	1	4	10 (If condition is true)
RETI	2	4	12
RETN	2	4	12

(to be continued)

MNEMONICS	Bytes	Machine Cycles	States
RLA	1	1	3
RLCA	1	1	3
RLC (HL)	2	5	13
RLC (IX+d)	4	7	19
RLC (IY+d)	4	7	19
RLC g	2	3	7
RLD	2	8	16
RL (HL)	2	5	13
RL (IX+d)	4	7	19
RL (IY+d)	4	7	19
RL g	2	3	7
RRA	1	1	3
RRCA	1	1	3
RRC (HL)	2	5	13
RRC (IX+d)	4	7	19
RRC (IY+d)	4	7	19
RRC g	2	3	7
RRD	2	8	16
RR (HL)	2	5	13
RR (IX+d)	4	7	19
RR (IY+d)	4	7	19
RR g	2	3	7
RST v	1	5	11
SBC A,(HL)	1	2	6
SBC A,(IX+d)	3	6	14
SBC A,(IY+d)	3	6	14
SBC A,m	2	2	6

(to be continued)

MNEMONICS	Bytes	Machine Cycles	States
SBC A,g	1	2	4
SBC HL,ww	2	6	10
SCF	1	1	3
SET b,(HL)	2	5	13
SET b,(IX+d)	4	7	19
SET b,(IY+d)	4	7	19
SET b,g	2	3	7
SLA (HL)	2	5	13
SLA (IX+d)	4	7	19
SLA (IY+d)	4	7	19
SLA g	2	3	7
SLP	2	2	8
SRA (HL)	2	5	13
SRA (IX+d)	4	7	19
SRA (IY+d)	4	7	19
SRA g	2	3	7
SRL (HL)	2	5	13
SRL (IX+d)	4	7	19
SRL (IY+d)	4	7	19
SRL g	2	3	7
SUB (HL)	1	2	6
SUB (IX+d)	3	6	14
SUB (IY+d)	3	6	14
SUB m	2	2	6
SUB g	1	2	4
TSTIO m	3	4	12
TST g	2	3	7

(to be continued)

MNEMONICS	Bytes	Machine Cycles	States
TST m	3	3	9
TST (HL)	2	4	10
XOR (HL)	1	2	6
XOR (IX+d)	3	6	14
XOR (IY+d)	3	6	14
XOR m	2	2	6
XOR g	1	2	4

21 OP-CODE MAP

Table 18 1st op-code map
Instruction format : XX

				ww(LO=ALL)												LO=0~7				
				BC	DE	HL	SP									BC	DE	HL	AF	zz
				g (LO=0~7)												NZ	NC	PO	P	f
				B	D	H	(HL)	B	D	H	(HL)					00H	10H	20H	30H	v
			HI	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111	
		LO		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
S (HI=ALL)	B	0000	0	NOP	DJNZ j	JR NZ, j	JR NC, j	LD g, s				ADD A ,s	SUB s	AND s	OR s	RET f				0
	C	0001	1	LD ww, mn							NOTE1)					POP zz				1
	D	0010	2	LD (ww), A		LD (mn) ,HL	LD (mn) ,A									JP f, mn				2
	E	0011	3	INC ww												JP mn	OUT (m) ,A	EX (SP) ,HL	DI	3
	H	0100	4	INC g			NOTE1)									CALL f, mn				4
	L	0101	5	DEC g			NOTE1)									PUSH zz				5
	(HL)	0110	6	LD g, m			NOTE1)	NOTE2)			HALT	NOTE2)	NOTE2)	NOTE2)	NOTE2)	ADD A,m	SUB m	AND m	OR m	6
	A	0111	7	RLCA	RLA	DAA	SCF									RST v				7
	B	1000	8	EX AF,AF'	JR j	JR Z, j	JR C, j	LD g, s				ADC A ,s	SBC A ,s	XOR s	CP s	RET f				8
	C	1001	9	ADD HL, ww												RET	EXX	JP (HL)	LD SP, HL	9
	D	1010	A	LD A, (ww)		LD HL, (mn)	LD A, (mn)									JP f, mn				A
	E	1011	B	DEC ww												Table2	IN A, (m)	EX DE, HL	EI	B
	H	1100	C	INC g												CALL f, mn				C
	L	1101	D	DEC g												CALL mn	NOTE3)	Table3	NOTE3)	D
	(HL)	1110	E	LD g, m				NOTE2)				NOTE2)	NOTE2)	NOTE2)	NOTE2)	ADC A,m	SBC A,m	XOR m	CP m	E
	A	1111	F	RRCA	RRA	CPL	CCF									RST v				F
				0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
				C	E	L	A	C	E	L	A					Z	C	PE	M	f
				g(LO=8~F)												08H	18H	28H	38H	v
																LO=8~F				

NOTE1) (HL) replaces g.
2) (HL) replaces s.
3) If DDH is supplemented as 1st op-code for the instructions which have HL or (HL) as an operand in Table 18, the instructions are executed replacing HL with IX and (HL) with (IX+d).

```
ex.     22H : LD (mn), HL
    DDH 22H : LD (mn), IX
```

If FDH is supplemented as 1st op-code for the instructions which have HL or (HL) as an operand in Table 18, the instructions are executed replacing HL with IY and (HL) with (IY+d).

```
ex.     34H : INC (HL)
    FDH 34H : INC (IY+d)
```

However, JP (HL) and EX DE, HL are exception and note the followings.
If DDH is supplemented as 1st op-code for JP (HL), (IX) replaces (HL) as operand and JP (IX) is executed.
If FDH is supplemented as 1st op-code for JP (HL), (IX) replaces (HL) as operand and JP (IY) is executed.
Even if DDH or FDH is supplemented as 1st op-code for EX DE, HL, HL is not replaced and the instruction is regarded as illegal instruction.

Table 19 2nd op-code map
Instruction format : CB XX

								b (LO=0~7)												
								0	2	4	6	0	2	4	6	0	2	4	6	
			HI	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111	
		LO		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
g (HI=ALL)	B	0000	0	RLC g	RL g	SLA g		BIT b,g				RES b,g				SET b,g				0
	C	0001	1																	1
	D	0010	2																	2
	E	0011	3																	3
	H	0100	4																	4
	L	0101	5																	5
	(HL)	0110	6	NOTE1)	NOTE1)	NOTE1)		NOTE1)				NOTE1)				NOTE1)				6
	A	0111	7																	7
	B	1000	8	RRC g	RR g	SRA g	SRL g	BIT b,g				RES b,g				SET b,g				8
	C	1001	9																	9
	D	1010	A																	A
	E	1011	B																	B
	H	1100	C																	C
	L	1101	D																	D
	(HL)	1110	E	NOTE1)	NOTE1)	NOTE1)	NOTE1)	NOTE1)				NOTE1)				NOTE1)				E
	A	1111	F																	F
				0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
								1	3	5	7	1	3	5	7	1	3	5	7	
								b (LO=8~F)												

NOTE1) If DDH is supplemented as 1st op-code for the instructions which have (HL) as operand in Table 19, the instructions are executed replacing (HL) with (IX+d).
If FDH is supplemented as 1st op-code for the instructions which have (HL) as operand in Table 19, the instructions are executed replacing (HL) with (IY+d).

Table 20 2nd op-code map
Instruction format : ED XX

						ww (LO=ALL)												
						BC	DE	HL	SP									
		g (LO=0~7)																
		B	D	H		B	D	H										
	HI	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111	
LO		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0000	0	IN0 g, (m)				IN g, (C)						LDI	LDIR					0
0001	1	OUT0 (m),g				OUT (C),g						CPI	CPIR					1
0010	2					SBC HL, ww						INI	INIR					2
0011	3					LD (mn), ww				OTIM	OTIMR	OUTI	OTIR					3
0100	4	TST g			TST (HL)	NEG		TST m	TSTIO m									4
0101	5					RETN												5
0110	6					IM 0	IM 1		SLP									6
0111	7					LD I,A	LD A,I	RRD										7
1000	8	IN0 g, (m)				IN g, (C)						LDD	LDDR					8
1001	9	OUT0 (m),g				OUT (C),g						CPD	CPDR					9
1010	A					ADC HL, ww						IND	INDR					A
1011	B					LD ww, (mn)				OTDM	OTDMR	OUTD	OTDR					B
1100	C	TST g				MLT ww												C
1101	D					RETI												D
1110	E						IM 2											E
1111	F					LD R,A	LD A,R	RLD										F
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
		C	E	L	A	C	E	L	A									
		g (LO=8~F)																

22 BUS AND CONTROL SIGNAL CONDITION IN EACH MACHINE CYCLE

* (ADDRESS) : invalid
Z (DATA) : high impedance.

Instruction	Machine Cycle	States	ADDRESS	DATA	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{LIR}$	$\overline{HALT}$	ST
ADD HL,ww	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2 ~MC_5	TiTiTiTi	*	Z	1	1	1	1	1	1	1
ADD IX,xx ADD IY,yy	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC_3 ~MC_6	TiTiTiTi	*	Z	1	1	1	1	1	1	1
ADC HL,ww SBC HL,ww	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC_3 ~MC_6	TiTiTiTi	*	Z	1	1	1	1	1	1	1
ADD A,g ADC A,g SUB g SBC A,g AND g OR g XOR g CP g	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	Ti	*	Z	1	1	1	1	1	1	1
ADD A,m ADC A,m SUB m SBC A,m AND m OR m XOR m CP m	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	1st operand Address	m	0	1	0	1	1	1	1
ADD A, (HL) ADC A, (HL) SUB (HL) SBC A, (HL) AND (HL) OR (HL) XOR (HL) CP (HL)	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	HL	DATA	0	1	0	1	1	1	1
ADD A, (IX+d) ADD A, (IY+d) ADC A, (IX+d) ADC A, (IY+d) SUB (IX+d) SUB (IY+d) SBC A, (IX+d)	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1

(to be continued)

Instruction	Machine Cycle	States	ADDRESS	DATA	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{LIR}$	$\overline{HALT}$	ST
SBC A, (IY+d) AND (IX+d) AND (IY+d) OR (IX+d) OR (IY+d) XOR (IX+d) XOR (IY+d) CP (IX+d) CP (IY+d)	MC_3	$T_1T_2T_3$`	1st operand Address	d	0	1	0	1	1	1	1
	MC_4 $\sim MC_5$	T_iT_i	•	Z	1	1	1	1	1	1	1
	MC_6	$T_1T_2T_3$	IX+d IY+d	DATA	0	1	0	1	1	1	1
BIT b,g	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
BIT b, (HL)	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	HL	DATA	0	1	0	1	1	1	1
BIT b, (IX+d) BIT b, (IY+d)	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	1st operand Address	d	0	1	0	1	1	1	1
	MC_4	$T_1T_2T_3$	3rd op-code Address	3rd op-code	0	1	0	1	0	1	1
	MC_5	$T_1T_2T_3$	IX+d IY+d	DATA	0	1	0	1	1	1	1
CALL mn	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	1st operand Address	n	0	1	0	1	1	1	1
	MC_3	$T_1T_2T_3$	2nd operand Address	m	0	1	0	1	1	1	1
	MC_4	T_i	•	Z	1	1	1	1	1	1	1
	MC_5	$T_1T_2T_3$	SP−1	PCH	1	0	0	1	1	1	1
	MC_6	$T_1T_2T_3$	SP−2	PCL	1	0	0	1	1	1	1
CALL f,mn (If condition is false)	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	1st operand Address	n	0	1	0	1	1	1	1

(to be continued)

Instruction	Machine Cycle	States	ADDRESS	DATA	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{LIR}$	$\overline{HALT}$	ST
CALL f,mn (If condition is true)	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	1st operand Address	n	0	1	0	1	1	1	1
	MC_3	$T_1T_2T_3$	2nd operand Address	m	0	1	0	1	1	1	1
	MC_4	Ti	•	Z	1	1	1	1	1	1	1
	MC_5	$T_1T_2T_3$	SP−1	PCH	1	0	0	1	1	1	1
	MC_6	$T_1T_2T_3$	SP−2	PCL	1	0	0	1	1	1	1
CCF	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
CPI CPD	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	HL	DATA	0	1	0	1	1	1	1
	MC_4 $\sim MC_6$	TiTiTi	•	Z	1	1	1	1	1	1	1
CPIR CPDR (If $BC_R \neq 0$ and $Ar \neq (HL)_M$)	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	HL	DATA	0	1	0	1	1	1	1
	MC_4 $\sim MC_8$	TiTiTiTiTi	•	Z	1	1	1	1	1	1	1
CPIR CPDR (If $BC_R = 0$ or $Ar = (HL)_M$)	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	HL	DATA	0	1	0	1	1	1	1
	MC_4 $\sim MC_6$	TiTiTi	•	Z	1	1	1	1	1	1	1
CPL	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
DAA	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	Ti	•	Z	1	1	1	1	1	1	1
DI	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0

(to be continued)

Instruction	Machine Cycle	States	ADDRESS	DATA	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{LIR}$	$\overline{HALT}$	ST
DJNZ j (If Br≠0)	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	T_i *1	•	Z	1	1	1	1	1	1	1
	MC_3	$T_1T_2T_3$	1st operand Address	j-2	0	1	0	1	1	1	1
	MC_4 $\sim MC_5$	T_iT_i	•	Z	1	1	1	1	1	1	1
DJNZ j (If Br=0)	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	T_i *1	•	Z	1	1	1	1	1	1	1
	MC_3	$T_1T_2T_3$	1st operand Address	j-2	0	1	0	1	1	1	1
EI	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
EX DE, HL EXX	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
EX AF, AF'	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	T_i	•	Z	1	1	1	1	1	1	1
EX (SP), HL	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	SP	DATA	0	1	0	1	1	1	1
	MC_3	$T_1T_2T_3$	SP+1	DATA	0	1	0	1	1	1	1
	MC_4	T_i	•	Z	1	1	1	1	1	1	1
	MC_5	$T_1T_2T_3$	SP+1	H	1	0	0	1	1	1	1
	MC_6	$T_1T_2T_3$	SP	L	1	0	0	1	1	1	1
EX (SP),IX EX (SP),IY	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	SP	DATA	0	1	0	1	1	1	1
	MC_4	$T_1T_2T_3$	SP+1	DATA	0	1	0	1	1	1	1
	MC_5	T_i	•	Z	1	1	1	1	1	1	1

*1 DMA, REFRESH, or BUS RELEASE cannot be executed after this state (Request is ignored)

(to be continued)

Instruction	Machine Cycle	States	ADDRESS	DATA	$\overline{\text{RD}}$	$\overline{\text{WR}}$	$\overline{\text{ME}}$	$\overline{\text{IOE}}$	$\overline{\text{LIR}}$	$\overline{\text{HALT}}$	ST
EX (SP), IX EX (SP), IY	MC_6	$T_1T_2T_3$	SP+1	IXH IYH	1	0	0	1	1	1	1
	MC_7	$T_1T_2T_3$	SP	IXL IYL	1	0	0	1	1	1	1
HALT	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	—	——	Next op-code Address	Next op-code	0	1	0	1	0	0	0
IM 0 IM 1 IM 2	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
INC g DEC g	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	Ti	•	Z	1	1	1	1	1	1	1
INC (HL) DEC (HL)	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	HL	DATA	0	1	0	1	1	1	1
	MC_3	Ti	•	Z	1	1	1	1	1	1	1
	MC_4	$T_1T_2T_3$	HL	DATA	1	0	0	1	1	1	1
INC (IX+d) INC (IY+d) DEC (IX+d) DEC (IY+d)	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	1st operand Address	d	0	1	0	1	1	1	1
	MC_4 $\sim MC_5$	TiTi	•	Z	1	1	1	1	1	1	1
	MC_6	$T_1T_2T_3$	IX+d IY+d	DATA	0	1	0	1	1	1	1
	MC_7	Ti	•	Z	1	1	1	1	1	1	1
	MC_8	$T_1T_2T_3$	IX+d IY+d	DATA	1	0	0	1	1	1	1
INC ww DEC ww	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	Ti	•	Z	1	1	1	1	1	1	1
INC IX INC IY DEC IX DEC IY	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-dode	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC_3	Ti	•	Z	1	1	1	1	1	1	1

(to be continued)

Instruction	Machine Cycle	States	ADDRESS	DATA	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{LIR}$	$\overline{HALT}$	ST
IN A,(m)	MC$_1$	T$_1$T$_2$T$_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	T$_1$T$_2$T$_3$	1st operand Address	m	0	1	0	1	1	1	1
	MC$_3$	T$_1$T$_2$T$_3$	m to A_0~A_7 A to A_8~A_{15}	DATA	0	1	1	0	1	1	1
IN g,(C)	MC$_1$	T$_1$T$_2$T$_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	T$_1$T$_2$T$_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC$_3$	T$_1$T$_2$T$_3$	BC	DATA	0	1	1	0	1	1	1
INO g,(m)	MC$_1$	T$_1$T$_2$T$_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	T$_1$T$_2$T$_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC$_3$	T$_1$T$_2$T$_3$	1st operand Address	m	0	1	0	1	1	1	1
	MC$_4$	T$_1$T$_2$T$_3$	m to A_0~A_7 00H to A_8~A_{15}	DATA	0	1	1	0	1	1	1
INI IND	MC$_1$	T$_1$T$_2$T$_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	T$_1$T$_2$T$_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC$_3$	T$_1$T$_2$T$_3$	BC	DATA	0	1	1	0	1	1	1
	MC$_4$	T$_1$T$_2$T$_3$	HL	DATA	1	0	0	1	1	1	1
INIR INDR (If Br≠0)	MC$_1$	T$_1$T$_2$T$_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	T$_1$T$_2$T$_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC$_3$	T$_1$T$_2$T$_3$	BC	DATA	0	1	1	0	1	1	1
	MC$_4$	T$_1$T$_2$T$_3$	HL	DATA	1	0	0	1	1	1	1
	MC$_5$ ~MC$_6$	TiTi	•	Z	1	1	1	1	1	1	1
INIR INDR (If Br=0)	MC$_1$	T$_1$T$_2$T$_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	T$_1$T$_2$T$_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC$_3$	T$_1$T$_2$T$_3$	BC	DATA	0	1	1	0	1	1	1
	MC$_4$	T$_1$T$_2$T$_3$	HL	DATA	1	0	0	1	1	1	1

(to be continued)

Instruction	Machine Cycle	States	ADDRESS	DATA	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{LIR}$	$\overline{HALT}$	ST
JP mn	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	1st operand Address	n	0	1	0	1	1	1	1
	MC_3	$T_1T_2T_3$	2nd operand Address	m	0	1	0	1	1	1	1
JP f,mn (If f is false)	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	1st operand Address	n	0	1	0	1	1	1	1
JP f,mn (If f is true)	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	1st operand Address	n	0	1	0	1	1	1	1
	MC_3	$T_1T_2T_3$	2nd operand Address	m	0	1	0	1	1	1	1
JP (HL)	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
JP (IX) JP (IY)	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
JR j	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	1st operand Address	j-2	0	1	0	1	1	1	1
	MC_3 $\sim MC_4$	TiTi	•	Z	1	1	1	1	1	1	1
JR C,j JR NC,j JR Z,j JR NZ,j (If condition is false)	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	1st operand Address	j-2	0	1	0	1	1	1	1
JR C,j JR NC,j JR Z,j JR NZ,j (If condition is true)	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	1st operand Address	j-2	0	1	0	1	1	1	1
	MC_3 $\sim MC_4$	TiTi	•	Z	1	1	1	1	1	1	1
LD g,g'	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	Ti	•	Z	1	1	1	1	1	1	1
LD g,m	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	1st operand Address	m	0	1	0	1	1	1	1

(to be continued)

Instruction	Machine Cycle	States	ADDRESS	DATA	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{LIR}$	$\overline{HALT}$	ST
LD g, (HL)	MC$_1$	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	$T_1T_2T_3$	HL	DATA	0	1	0	1	1	1	1
LD g, (IX+d) LD g, (IY+d)	MC$_1$	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC$_3$	$T_1T_2T_3$	1st operand Address	d	0	1	0	1	1	1	1
	MC$_4$ ~MC$_5$	T_iT_i	•	Z	1	1	1	1	1	1	1
	MC$_6$	$T_1T_2T_3$	IX+d IY+d	DATA	0	1	0	1	1	1	1
LD (HL),g	MC$_1$	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	T_i	•	Z	1	1	1	1	1	1	1
	MC$_3$	$T_1T_2T_3$	HL	g	1	0	0	1	1	1	1
LD (IX+d),g LD (IY+d),g	MC$_1$	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC$_3$	$T_1T_2T_3$	1st operand Address	d	0	1	0	1	1	1	1
	MC$_4$ ~MC$_6$	$T_iT_iT_i$	•	Z	1	1	1	1	1	1	1
	MC$_7$	$T_1T_2T_3$	IX+d IY+d	g	1	0	0	1	1	1	1
LD (HL),m	MC$_1$	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	$T_1T_2T_3$	1st operand Address	m	0	1	0	1	1	1	1
	MC$_3$	$T_1T_2T_3$	HL	DATA	1	0	0	1	1	1	1
LD (IX+d),m LD (IY+d),m	MC$_1$	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC$_3$	$T_1T_2T_3$	1st operand Address	d	0	1	0	1	1	1	1
	MC$_4$	$T_1T_2T_3$	2nd operand Address	m	0	1	0	1	1	1	1
	MC$_5$	$T_1T_2T_3$	IX+d IY+d	DATA	1	0	0	1	1	1	1
LD A, (BC) LD A, (DE)	MC$_1$	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0

(to be continued)

Instruction	Machine Cycle	States	ADDRESS	DATA	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{LIR}$	$\overline{HALT}$	ST
LD A, (BC) LD A, (DE)	MC$_2$	$T_1T_2T_3$	BC DE	DATA	0	1	0	1	1	1	1
LD A,(mn)	MC$_1$	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	$T_1T_2T_3$	1st operand Address	n	0	1	0	1	1	1	1
	MC$_3$	$T_1T_2T_3$	2nd operand Address	m	0	1	0	1	1	1	1
	MC$_4$	$T_1T_2T_3$	mn	DATA	0	1	0	1	1	1	1
LD (BC),A LD (DE),A	MC$_1$	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	T_i	•	Z	1	1	1	1	1	1	1
	MC$_3$	$T_1T_2T_3$	BC DE	A	1	0	0	1	1	1	1
LD (mn),A	MC$_1$	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	$T_1T_2T_3$	1st operand Address	n	0	1	0	1	1	1	1
	MC$_3$	$T_1T_2T_3$	2nd operand Address	m	0	1	0	1	1	1	1
	MC$_4$	T_i	•	Z	1	1	1	1	1	1	1
	MC$_5$	$T_1T_2T_3$	mn	A	1	0	0	1	1	1	1
LD A,I LD A,R LD I,A LD R,A	MC$_1$	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
LD ww, mn	MC$_1$	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	$T_1T_2T_3$	1st operand Address	n	0	1	0	1	1	1	1
	MC$_3$	$T_1T_2T_3$	2nd operand Address	m	0	1	0	1	1	1	1
LD IX,mn LD IY,mn	MC$_1$	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC$_3$	$T_1T_2T_3$	1st operand Address	n	0	1	0	1	1	1	1
	MC$_4$	$T_1T_2T_3$	2nd operand Address	m	0	1	0	1	1	1	1
LD HL, (mn)	MC$_1$	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	$T_1T_2T_3$	1st operand Address	n	0	1	0	1	1	1	1

(to be continued)

Instruction	Machine Cycle	States	ADDRESS	DATA	$\overline{\text{RD}}$	$\overline{\text{WR}}$	$\overline{\text{ME}}$	$\overline{\text{IOE}}$	$\overline{\text{LIR}}$	$\overline{\text{HALT}}$	ST
LD HL, (mn)	MC_3	$T_1T_2T_3$	2nd operand Address	m	0	1	0	1	1	1	1
	MC_4	$T_1T_2T_3$	mn	DATA	0	1	0	1	1	1	1
	MC_5	$T_1T_2T_3$	mn+1	DATA	0	1	0	1	1	1	1
LD ww,(mn)	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	1st operand Address	n	0	1	0	1	1	1	1
	MC_4	$T_1T_2T_3$	2nd operand Address	m	0	1	0	1	1	1	1
	MC_5	$T_1T_2T_3$	mn	DATA	0	1	0	1	1	1	1
	MC_6	$T_1T_2T_3$	mn+1	DATA	0	1	0	1	1	1	1
LD IX,(mn) LD IY,(mn)	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	1st operand Address	n	0	1	0	1	1	1	1
	MC_4	$T_1T_2T_3$	2nd operand Address	m	0	1	0	1	1	1	1
	MC_5	$T_1T_2T_3$	mn	DATA	0	1	0	1	1	1	1
	MC_6	$T_1T_2T_3$	mn+1	DATA	0	1	0	1	1	1	1
LD (mn),HL	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	1st operand Address	n	0	1	0	1	1	1	1
	MC_3	$T_1T_2T_3$	2nd operand Address	m	0	1	0	1	1	1	1
	MC_4	Ti	•	Z	1	1	1	1	1	1	1
	MC_5	$T_1T_2T_3$	mn	L	1	0	0	1	1	1	1
	MC_5	$T_1T_2T_3$	mn+1	H	1	0	0	1	1	1	1

(to be continued)

Instruction	Machine Cycle	States	ADDRESS	DATA	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{LIR}$	$\overline{HALT}$	ST
LD (mn),ww	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	1st operand Address	n	0	1	0	1	1	1	1
	MC_4	$T_1T_2T_3$	2nd operand Address	m	0	1	0	1	1	1	1
	MC_5	T_i	•	Z	1	1	1	1	1	1	1
	MC_6	$T_1T_2T_3$	mn	wwL	1	0	0	1	1	1	1.
	MC_7	$T_1T_2T_3$	mn+1	wwH	1	0	0	1	1	1	1
LD (mn),IX LD (mn),IY	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	1st operand Address	n	0	1	0	1	1	1	1
	MC_4	$T_1T_2T_3$	2nd operand Address	m	0	1	0	1	1	1	1
	MC_5	T_i	•	Z	1	1	1	1	1	1	1
	MC_6	$T_1T_2T_3$	mn	IXL IYL	1	0	0	1	1	1	1
	MC_7	$T_1T_2T_3$	mn+1	IXH IYH	1	0	0	1	1	1	1
LD SP, HL	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	T_i	•	Z	1	1	1	1	1	1	1
LD SP,IX LD SP,IY	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC_3	T_i	•	Z	1	1	1	1	1	1	1
LDI LDD	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	HL	DATA	0	1	0	1	1	1	1
	MC_4	$T_1T_2T_3$	DE	DATA	1	0	0	1	1	1	1

(to be continued)

Instruction	Machine Cycle	States	ADDRESS	DATA	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{LIR}$	$\overline{HALT}$	ST
LDIR LDDR (If $BC_R \neq 0$)	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	HL	DATA	0	1	0	1	1	1	1
	MC_4	$T_1T_2T_3$	DE	DATA	1	0	0	1	1	1	1
	MC_5 $\sim MC_6$	TiTi	•	Z	1	1	1	1	1	1	1
LDIR LDDR (If $BC_R = 0$)	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	HL	DATA	0	1	0	1	1	1	1
	MC_4	$T_1T_2T_3$	DE	DATA	1	0	0	1	1	1	1
MLT ww	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC_3 $\sim MC_{13}$	TiTiTiTi TiTiTiTi TiTiTi	•	Z	1	1	1	1	1	1	1
NEG	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
NOP	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
OUT (m),A	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	1st operand Address	m	0	1	0	1	1	1	1
	MC_3	Ti	•	Z	1	1	1	1	1	1	1
	MC_4	$T_1T_2T_3$	m to $A_0 \sim A_7$ A to $A_8 \sim A_{15}$	A	1	0	1	0	1	1	1

(to be continued)

Instruction	Machine Cycle	States	ADDRESS	DATA	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{LIR}$	$\overline{HALT}$	ST
OUT (C),g	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC_3	T_I	•	Z	1	1	1	1	1	1	1
	MC_4	$T_1T_2T_3$	BC	g	1	0	1	0	1	1	1
OUTO (m),g	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	1st operand Address	m	0	1	0	1	1	1	1
	MC_4	T_I	•	Z	1	1	1	1	1	1	1
	MC_5	$T_1T_2T_3$	m to $A_0 \sim A_7$ 00H to $A_8 \sim A_{15}$	g	1	0	1	0	1	1	1
OTIM OTDM	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC_3	T_I	•	Z	1	1	1	1	1	1	1
	MC_4	$T_1T_2T_3$	HL	DATA	0	1	0	1	1	1	1
	MC_5	$T_1T_2T_3$	C to $A_0 \sim A_7$ 00H to $A_8 \sim A_{15}$	DATA	1	0	1	0	1	1	1
	MC_6	Ti	•	Z	1	1	1	1	1	1	1
OTIMR OTDMR (If Br≠0)	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC_3	Ti	•	Z	1	1	1	1	1	1	1
	MC_4	$T_1T_2T_3$	HL	DATA	0	1	0	1	1	1	1
	MC_5	$T_1T_2T_3$	C to $A_0 \sim A_7$ 00H to $A_8 \sim A_{15}$	DATA	1	0	1	0	1	1	1
	MC_6 $\sim MC_8$	$T_iT_iT_I$	•	Z	1	1	1	1	1	1	1

(to be continued)

Instruction	Machine Cycle	States	ADDRESS	DATA	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{LIR}$	$\overline{HALT}$	ST
OTIMR OTDMR (If Br=0)	MC$_1$	T$_1$T$_2$T$_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	T$_1$T$_2$T$_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC$_3$	T$_i$	•	Z	1	1	1	1	1	1	1
	MC$_4$	T$_1$T$_2$T$_3$	HL	DATA	0	1	0	1	1	1	1
	MC$_5$	T$_1$T$_2$T$_3$	C to A$_0$~A$_7$ 00H to A$_8$~A$_{15}$	DATA	1	0	1	0	1	1	1
	MC$_6$	T$_i$	•	Z	1	1	1	1	1	1	1
OUTI OUTD	MC$_1$	T$_1$T$_2$T$_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	T$_1$T$_2$T$_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC$_3$	T$_1$T$_2$T$_3$	HL	DATA	0	1	0	1	1	1	1
	MC$_4$	T$_1$T$_2$T$_3$	BC	DATA	1	0	1	0	1	1	1
OTIR OTDR (If Br≠0)	MC$_1$	T$_1$T$_2$T$_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	T$_1$T$_2$T$_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC$_3$	T$_1$T$_2$T$_3$	HL	DATA	0	1	0	1	1	1	1
	MC$_4$	T$_1$T$_2$T$_3$	BC	DATA	1	0	1	0	1	1	1
	MC$_5$ ~MC$_6$	T$_i$T$_i$	•	Z	1	1	1	1	1	1	1
OTIR OTDR (If Br=0)	MC$_1$	T$_1$T$_2$T$_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	T$_1$T$_2$T$_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC$_3$	T$_1$T$_2$T$_3$	HL	DATA	0	1	0	1	1	1	1
	MC$_4$	T$_1$T$_2$T$_3$	BC	DATA	1	0	1	0	1	1	1
POP zz	MC$_1$	T$_1$T$_2$T$_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	T$_1$T$_2$T$_3$	SP	DATA	0	1	0	1	1	1	1
	MC$_3$	T$_1$T$_2$T$_3$	SP+1	DATA	0	1	0	1	1	1	1
POP IX POP IY	MC$_1$	T$_1$T$_2$T$_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0

(to be continued)

Instruction	Machine Cycle	States	ADDRESS	DATA	$\overline{\text{RD}}$	$\overline{\text{WR}}$	$\overline{\text{ME}}$	$\overline{\text{IOE}}$	$\overline{\text{LIR}}$	$\overline{\text{HALT}}$	ST
POP IX POP IY	MC$_2$	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	• 1	1
	MC$_3$	$T_1T_2T_3$	SP	DATA	0	1	0	1	1	1	1
	MC$_4$	$T_1T_2T_3$	SP+1	DATA	0	1	0	1	1	1	1
PUSH zz	MC$_1$	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC$_2$ ~MC$_3$	T_iT_i	•	Z	1	1	1	1	1	1	1
	MC$_4$	$T_1T_2T_3$	SP−1	zzH	1	0	0	1	1	1	1
	MC$_5$	$T_1T_2T_3$	SP−2	zzL	1	0	0	1	1	1	1
PUSH IX PUSH IY	MC$_1$	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC$_3$ ~MC$_4$	T_iT_i	•	Z	1	1	1	1	1	1	1
	MC$_5$	$T_1T_2T_3$	SP−1	IXH IYH	1	0	0	1	1	1	1
	MC$_6$	$T_1T_2T_3$	SP−2	IXL IYL	1	0	0	1	1	1	1
RET	MC$_1$	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	$T_1T_2T_3$	SP	DATA	0	1	0	1	1	1	1
	MC$_3$	$T_1T_2T_3$	SP+1	DATA	0	1	0	1	1	1	1
RET f (If condition is false)	MC$_1$	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC$_2$ ~MC$_3$	T_iT_i	•	Z	1	1	1	1	1	1	1
RET f (If condition is true)	MC$_1$	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	T_i	•	Z	1	1	1	1	1	1	1
	MC$_3$	$T_1T_2T_3$	SP	DATA	0	1	0	1	1	1	1
	MC$_4$	$T_1T_2T_3$	SP+1	DATA	0	1	0	1	1	1	1
RETI RETN	MC$_1$	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1

(to be continued)

Instruction	Machine Cycle	States	ADDRESS	DATA	$\overline{\text{RD}}$	$\overline{\text{WR}}$	$\overline{\text{ME}}$	$\overline{\text{IOE}}$	$\overline{\text{LIR}}$	$\overline{\text{HALT}}$	ST
RETI RETN	MC_3	$T_1T_2T_3$	SP	DATA	0	1	0	1	1	1	1
	MC_4	$T_1T_2T_3$	SP+1	DATA	0	1	0	1	1	1	1
RLCA RLA RRCA RRA	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
RLC g RL g RRC g RR g SLA g SRA g SRL g	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC_3	Ti	•	Z	1	1	1	1	1	1	1
RLC (HL) RL (HL) RRC (HL) RR (HL) SLA (HL) SRA (HL) SRL (HL)	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	HL	DATA	0	1	0	1	1	1	1
	MC_4	Ti	•	Z	1	1	1	1	1	1	1
	MC_5	$T_1T_2T_3$	HL	DATA	1	0	0	1	1	1	1
RLC (IX+d) RLC (IY+d) RL (IX+d) RL (IY+d) RRC (IX+d) RRC (IY+d) RR (IX+d) RR (IY+d) SLA (IX+d) SLA (IY+d) SRA (IX+d) SRA (IY+d) SRL (IX+d) SRL (IY+d)	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	1st operand Address	d	0	1	0	1	1	1	1
	MC_4	$T_1T_2T_3$	3rd op-code Address	3rd op-code	0	1	0	1	0	1	1
	MC_5	$T_1T_2T_3$	IX+d IY+d	DATA	0	1	0	1	1	1	1
	MC_6	Ti	•	Z	1	1	1	1	1	1	1
	MC_7	$T_1T_2T_3$	IX+d IY+d	DATA	1	0	0	1	1	1	1
RLD RRD	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	HL	DATA	0	1	0	1	1	1	1

(to be continued)

Instruction	Machine Cycle	States	ADDRESS	DATA	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{LIR}$	$\overline{HALT}$	ST
RLD RRD	MC_4 ~MC_7	TiTiTiTi	•	Z	1	1	1	1	1	1	1
	MC_8	$T_1T_2T_3$	HL	DATA	1	0	0	1	1	1	1
RST v	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2 ~MC_3	TiTi	•	Z	1	1	1	1	1	1	1
	MC_4	$T_1T_2T_3$	SP−1	PCH	1	0	0	1	1	1	1
	MC_5	$T_1T_2T_3$	SP−2	PCL	1	0	0	1	1	1	1
SCF	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
SET b,g RES b,g	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC_3	Ti	•	Z	1	1	1	1	1	1	1
SET b, (HL) RES b, (HL)	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	HL	DATA	0	1	0	1	1	1	1
	MC_4	Ti	•	Z	1	1	1	1	1	1	1
	MC_5	$T_1T_2T_3$	HL	DATA	1	0	0	1	1	1	1

SET b, (IX+d)
SET b, (IY+d)
RES b, (IX+d)
RES b, (IY+d)
	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0

	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	1st operand Address	d	0	1	0	1	1	1	1
	MC_4	$T_1T_2T_3$	3rd op-code Address	3rd op-code	0	1	0	1	0	1	1
	MC_5	$T_1T_2T_3$	IX+d IY+d	DATA	0	1	0	1	1	1	1
	MC_6	Ti	•	Z	1	1	1	1	1	1	1
	MC_7	$T_1T_2T_3$	IX+d IY+d	DATA	1	0	0	1	1	1	1

(to be continued)

Instruction	Machine Cycle	States	ADDRESS	DATA	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{LIR}$	$\overline{HALT}$	ST
SLP	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	——	———	7FFFFH	Z	1	1	1	1	1	0	1
TSTIO m	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	1st operand Address	m	0	1	0	1	1	1	1
	MC_4	$T_1T_2T_3$	C to $A_0 \sim A_7$ 00H to $A_8 \sim A_{15}$	DATA	0	1	1	0	1	1	1
TST g	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC_3	T_i	•	Z	1	1	1	1	1	1	1
TST m	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	1st operand Address	m	0	1	0	1	1	1	1
TST (HL)	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC_3	T_i	•	Z	1	1	1	1	1	1	1
	MC_4	$T_1T_2T_3$	HL	DATA	0	1	0	1	1	1	1

INTERRUPT

$\overline{NMI}$	MC_1	$T_1T_2T_3$	Next op-code Address (PC)		0	1	0	1	0	1	0
	MC_2 $\sim MC_3$	T_iT_i	•	Z	1	1	1	1	1	1	1
	MC_4	$T_1T_2T_3$	SP−1	PCH	1	0	0	1	1	1	1
	MC_5	$T_1T_2T_3$	SP−2	PCL	1	0	0	1	1	1	1
$\overline{INT_0}$ MODE 0 (RST INSERTED)	MC_1	$T_1T_2T_W$ T_WT_3	Next op-code Address (PC)	1st op-code	1	1	1	0	0	1	0
	MC_2 $\sim MC_3$	T_iT_i	•	Z	1	1	1	1	1	1	1

(to be continued)

Instruction	Machine Cycle	States	ADDRESS	DATA	$\overline{\text{RD}}$	$\overline{\text{WR}}$	$\overline{\text{ME}}$	$\overline{\text{IOE}}$	$\overline{\text{LIR}}$	$\overline{\text{HALT}}$	ST
$\overline{\text{INT}_0}$ MODE 0 (RST INSERTED)	MC_4	$T_1T_2T_3$	SP−1	PCH	1	0	0	1	1	1	1
	MC_5	$T_1T_2T_3$	SP−2	PCL	1	0	0	1	1	1	1
$\overline{\text{INT}_0}$ MODE 0 (CALL INSERTED)	MC_1	$T_1T_2T_W$ T_WT_3	Next op-code Address (PC)	1st op-code	1	1	1	0	0	1	0
	MC_2	$T_1T_2T_3$	PC	n	0	1	0	1	1	1	1
	MC_3	$T_1T_2T_3$	PC+1	m	0	1	0	1	1	1	1
	MC_4	T_i	•	Z	1	1	1	1	1	1	1
	MC_5	$T_1T_2T_3$	SP−1	PC+2(H)	1	0	0	1	1	1	1
	MC_6	$T_1T_2T_3$	SP−2	PC+2(L)	1	0	0	1	1	1	1
$\overline{\text{INT}_0}$ MODE 1	MC_1	$T_1T_2T_W$ T_WT_3	Next op-code Address (PC)		1	1	1	0	0	1	0
	MC_2	$T_1T_2T_3$	SP−1	PCH	1	0	0	1	1	1	1
	MC_3	$T_1T_2T_3$	SP−2	PCL	1	0	0	1	1	1	1
$\overline{\text{INT}_0}$ MODE 2	MC_1	$T_1T_2T_W$ T_WT_3	Next op-code Address (PC)	Vector	1	1	1	0	0	1	0
	MC_2	T_i	•	Z	1	1	1	1	1	1	1
	MC_3	$T_1T_2T_3$	SP−1	PCH	1	0	0	1	1	1	1
	MC_4	$T_1T_2T_3$	SP−2	PCL	1	0	0	1	1	1	1
	MC_5	$T_1T_2T_3$	I, Vector	DATA	0	1	0	1	1	1	1
	MC_6	$T_1T_2T_3$	I, Vector+1	DATA	0	1	0	1	1	1	1
$\overline{\text{INT}_1}$ $\overline{\text{INT}_2}$ Internal Interrupts	MC_1	$T_1T_2T_W$ T_WT_3	Next op-code Address (PC)		1	1	1	1	1	1	0
	MC_2	T_i	•	Z	1	1	1	1	1	1	1
	MC_3	$T_1T_2T_3$	SP−1	PCH	1	0	0	1	1	1	1
	MC_4	$T_1T_2T_3$	SP−2	PCL	1	0	0	1	1	1	1
	MC_5	$T_1T_2T_3$	I, Vector	DATA	0	1	0	1	1	1	1
	MC_6	$T_1T_2T_3$	I, Vector+1	DATA	0	1	0	1	1	1	1

23 REQUEST ACCEPTANCES IN EACH OPERATING MODE

Request \ Current Status		Normal Operation (CPU mode) (IOSTOP mode)	WAIT State	Refresh Cycle	Interrupt Acknowledge Cycle	DMA Cycle	BUS RELEASE mode	SLEEP mode	SYSTEM STOP mode
$\overline{\text{WAIT}}$		Acceptable	Acceptable	Not acceptable	Acceptable	Acceptable	Not acceptable	Not acceptable	Not acceptable
Refresh Request (Request of Refresh by the on-chip Refresh Controller)		Refresh cycle begins at the end of MC	Not acceptable	Not acceptable	Refresh cycle begins at the end of MC	Refresh cycle begins at the end of MC	Not acceptable	Not acceptable	Not acceptable
$\overline{\text{DREQ}}_0$ $\overline{\text{DREQ}}_1$		DMA cycle begins at the end of MC	DMA cycle begins at the end of MC	Acceptable * Refresh cycle precedes DMA cycle begins at the end of one MC	Acceptable DMA cycle begins at the end of MC	Acceptable Refer to "2 9 DMA Controller" for details	Acceptable * After BUS RELEASE cycle, DMA cycle begins at the end of one MC	Not acceptable	Not acceptable
$\overline{\text{BUSREQ}}$		Bus is released at the end of MC	Not acceptable	Not acceptable	Bus is released at the end of MC	Bus is released at the end of MC	Continue BUS RELEASE mode.	Acceptable	Acceptable
Interrupt	$\overline{\text{INT}}_0$, $\overline{\text{INT}}_1$, $\overline{\text{INT}}_2$	Accepted after executing the current instruction	Accepted after executing the current instruction	Not acceptable	Not acceptable	Not acceptable	Not acceptable	Acceptable Return from SLEEP mode to normal operation	Acceptable Return from SYSTEM STOP mode to normal operation
	Internal I/O Interrupt	↑	↑	↑	↑	↑	↑	↑	Not acceptable
	$\overline{\text{NMI}}$	↑	↑	↑	Not acceptable Interrupt acknowledge cycle precedes $\overline{\text{NMI}}$ is accepted after executing the next instruction	Acceptable DMA cycle stops	↑	↑	Acceptable Return from SYSTEM STOP mode to normal operation

NOTE)		
	*	not acceptable when DMA Request is in level sense
	↑	same as the above
	MC	Machine Cycle

24 REQUEST PRIORITY

The HD64180 has the following three types of requests.

Type 1.

To be accepted in specified state........... WAIT

Type 2.

To be accepted in each machine cycle...... Refresh Req.
DMA Req.
Bus Req.

Type 3.

To be accepted in each instruction......... Interrupt Req.

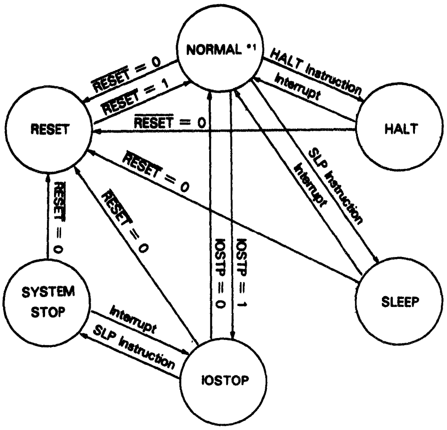
Type 1, Type 2, and Type 3 requests priority is shown as follows.
highest priority Type 1 > Type 2 > Type 3 lowest priority

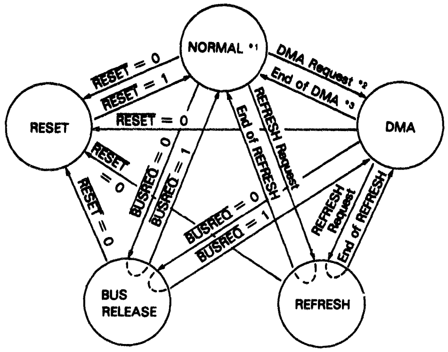
Each request priority in Type 2 is shown as follows.
highest priority Bus Req. > Refresh Req. > DMA Req. lowest priority

(NOTE) If Bus Req. and Refresh Req. occurs simultaneously, Bus Req. is accepted but Refresh Req. is cleared.

Refer to "2.7 Interrupts" for each request priority in Type 3.

25 OPERATION MODE TRANSITION





NOTE) *1 NORMAL· CPU executes instructions normally in NORMAL mode.
*2 DMA request· DMA is requested in the following cases
(1) $\overline{DREQ_0}$, $\overline{DREQ_1}$ = 0 (memory ←→ (memory mapped) I/O DMA transfer)
(2) DE0 = 1 (memory ←→ memory DMA transfer)
*3 DMA end: DMA ends in the following cases.
(1) $\overline{DREQ_0}$, $\overline{DREQ_1}$ = 1 (memory ←→ (memory mapped) I/O DMA transfer)
(2) BCR0, BCR1 = 0000H (all DMA transfers)
(3) $\overline{NMI}$ = 0 (all DMA transfers)

Other operation mode transitions

The following operation mode transitions are also possible.

1. $\text{HALT} \rightleftarrows \left\{\begin{array}{l}\text{DMA}\\ \text{REFRESH}\\ \text{BUS RELEASE}\end{array}\right\}$

   $\text{IOSTOP} \rightleftarrows \left\{\begin{array}{l}\text{DMA}\\ \text{REFRESH}\\ \text{BUS RELEASE}\end{array}\right\}$

2 SLEEP ⇄ BUS RELEASE

SYSTEM STOP ⇄ BUS RELEASE

26 STATUS SIGNALS

The following table shows pin outputs in each operating mode.

Mode		$\overline{\text{LIR}}$	$\overline{\text{ME}}$	$\overline{\text{IOE}}$	$\overline{\text{RD}}$	$\overline{\text{WR}}$	$\overline{\text{REF}}$	$\overline{\text{HALT}}$	$\overline{\text{BUSACK}}$	ST	Address BUS	Data BUS
CPU operation	Op-code Fetch (1st op-code)	0	0	1	0	1	1	1	1	0	A	IN
	Op-code Fetch (except 1st op-code)	0	0	1	0	1	1	1	1	1	A	IN
	Memory Read	1	0	1	0	1	1	1	1	1	A	IN
	Memory Write	1	0	1	1	0	1	1	1	1	A	OUT
	I/O Read	1	1	0	0	1	1	1	1	1	A	IN
	I/O Write	1	1	0	1	0	1	1	1	1	A	OUT
	Internal Operation	1	1	1	1	1	1	1	1	1	A	IN
Refresh		1	0	1	1	1	0	1	1	•	A	IN
Interrupt Acknowledge Cycle (1st machine cycle)	$\overline{\text{NMI}}$	0	0	1	0	1	1	1	1	0	A	IN
	$\overline{\text{INT}_0}$	0	1	0	1	1	1	1	1	0	A	IN
	$\overline{\text{INT}_1}$, $\overline{\text{INT}_2}$ & Internal Interrupts	1	1	1	1	1	1	1	1	0	A	IN
BUS RELEASE		1	Z	Z	Z	Z	1	1	0	•	Z	IN
HALT		0	0	1	0	1	1	0	1	0	A	IN
SLEEP		1	1	1	1	1	1	0	1	1	1	IN
Internal DMA	Memory Read	1	0	1	0	1	1	1	1	0	A	IN
	Memory Write	1	0	1	1	0	1	1	1	0	A	OUT
	I/O Read	1	1	0	0	1	1	1	1	0	A	IN
	I/O Write	1	1	0	1	0	1	1	1	0	A	OUT
RESET		1	1	1	1	1	1	1	1	1	Z	IN

NOTE)		
1	:	HIGH
0	:	LOW
A	:	Programmable
Z	:	High Impedance
IN	:	Input
OUT	.	Output
•	·	Invalid

27 PIN STATUS DURING RESET AND LOW POWER OPERATION MODES

Pin No.	Symbol	Pin function	Pin status in each operation mode			
			RESET	SLEEP	IOSTOP	SYSTEM STOP
4	$\overline{\text{WAIT}}$	–	IN (N)	IN (N)	IN (A)	IN (N)
5	$\overline{\text{BUSACK}}$	–	1	OUT	OUT	OUT
6	$\overline{\text{BUSREQ}}$	–	IN (N)	IN (A)	IN (A)	IN (A)
7	$\overline{\text{RESET}}$	–	0	IN (A)	IN (A)	IN (A)
8	$\overline{\text{NMI}}$	–	IN (N)	IN (A)	IN (A)	IN (A)
9	$\overline{\text{INT}_0}$	–	IN (N)	IN (A)	IN (A)	IN (A)
10	$\overline{\text{INT}_1}$	–	IN (N)	IN (A)	IN (A)	IN (A)
11	$\overline{\text{INT}_2}$	–	IN (N)	IN (A)	IN (A)	IN (A)
12	ST	–	1	1	OUT	1
13~30	A_0~A_{17}	–	Z	1	A	1
31	A_{18}/TOUT	A_{18}	Z	1	A	1
		TOUT	Z	OUT	H	H
34~41	D_0~D_7	–	Z	Z	A	Z
42	$\overline{\text{RTS}_0}$	–	1	H	OUT	H
43	$\overline{\text{CTS}_0}$	–	IN (N)	IN (A)	IN (N)	IN (N)
44	$\overline{\text{DCD}_0}$	–	IN (N)	IN (A)	IN (N)	IN (N)
45	TXA_0	–	1	OUT	H	H
46	RXA_0	–	IN (N)	IN (A)	IN (N)	IN (N)
47	$\text{CKA}_0/\overline{\text{DREQ}_0}$	CKA_0 (internal clock mode)	Z	OUT	Z	Z
		CKA_0 (external clock mode)	Z	IN (A)	IN (N)	IN (N)
		$\overline{\text{DREQ}_0}$	Z	IN (N)	IN (A)	IN (N)
48	TXA_1	–	1	OUT	H	H
49	RXA_1	–	IN (N)	IN (A)	IN (N)	IN (N)
50	$\text{CKA}_1/\overline{\text{TEND}_0}$	CKA_1 (internal clock mode)	Z	OUT	Z	Z
		CKA_1 (external clock mode)	Z	IN (A)	IN (N)	IN (N)
		$\overline{\text{TEND}_0}$	Z	1	OUT	1
51	TXS	–	1	OUT	H	H
52	$\text{RXS}/\overline{\text{CTS}_1}$	RXS	IN (N)	IN (A)	IN (N)	IN (N)
		$\overline{\text{CTS}_1}$	IN (N)	IN (A)	IN (N)	IN (N)
53	CKS	CKS (internal clock mode)	Z	OUT	1	1
		CKS (external clock mode)	Z	IN (A)	Z	Z
54	$\overline{\text{DREQ}_1}$	–	IN (N)	IN (N)	IN (A)	IN (N)
55	$\overline{\text{TEND}_1}$	–	1	1	OUT	1
56	$\overline{\text{HALT}}$	–	1	0	OUT	0
57	$\overline{\text{REF}}$	–	1	1	OUT	1
58	$\overline{\text{IOE}}$	–	1	1	OUT	1
59	$\overline{\text{ME}}$	–	1	1	OUT	1
60	E	–	0	E clock output	←	←
61	$\overline{\text{LIR}}$	–	1	1	OUT	1
62	$\overline{\text{WR}}$	–	1	1	OUT	1
63	$\overline{\text{RD}}$	–	1	1	OUT	1
64	ϕ	–	ϕ clock output	←	←	←

1: HIGH 0: LOW A: Programmable Z: High Impedance
IN (A): Input (Active) IN (N). Input (Not active) OUT: Output
H. Holds the previous state
←: same as the left

28 INTERNAL I/O REGISTERS

By programming IOA7 and IOA6 in the I/O control register, internal I/O register addresses are relocatable within ranges from 0000H to 00FFH in the I/O address space.

<table>
<tr><th>REGISTER</th><th>MNEMONICS</th><th>ADDRESS</th><th>REMARKS</th></tr>
<tr><td>ASCI Control Register A Channel 0</td><td>CNTLA0</td><td>0 0</td><td>
<table>
<tr><td>bit</td><td>MPE</td><td>RE</td><td>TE</td><td>$\overline{RTS0}$</td><td>MPBR/
EFR</td><td>MOD2</td><td>MOD1</td><td>MOD0</td></tr>
<tr><td>during RESET</td><td>0</td><td>0</td><td>0</td><td>1</td><td>invalid</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td></tr>
</table>
MODE Selection
Multi Processor Bit Receive/ Error Flag Reset
Request To Send
Transmit Enable
Receive Enable
Multi Processor Enable
</td></tr>
<tr><td>ASCI Control Register A Channel 1</td><td>CNTLA1</td><td>0 1</td><td>
<table>
<tr><td>bit</td><td>MPE</td><td>RE</td><td>TE</td><td>CKA1D</td><td>MPBR/
EFR</td><td>MOD2</td><td>MOD1</td><td>MOD0</td></tr>
<tr><td>during RESET</td><td>0</td><td>0</td><td>0</td><td>1</td><td>invalid</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td></tr>
</table>
MODE Selection
Multi Processor Bit Receive/ Error Flag Reset
CKA1 Disable
Transmit Enable
Receive Enable
Multi Processor Enable
<table>
<tr><td>MOD2, 1, 0</td><td></td></tr>
<tr><td>0 0 0</td><td>Start + 7 bit Data + 1 Stop</td></tr>
<tr><td>0 0 1</td><td>Start + 7 bit Data + 2 Stop</td></tr>
<tr><td>0 1 0</td><td>Start + 7 bit Data + Parity + 1 Stop</td></tr>
<tr><td>0 1 1</td><td>Start + 7 bit Data + Parity + 2 Stop</td></tr>
<tr><td>1 0 0</td><td>Start + 8 bit Data + 1 Stop</td></tr>
<tr><td>1 0 1</td><td>Start + 8 bit Data + 2 Stop</td></tr>
<tr><td>1 1 0</td><td>Start + 8 bit Data + Parity + 1 Stop</td></tr>
<tr><td>1 1 1</td><td>Start + 8 bit Data + Parity + 2 Stop</td></tr>
</table>
</td></tr>
<tr><td>ASCI Control Register B Channel 0</td><td>CNTLB0</td><td>0 2</td><td>
<table>
<tr><td>bit</td><td>MPBT</td><td>MP</td><td>$\overline{CTS}$/
PS</td><td>PEO</td><td>DR</td><td>SS2</td><td>SS1</td><td>SS0</td></tr>
<tr><td>during RESET</td><td>invalid</td><td>0</td><td>•</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td></tr>
<tr><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td></tr>
</table>
Clock Source and Speed Select
Divide Ratio
Parity Even or Odd
Clear To Send/Prescale
Multi Processor
Multi Processor Bit Transmit

• $\overline{CTS}$ · Depending on the condition of $\overline{CTS}$ Pin.
PS · Cleared to 0
</td></tr>
</table>

(to be continued)

REGISTER	MNEMONICS	ADDRESS	REMARKS
ASCI Control Register B Channel 1	· CNTLB1	0 3	

bit	MPBT	MP	$\overline{CTS}$/PS	PEO	DR	SS2	SS1	SS0
during RESET	invalid	0	0	0	0	1	1	1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

SS2, SS1, SS0 — Clock Source and Speed Select
DR — Divide Ratio
PEO — Parity Even or Odd
$\overline{CTS}$/PS — Clear To Send/Prescale
MP — Multi Processor
MPBT — Multi Processor Bit Transmit

General divide ratio	PS=0 (divide ratio=10)		PS=1 (divide ratio=30)	
SS2,1,0	DR=0 (×16)	DR=1 (×64)	DR=0 (×16)	DR=1 (×64)
000	$\phi\div$ 160	$\phi\div$ 640	$\phi\div$ 480	$\phi\div$ 1920
001	÷ 320	÷ 1280	÷ 960	÷ 3840
010	÷ 640	÷ 2560	÷ 1920	÷ 7680
011	÷ 1280	÷ 5120	÷ 3840	÷ 15360
100	÷ 2560	÷ 10240	÷ 7680	÷ 30720
101	÷ 5120	÷ 20480	÷ 15360	÷ 61440
110	÷ 10240	÷ 40960	÷ 30720	÷ 122880
111	External clock (frequency < ϕ ÷ 40)			

REGISTER	MNEMONICS	ADDRESS	REMARKS
ASCI Status Register Channel 0	STAT0	0 4	

bit	RDRF	OVRN	PE	FE	RIE	$\overline{DCD0}$	TDRE	TIE
during RESET	0	0	0	0	0	*	**	0
R/W	R	R	R	R	R/W	R	R	R/W

TIE — Transmit Interrupt Enable
TDRE — Transmit Data Register Empty
$\overline{DCD0}$ — Data Carrier Detect
RIE — Receive Interrupt Enable
FE — Framing Error
PE — Parity Error
OVRN — Over Run Error
RDRF — Receive Data Register Full

**

$\overline{CTS_0}$ Pin	TDRE
L	1
H	0

* $\overline{DCD_0}$: Depending on the condition of $\overline{DCD_0}$ Pin.

REGISTER	MNEMONICS	ADDRESS	REMARKS
ASCI Status Register Channel 1	: STAT1	0 5	

bit	RDRF	OVRN	PE	FE	RIE	CTS1E	TDRE	TIE
during RESET	0	0	0	0	0	0	1	0
R/W	R	R	R	R	R/W	R/W	R	R/W

TIE — Transmit Interrupt Enable
TDRE — Transmit Data Register Empty
CTS1E — $\overline{CTS1}$ Enable
RIE — Receive Interrupt Enable
FE — Framing Error
PE — Parity Error
OVRN — Over Run Error
RDRF — Receive Data Register Full

(to be continued)

<table>
<tr><th>REGISTER</th><th>MNEMONICS</th><th>ADDRESS</th><th>REMARKS</th></tr>
<tr><td>ASCI Transmit Data Register Channel 0</td><td>: TDR0</td><td>0 6</td><td></td></tr>
<tr><td>ASCI Transmit Data Register Channel 1</td><td>TDR1</td><td>0 7</td><td></td></tr>
<tr><td>ASCI Receive Data Register Channel 0</td><td>TSR0</td><td>0 8</td><td></td></tr>
<tr><td>ASCI Receive Data Register Channel 1</td><td>. TSR1</td><td>0 9</td><td></td></tr>
<tr><td>CSI/O Control Register</td><td>· CNTR</td><td>0 A</td><td>
<table>
<tr><td>bit</td><td>EF</td><td>EIE</td><td>RE</td><td>TE</td><td>–</td><td>SS2</td><td>SS1</td><td>SS0</td></tr>
<tr><td>during RESET</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td></tr>
<tr><td>R/W</td><td>R</td><td>R/W</td><td>R/W</td><td>R/W</td><td></td><td>R/W</td><td>R/W</td><td>R/W</td></tr>
</table>
Speed Select
Transmit Enable
Receive Enable
End Interrupt Enable
End Flag
<table>
<tr><th>SS2,1,0</th><th>Baud Rate</th><th>SS2,1,0</th><th>Baud Rate</th></tr>
<tr><td>000</td><td>φ÷ 20</td><td>100</td><td>φ÷ 320</td></tr>
<tr><td>001</td><td>÷ 40</td><td>101</td><td>÷ 640</td></tr>
<tr><td>010</td><td>÷ 80</td><td>110</td><td>÷1280</td></tr>
<tr><td>011</td><td>÷160</td><td>111</td><td>External
(frequency &lt; ÷ 20)</td></tr>
</table>
</td></tr>
<tr><td>CSI/O Transmit/Receive Data Register</td><td>TRDR</td><td>0 B</td><td></td></tr>
<tr><td>Timer Data Register Channel 0L</td><td>. TMDR0L</td><td>0 C</td><td></td></tr>
<tr><td>Timer Data Register Channel 0H</td><td>· TMDR0H</td><td>0 D</td><td></td></tr>
<tr><td>Timer Reload Register Channel 0L</td><td>· RLDR0L</td><td>0 E</td><td></td></tr>
<tr><td>Timer Reload Register Channel 0H</td><td>. RLDR0H</td><td>0 F</td><td></td></tr>
<tr><td>Timer Control Register</td><td>. TCR</td><td>1 0</td><td>
<table>
<tr><td>bit</td><td>TIF1</td><td>TIF0</td><td>TIE1</td><td>TIE0</td><td>TOC1</td><td>TOC0</td><td>TDE1</td><td>TDE0</td></tr>
<tr><td>during RESET</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>R/W</td><td>R</td><td>R</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td></tr>
</table>
Timer Down Count Enable 1,0
Timer Output Control 1,0
Timer Interrupt Enable 1,0
Timer Interrupt Flag 1,0
<table>
<tr><th>TOC1,0</th><th>A₁₈/TOUT</th></tr>
<tr><td>00</td><td>Inhibited</td></tr>
<tr><td>01</td><td>Toggle</td></tr>
<tr><td>10</td><td>0</td></tr>
<tr><td>11</td><td>1</td></tr>
</table>
</td></tr>
</table>

(to be continued)

<table>
<tr><th>REGISTER</th><th>MNEMONICS</th><th>ADDRESS</th><th>REMARKS</th></tr>
<tr><td>Timer Data Register Channel 1L</td><td>. TMDR1L</td><td>1 4</td><td></td></tr>
<tr><td>Timer Data Register Channel 1H</td><td>: TMDR1H</td><td>1 5</td><td></td></tr>
<tr><td>Timer Reload Register Channel 1L</td><td>: RLDR1L</td><td>1 6</td><td></td></tr>
<tr><td>Timer Reload Register Channel 1H</td><td>· RLDR1H</td><td>1 7</td><td></td></tr>
<tr><td>Free Running Counter</td><td>FRC</td><td>1 8</td><td>read only</td></tr>
<tr><td>DMA Source Address Register Channel 0L</td><td>SAR0L</td><td>2 0</td><td></td></tr>
<tr><td>DMA Source Address Register Channel 0H</td><td>· SAR0H</td><td>2 1</td><td></td></tr>
<tr><td>DMA Source Address Register Channel 0B</td><td>SAR0B</td><td>2 2</td><td>Bits 0-2 are used for SAR0B
<table>
<tr><th>A_{18}, A_{17}, A_{16}</th><th>DMA Transfer Request</th></tr>
<tr><td>X 0 0</td><td>$\overline{DREQ_0}$ (external)</td></tr>
<tr><td>X 0 1</td><td>RDR0 (ASCI0)</td></tr>
<tr><td>X 1 0</td><td>RDR1 (ASCI1)</td></tr>
<tr><td>X 1 1</td><td>Not Used</td></tr>
</table></td></tr>
<tr><td>DMA Destination Address Register Channel 0L</td><td>: DAR0L</td><td>2 3</td><td></td></tr>
<tr><td>DMA Destination Address Register Channel 0H</td><td>· DAR0H</td><td>2 4</td><td></td></tr>
<tr><td>DMA Destination Address Register Channel 0B</td><td>: DAR0B</td><td>2 5</td><td>Bits 0-2 are used for DAR0B.
<table>
<tr><th>A_{18}, A_{17}, A_{16}</th><th>DMA Transfer Request</th></tr>
<tr><td>X 0 0</td><td>$\overline{DREQ_0}$ (external)</td></tr>
<tr><td>X 0 1</td><td>TDR0 (ASCI0)</td></tr>
<tr><td>X 1 0</td><td>TDR1 (ASCI1)</td></tr>
<tr><td>X 1 1</td><td>Not Used</td></tr>
</table></td></tr>
<tr><td>DMA Byte Count Register Channel 0L</td><td>. BCR0L</td><td>2 6</td><td></td></tr>
<tr><td>DMA Byte Count Register Channel 0H</td><td>: BCR0H</td><td>2 7</td><td></td></tr>
<tr><td>DMA Memory Address Register Channel 1L</td><td>: MAR1L</td><td>2 8</td><td></td></tr>
<tr><td>DMA Memory Address Register Channel 1H</td><td>: MAR1H</td><td>2 9</td><td></td></tr>
<tr><td>DMA Memory Address Register Channel 1B</td><td>: MAR1B</td><td>2 A</td><td>Bits 0-2 are used for MAR1B</td></tr>
<tr><td>DMA I/O Address Register Channel 1L</td><td>: IAR1L</td><td>2 B</td><td></td></tr>
<tr><td>DMA I/O Address Register Channel 1H</td><td>IAR1H</td><td>2 C</td><td></td></tr>
</table>

(to be continued)

REGISTER	MNEMONICS	ADDRESS	REMARKS
DMA Byte Count Register Channel 1L	: BCR1L	2 E	
DMA Byte Count Register Channel 1H	: BCR1H	2 F	
DMA Status Register	: DSTAT	3 0	

bit	DE1	DE0	$\overline{\text{DWE1}}$	$\overline{\text{DWE0}}$	DIE1	DIE0	–	DME
during RESET	0	0	1	1	0	0	1	0
R/W	R/W	R/W	W	W	R/W	R/W		R

DMA Master Enable

DMA Interrupt Enable 1,0

DMA Enable Bit Write Enable 1,0

DMA Enable ch 1,0

REGISTER	MNEMONICS	ADDRESS	REMARKS
DMA Mode Register	: DMODE	3 1	

bit	–	–	DM1	DM0	SM1	SM0	MMOD	–
during RESET	1	1	0	0	0	0	0	1
R/W			R/W	R/W	R/W	R/W	R/W	

Memory MODE Select

Ch 0 Source Mode 1,0

Ch 0 Destination Mode 1, 0

DM1, 0	Destination	Address
0 0	M	DAR0+1
0 1	M	DAR0−1
1 0	M	DAR0 fixed
1 1	I/O	DAR0 fixed

SM1, 0	Source	Address
0 0	M	SAR0+1
0 1	M	SAR0−1
1 0	M	SAR0 fixed
1 1	I/O	SAR0 fixed

MMOD	Mode
0	Cycle Steal Mode
1	Burst Mode

(to be continued)

<table>
<tr><th>REGISTER</th><th>MNEMONICS</th><th>ADDRESS</th><th>REMARKS</th></tr>
<tr><td>DMA/WAIT Control Register</td><td>. DCNTL</td><td>3 2</td><td>
<table>
<tr><td>bit</td><td>MWI1</td><td>MWI0</td><td>IWI1</td><td>IWI0</td><td>DMS1</td><td>DMS0</td><td>DIM1</td><td>DIM0</td></tr>
<tr><td>during RESET</td><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td></tr>
</table>
DMA Ch 1
I/O Memory
Mode Select
$\overline{\text{DREQi}}$ Select, i = 1,0
I/O Wait Insertion
Memory Wait Insertion
<table>
<tr><th>MWI1,0</th><th>The number of wait states</th><th>IWI1,0</th><th>The number of wait states</th></tr>
<tr><td>0 0</td><td>0</td><td>0 0</td><td>0</td></tr>
<tr><td>0 1</td><td>1</td><td>0 1</td><td>2</td></tr>
<tr><td>1 0</td><td>2</td><td>1 0</td><td>3</td></tr>
<tr><td>1 1</td><td>3</td><td>1 1</td><td>4</td></tr>
</table>
<table>
<tr><th>DMSi</th><th>Sense</th></tr>
<tr><td>1</td><td>Edge sense</td></tr>
<tr><td>0</td><td>Level sense</td></tr>
</table>
<table>
<tr><th>DIM1,0</th><th>Transfer Mode</th><th colspan="2">Address Increment/Decrement</th></tr>
<tr><td>0 0</td><td>M→I/O</td><td>MAR1+1</td><td>IAR1 fixed</td></tr>
<tr><td>0 1</td><td>M→I/O</td><td>MAR1−1</td><td>IAR1 fixed</td></tr>
<tr><td>1 0</td><td>I/O→M</td><td>IAR1 fixed</td><td>MAR1+1</td></tr>
<tr><td>1 1</td><td>I/O→M</td><td>IAR1 fixed</td><td>MAR1−1</td></tr>
</table>
</td></tr>
<tr><td>Interrupt Vector Low Register</td><td>: IL</td><td>3 3</td><td>
<table>
<tr><td>bit</td><td>IL7</td><td>IL6</td><td>IL5</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td></tr>
<tr><td>during RESET</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td></td><td></td><td></td><td></td><td></td></tr>
</table>
Interrupt Vector Low
</td></tr>
<tr><td>INT/TRAP Control Register</td><td>. ITC</td><td>3 4</td><td>
<table>
<tr><td>bit</td><td>TRAP</td><td>UFO</td><td>–</td><td>–</td><td>–</td><td>ITE2</td><td>ITE1</td><td>ITE0</td></tr>
<tr><td>during RESET</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td><td>1</td></tr>
<tr><td>R/W</td><td>R/W</td><td>R</td><td></td><td></td><td></td><td>R/W</td><td>R/W</td><td>R/W</td></tr>
</table>
$\overline{\text{INT}}$ Enable 2,1,0
Undefined Fetch Object
TRAP
</td></tr>
<tr><td>Refresh Control Register</td><td>. RCR</td><td>3 6</td><td>
<table>
<tr><td>bit</td><td>REFE</td><td>REFW</td><td>–</td><td>–</td><td>–</td><td>–</td><td>CYC1</td><td>CYC0</td></tr>
<tr><td>during RESET</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td></tr>
<tr><td>R/W</td><td>R/W</td><td>R/W</td><td></td><td></td><td></td><td></td><td>R/W</td><td>R/W</td></tr>
</table>
Cycle Select
Refresh Wait State
Refresh Enable
<table>
<tr><th>CYC1,0</th><th>Interval of Refresh Cycle</th></tr>
<tr><td>0 0</td><td>10 States</td></tr>
<tr><td>0 1</td><td>20</td></tr>
<tr><td>1 0</td><td>40</td></tr>
<tr><td>1 1</td><td>80</td></tr>
</table>
</td></tr>
</table>

(to be continued)

<table>
<tr><th>REGISTER</th><th>MNEMONICS</th><th>ADDRESS</th><th>REMARKS</th></tr>
<tr><td>MMU Common Base Register</td><td>. CBR</td><td>3 8</td><td>
<table>
<tr><td>bit</td><td>—</td><td>CB6</td><td>CB5</td><td>CB4</td><td>CB3</td><td>CB2</td><td>CB1</td><td>CB0</td></tr>
<tr><td>during RESET</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td></tr>
</table>
CB6–CB0: MMU Common Base Register</td></tr>
<tr><td>MMU Bank Base Register</td><td>: BBR</td><td>3 9</td><td>
<table>
<tr><td>bit</td><td>—</td><td>BB6</td><td>BB5</td><td>BB4</td><td>BB3</td><td>BB2</td><td>BB1</td><td>BB0</td></tr>
<tr><td>during RESET</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td></tr>
</table>
BB6–BB0: MMU Bank Base Register</td></tr>
<tr><td>MMU Common/Bank Area Register</td><td>: CBAR</td><td>3 A</td><td>
<table>
<tr><td>bit</td><td>CA3</td><td>CA2</td><td>CA1</td><td>CA0</td><td>BA3</td><td>BA2</td><td>BA1</td><td>BA0</td></tr>
<tr><td>during RESET</td><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td></tr>
</table>
CA3–CA0: MMU Common Area Register
BA3–BA0: MMU Bank Area Register</td></tr>
<tr><td>I/O Control Register</td><td>ICR</td><td>3 F</td><td>
<table>
<tr><td>bit</td><td>IOA7</td><td>IOA6</td><td>IOSTP</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td></tr>
<tr><td>during RESET</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td></tr>
<tr><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td></td><td></td><td></td><td></td><td></td></tr>
</table>
IOSTP: I/O Stop
IOA7, IOA6: I/O Address</td></tr>
</table>

HD64180S
NPU (Network Processing Unit)

■ DESCRIPTION

The HD64180S, network processing unit (NPU), provides multipurpose high-speed communication control functions on a single LSI chip. The HD64180S offers high performance communication protocol processing, as well as user system application processing, at a low cost.

Built-in features, such as an 8-bit CPU, 2 serial I/O channels, and a direct memory access controller (DMAC, support high-speed data transfer by reducing communications overheads.

The HD64180S has a variety of applications. It can be used as a communication subsystem processor or as a controller in a distributed control system for industrial robots.

In addition, the HD64180S is designed to interface with existing communication chips and to be compatible with existing communication software. It can be used with virtually any kind of communication system.

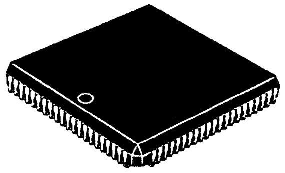
This manual describes HD64180S hardware. For details about programming instructions refer to the HD64180 Programming Manual (M21T038).

■ Overview

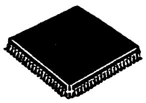
The HD64180S network processing unit (NPU) contains a 2-channel serial interface, 8-bit CPU, 2-channel direct memory access controller (DMAC) with a proprietary chained-block transfer function, timers, etc., all integrated on a single LSI chip. The HD64180S is thus well suited to multiprotocol communications processing.

The multiprotocol serial communications interface (MSCI) and the asynchronous serial communications interface/clocked serial I/O port (ASCI/CSIO) allow high speed data transfer using various communications protocols.

In particular, the MSCI is capable of handling asynchronous, byte synchronous, and bit synchronous communications protocols. Since the MSCI is connected to the on-chip DMAC, it is possible to realize high speed single-address DMA transfer (chained-block transfer) in frame units during bit synchronous communications. Furthermore, the flexible processing capability of the HD64180S's CPU ensures compatibility with a wide range of communications protocols.



(CP-84)



(FP-80A)

■ FEATURES

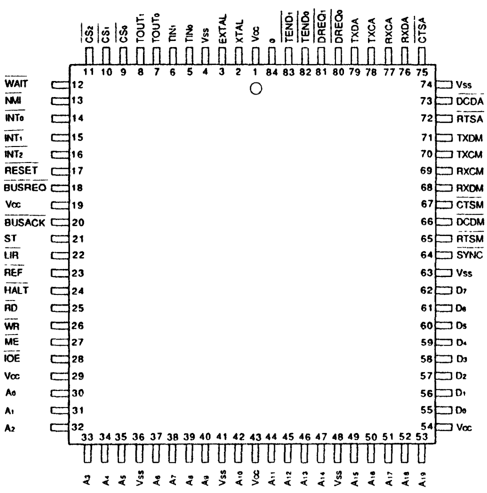
CPU	•Software-Compatible with HD64180Z • 80 type bus interface • On-chip MMU (1 Mbyte physical address space)
DMAC	• 2 channels • DMA transfer between memory and memory, memory and I/O (memory-mapped I/O), and memory and MSCI • Chained-block transfer between memory and MSCI • Internal interrupt requests available
Multiprotocol serial communications interface (MSCI)	• Full duplex channel • Asynchronous, byte synchronous (mono-, bi-, or external synchronous), or bit synchronous (HDLC or loop) selectable • Transmit/receive control using modem control signals ($\overline{RTSM}$, $\overline{CTSM}$, and $\overline{DCDM}$) • Internal Advanced Digital PLL (ADPLL) clock extraction receive data and/or receive clock noise suppression • On-chip baud rate generator • Internal interrupt requests available • Maximum transfer rate 7.1 Mbps (with 10 MHz clock)

Asynchronous serial communications interface/clocked serial I/O port (ASCI/CSIO)	• Full duplex channel • Asynchronous clocked serial mode (selectable) • Transmit/receive control using modem control signals ($\overline{RTSA}$, $\overline{CTSA}$, and $\overline{DCDA}$) • On-chip baud rate generator • Internal interrupt requests available
Timers	• 2 channels • 8-bit reloadable up-counter • Output waveform generator and external event count functions • Internal interrupt requests available
Interrupt controller	• Four external interrupt lines ($\overline{NMI}$, $\overline{INT_0}$, $\overline{INT_1}$, and $\overline{INT_2}$) • Fifteen internal interrupt sources
Memory access support function	Internal refresh controller • Internal wait state controller • Internal chip-select controller
Other functions	• On-chip clock oscillator circuit • Low power dissipation modes (sleep and system stop)

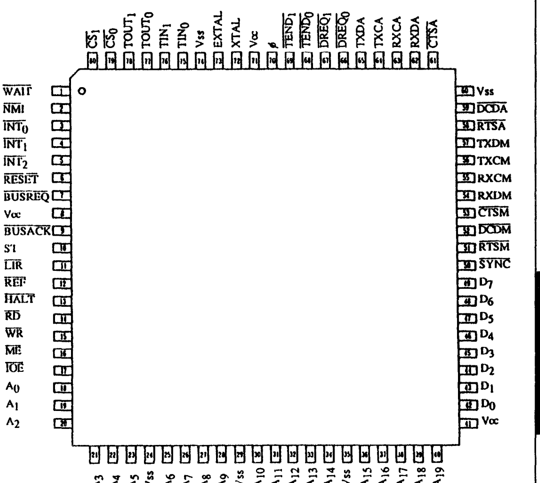
■ TYPE OF PRODUCTS

Product Name	Max. Operating Frequency	Package
HD64180SCP6	6.17 MHz	CP-84 (84-pin PLCC)
HD64180SCP8	8 MHz	
HD64180SCP10	10 MHz	
HD64180SH6	6.17 MHz	FP-80A (80-pin QFP)
HD64180SH8	8 MHz	
HD64180SH10	10 MHz	

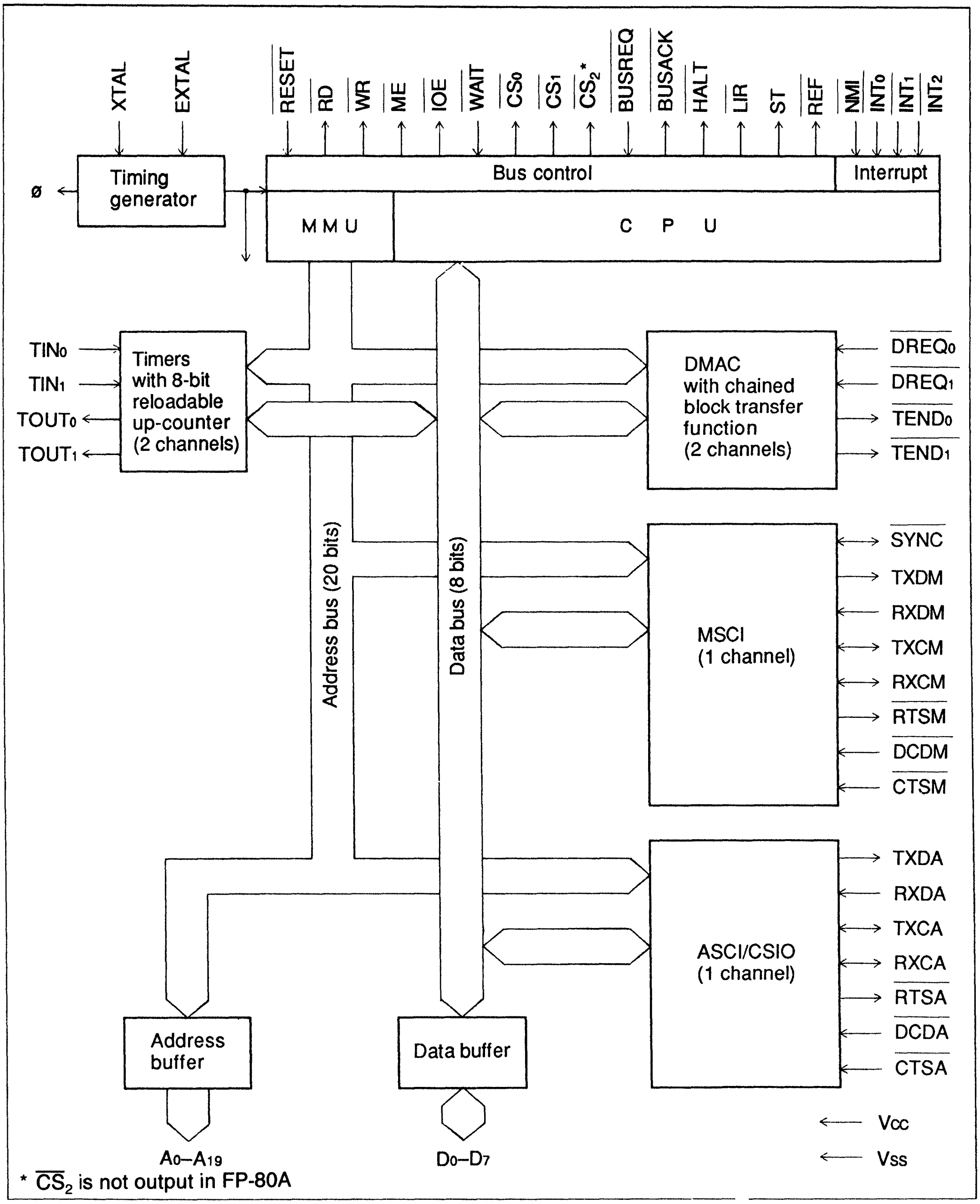
■ PIN ASSIGNMENT



Top View
(CP-84)



Top View
(FP-80A)



* $\overline{CS_2}$ is not output in FP-80A

Figure 1. Block Diagram of the HD64180S

■ Applications

• Position in Product Line

The HD64180S's on-chip CPU (software-compatible with the HD64180Z) is capable of processing both communications protocols and user application programs. If the on-chip CPU is programmed for use mainly as a communications processor, application processing can be carried out by another CPU. Figure 2 illustrates this concept.

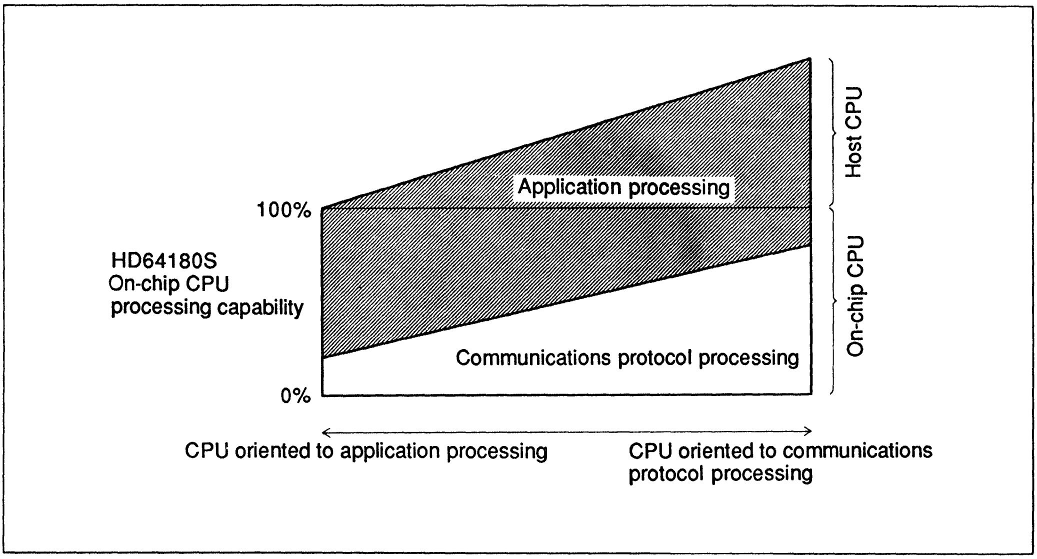

Application processing
100%
HD64180S
On-chip CPU
processing capability
Communications protocol processing
0%
Host CPU
On-chip CPU
CPU oriented to application processing
CPU oriented to communications protocol processing

Figure 2. Allocating CPU Processing Capability

For example, the HD64180S's CPU can be used mainly for communications protocol processing to provide various communications functions for a host CPU. This is suitable in situations requiring high-speed data transfer and/or complicated protocol processing. In this case, a flexible interface can be configured with the host CPU by selecting appropriate software and I/O devices.

On the other hand, the HD64180S's CPU can be used for application processing (i.e., when data transfer occurs infrequently and/or at low speeds). In this case, the MSCI, ASCI/CSIO, and DMAC in the HD64180S can process the communications data so as to reduce CPU overhead.
Thus the HD64180S can be used in a wide range of applications—from small-scale configurations containing two or three chips to large-scale configurations containing mass memory and numerous I/O devices.

■ Examples of System Configuration

(1) Data communications system

Figure 3 shows a system configured with a data communications subsystem. This system can be used for communications between computers in a public network or in an office automation (OA) system. system.

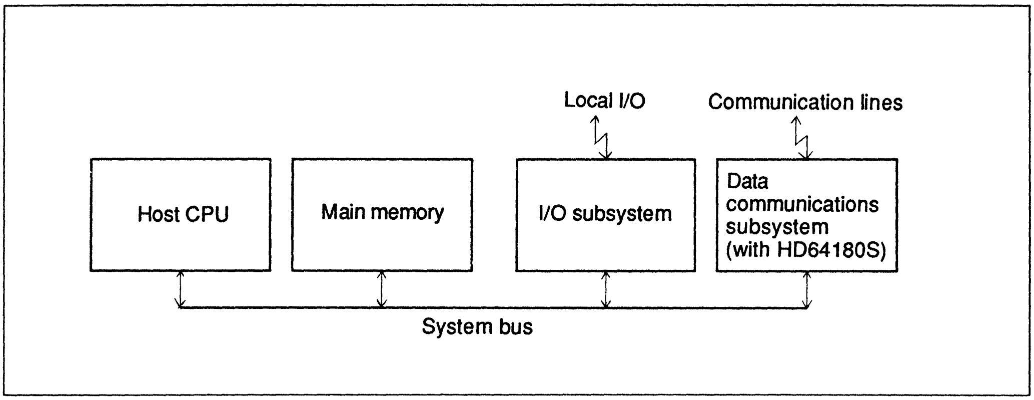


Figure 3. Example Configured with a Data Communications Subsystem

Figure 4 shows a minimum configuration example for the data communications subsystem shown in figure 3. In this configuration, the host CPU loads the HD64180S control program from main memory into the dual port RAM (DPRAM). The DPRAM has a transmit buffer, receive buffer, and communications data status area for interfacing between the host CPU and the HD64180S. Since the memory area allocated to this subsystem's communications program and transmit/receive buffers is relatively small, the subsystem is well suited for low-speed, simple communications protocols.

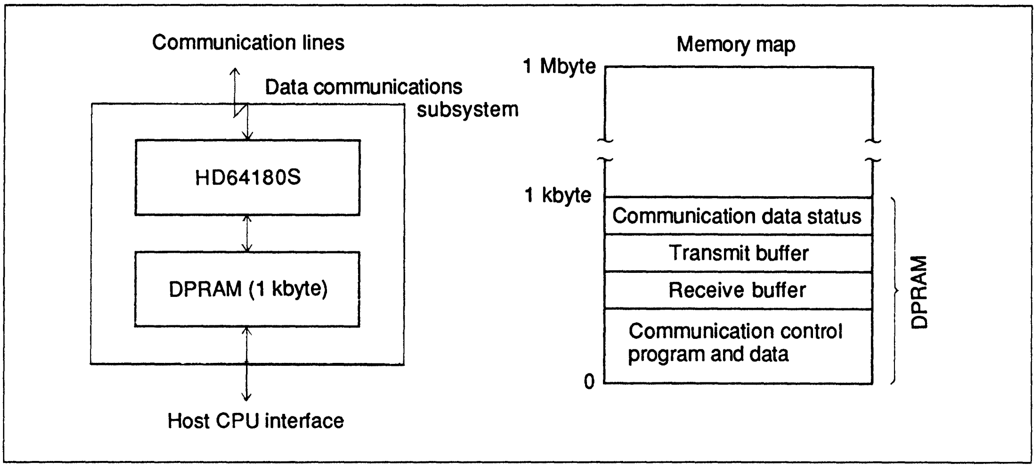


Figure 4. Example of Data Communications Subsystem (minimum configuration)

Figure 5 shows an extended communications subsystem for complex protocol processing and high-speed data transfer. This subsystem incorporates external memory and two stages of transmit/receive buffers. The HD64180S control program is loaded into external memory. This subsystem is easily realized because the HD64180S can directly access up to 1 Mbyte of memory using its 20-bit address bus.

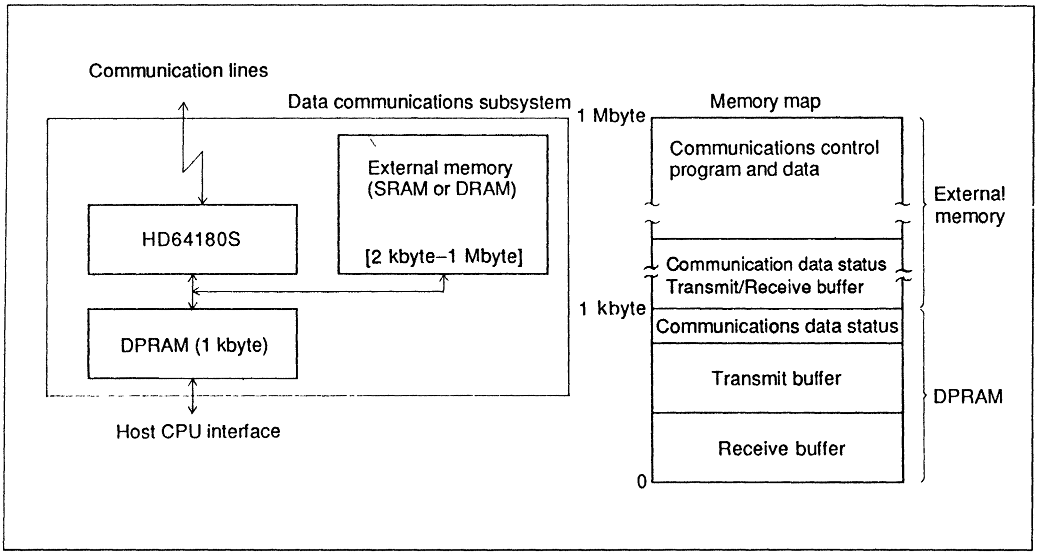


Figure 5. Example of Data Communications Subsystem (extended configuration)

(2) Distributed control system

Figure 6 shows an example in which the HD64180S is used as a distributed control device. This configuration can be used for controlling industrial machinery or for communicating between control devices of automobiles, OA systems, point-of-sales (POS) terminals, etc.

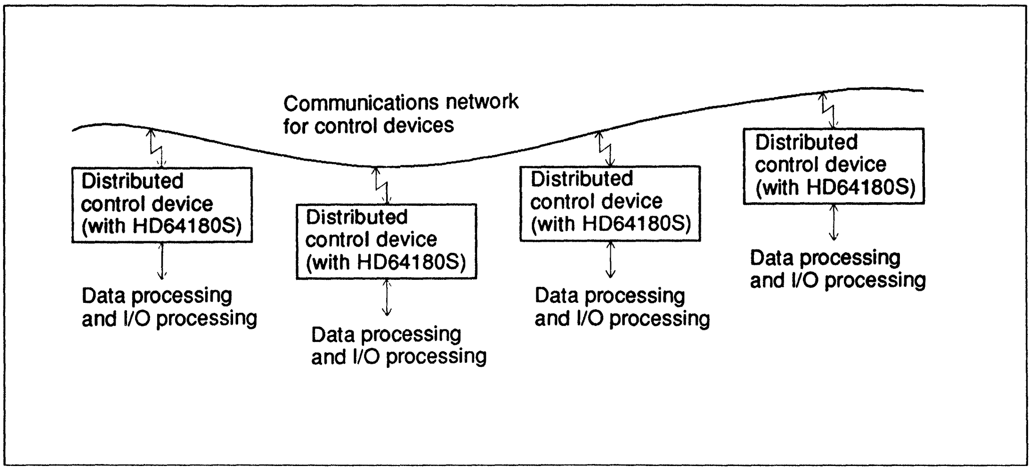

Communications network
for control devices
Distributed control device (with HD64180S)
Data processing and I/O processing
Distributed control device (with HD64180S)
Data processing and I/O processing
Distributed control device (with HD64180S)
Data processing and I/O processing
Distributed control device (with HD64180S)
Data processing and I/O processing

Figure 6. The HD64180S in a Distributed Control System

Figure 7 shows the internal configuration of the distributed control devices shown in figure 6. In this configuration, the HD64180S is directly connected to an I/O device, and the external memory (EPROM and RAM) contains the HD64180S control program and application programs. This simple system also allows high-speed data processing by providing direct access to up to 1 Mbyte of memory space including the data/stack and transmit/receive buffer areas.

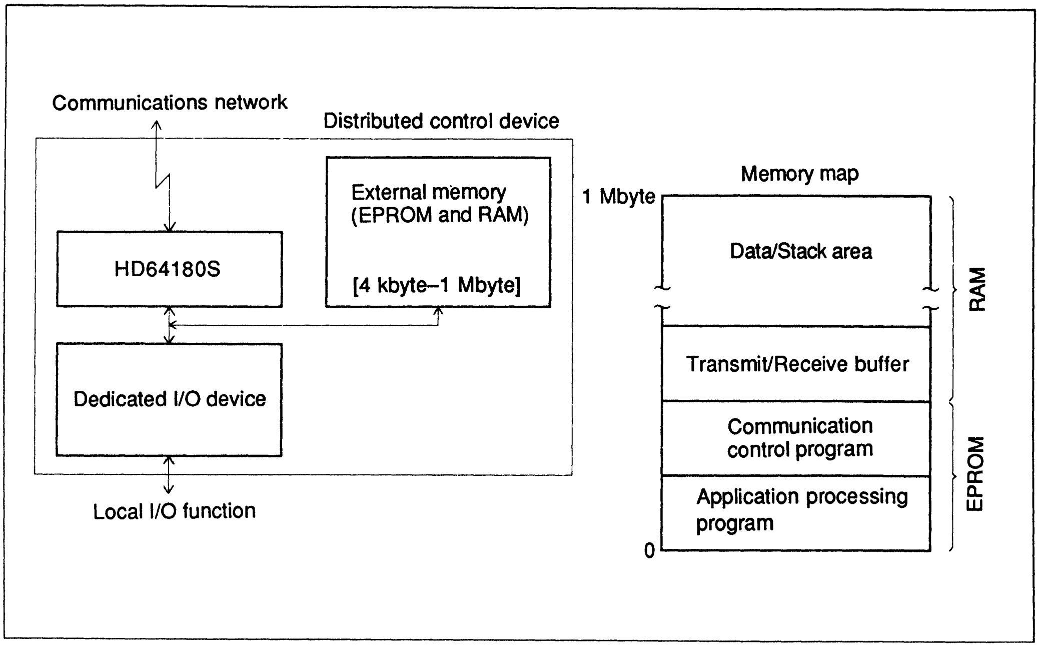


Figure 7. Internal Configuration of a Distributed Control Device Using the HD64180S

In the two configuration examples given above, the HD64180S is used either as a part of a data communications subsystem or as a distributed control device. In addition, the HD64180S can be used with various kinds of communications equipment.

Signal Descriptions

- **Power Supply**

Symbol	Pin Number CP-84	Pin Number FP-80A	Input/ Output	Remarks
V_{CC}	1, 19, 29, 43, 54	8, 41, 71	Input	+5V power supply: All V_{CC} pins must be connected to the +5V system power supply.
V_{SS}	4, 36, 41, 48, 63, 74	24, 29, 35, 60, 74	Input	Ground: All V_{SS} pins must be connected to the system ground.

Note: To minimize potential difference in the chip, use the shortest possible lead length to the Vcc and Vss pins.

- **Clock**

Symbol	Pin Number CP-84	Pin Number FP-80A	Input/ Output	Remarks
XTAL	2	72	Input	Crystal resonator input: The input frequency must be double that of the ø clock. When the EXTAL pin is connected to an external clock, the XTAL pin should be left floating.
EXTAL	3	73	Input	Crystal resonator or external clock input: The input frequency must be double that of the ø clock. Figures 8 and 9 show crystal resonator and external clock connection diagrams, respectively.
ø	84	70	Output	System clock: Supplies the ø clock to peripheral devices.

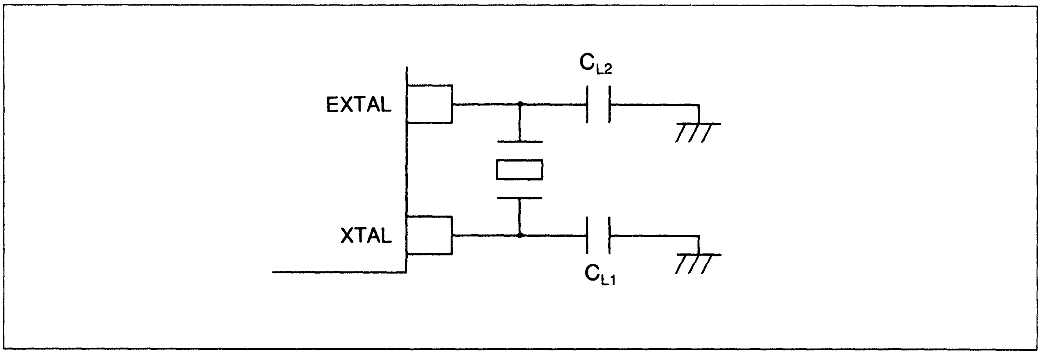


Figure 8. Example of Crystal Resonator Connection

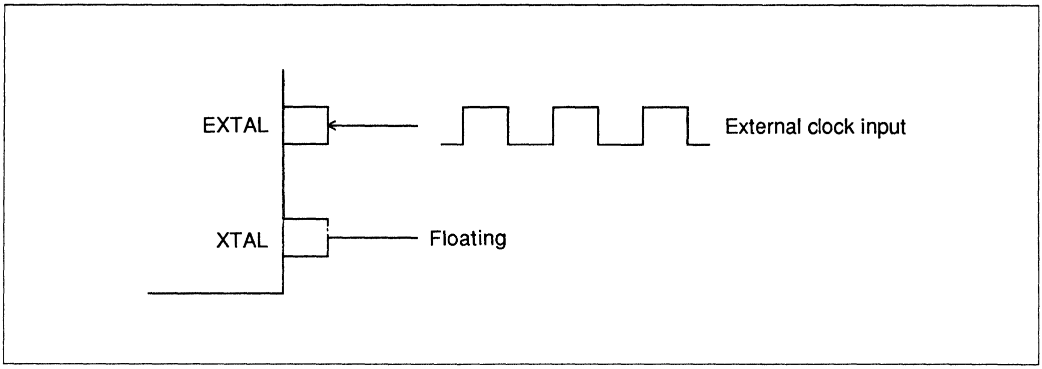


Figure 9. Example of External Clock Configuration

- **Reset Line**

	Pin Number		**Input/**	
Symbol	**CP-84**	**FP-80A**	**Output**	**Remarks**
$\overline{\text{RESET}}$	17	6	Input	Reset: When this line is driven active low for 6 or more clock cycles, the HD64180S enters the reset mode and all functions are reset.

- **Address Lines**

Symbol	Pin Number CP-84	Pin Number FP-80A	Input/ Output	Remarks
A_0–A_{19}	30–35, 37–40, 42, 44–47, 49–53	18–23, 25–28, 30–34, 36-40	Output (Three State)	Address bus: This 20-bit address bus supports 1Mbyte of memory and a 64kbyte (16-bit address width) I/O space. The address bus goes to high impedance during: • Reset mode • Passing control of the bus to another device (the HD64180S is placed in the bus release mode when the $\overline{\text{BUSREQ}}$ line is asserted).

- **Data Lines**

Symbol	Pin Number CP-84	Pin Number FP-80A	Input/ Output	Remarks
D_0–D_7	55–62	42–49	Input/ Output (Three State)	Data bus: The 8-bit handles bi-directional data passing (input and output of data.)

- **Memory and I/O Interface Lines**

Symbol	Pin Number CP-84	Pin Number FP-80A	Input/ Output	Remarks
$\overline{\text{RD}}$	25	14	Output (Three State)	Read: This line is asserted during read cycles. When this line is driven active low, the data lines are used as inputs.
$\overline{\text{WR}}$	26	15	Output (Three State)	Write: This line is asserted during write cycles. When this line is driven active low, the data lines are used as outputs.

	Pin			
	Number		Input/	
Symbol	CP-84	FP-80A	Output	Remarks
$\overline{ME}$	27	16	Output (Three State)	Memory enable: This line is used to indicate a memory read or write operation. It is asserted in the following cases: • Instruction fetch, operand read, and memory read/write instructions • Memory access during DMA cycles • Refresh cycles
$\overline{IOE}$	28	17	Output (Three State)	I/O enable: This line is used to indicate an I/O read/write operation. It is asserted in the following cases: • I/O read/write instructions • I/O access during DMA cycles • $\overline{INT_0}$ acknowledge cycles
$\overline{WAIT}$	12	1	Input	Wait: This line is used to extend either memory or I/O read/write cycles. If this line is low at the falling edge of a T_2 state, a Tw state is inserted. If the line is still low at the falling edge of the inserted Tw state, an additional Tw state is inserted. This process is repeated until the signal level on this line is high at the falling edge.
$\overline{CS_0}$	9	79	Output	Chip select: These lines are used to access one of the three physical address areas: PAL, PAM, and PAH. The partition of the physical address space is the same as that of wait controllers.
$\overline{CS_1}$	10	80	Output	
$\overline{CS_2}$	11	—	Output	

	Physical address area accessed	Signal asserted
1	PAL area (lower physical address area)	$\overline{CS_0}$
2	PAM area (middle physical address area)	$\overline{CS_1}$
3	PAH area (upper physical address area)	$\overline{CS_2}$

- **System Control Lines**

	Pin Number			
Symbol	**CP-84**	**FP-80A**	**Input/ Output**	**Remarks**
$\overline{\text{BUSREQ}}$	18	7	Input	Bus request: This line is asserted by an external device to request control of the bus. When this line is driven active low, the internal bus master waits until the end of the current machine cycle, then places the address lines, the data lines, and some of the memory I/O interface lines ($\overline{\text{RD}}$, $\overline{\text{WR}}$, $\overline{\text{ME}}$, and $\overline{\text{IOE}}$) into the high impedance state.
$\overline{\text{BUSACK}}$	20	9	Output	Bus acknowledge: This line is used by the internal bus master to notify an external device by sending a $\overline{\text{BUSACK}}$ signal that a $\overline{\text{BUSREQ}}$ signal has been received and the bus has been released.
$\overline{\text{HALT}}$	24	13	Output	HALT: This line is asserted whenever a HALT or SLP instruction is executed. It indicates that the HD64180S is in the halt, sleep, or system stop mode. This line is also used in conjunction with the $\overline{\text{LIR}}$ and ST lines to indicate the status of the CPU and internal DMAC.
$\overline{\text{LIR}}$	22	11	Output	Load instruction register: This line is asserted during opcode fetch cycles. This line can also be used to output the Z80 peripheral LSI interface signal.
ST	21	10	Output	Status: This line is used, together with $\overline{\text{LIR}}$ and $\overline{\text{HALT}}$, to indicate the internal status of the HD64180S (see table).

	$\overline{\text{HALT}}$	$\overline{\text{LIR}}$	ST	Status
(1)	1	0[*1] 1	0	CPU active (first byte of an opcode fetch)

[*1] The upper value shows the $\overline{\text{LIR}}$ pin status when the LIRE bit of the operation mode control register is 1, and the lower value shows the $\overline{\text{LIR}}$ pin status when the LIRE bit is 0.

	Pin							
	Number		Input/					
Symbol	CP-84	FP-80A	Output	Remarks				
				(2)	1	0*2 1	1	CPU active (second or third byte of an opcode fetch)
				(3)	X*1	1	0	DMAC operation
				(4)	1	1	1	Normal operating mode (other than (1), (2), or (3)) Reset mode
				(5)	0	0*2 1	0	Opcode fetch during halt mode (no instructions are executed)
				(6)	0	1	1	Halt mode (other than (3) or (5)) Sleep mode (other than (3)) System stop mode
$\overline{\text{REF}}$	23	12	Output	Refresh: This line is asserted during the DRAM refresh cycle. During this cycle, the refresh address is output on the 12 low-order lines (A_0 – A_{11}) of the address bus.				

*1 X: Don't care

*2 The upper value shows the $\overline{\text{LIR}}$ pin status when the LIRE bit of the operation mode control register is 1, and the lower value shows the $\overline{\text{LIR}}$ pin status when the LIRE bit is 0.

- **Interrupt Lines**

	Pin			
	Number		Input/	
Symbol	CP-84	FP-80A	Output	Remarks
$\overline{\text{NMI}}$	13	2	Input	Non-maskable interrupt: This line is used to request a non-maskable interrupt.

<table>
<tr><th></th><th colspan="2">Pin
Number</th><th>Input/</th><th></th></tr>
<tr><th>Symbol</th><th>CP-84</th><th>FP-80A</th><th>Output</th><th>Remarks</th></tr>
<tr><td>$\overline{\text{INT}_0}$</td><td>14</td><td>3</td><td>Input</td><td>Interrupt 0: This line is used to request a level-0 maskable interrupt. There are three different modes for level-0 interrupts (see table).
<table>
<tr><th>Mode</th><th>Function</th></tr>
<tr><td>0</td><td>Executing the instruction on the data bus</td></tr>
<tr><td>1</td><td>Executing the instruction at address 0038H</td></tr>
<tr><td>2</td><td>Vector mode</td></tr>
</table></td></tr>
<tr><td>$\overline{\text{INT}_1}$</td><td>15</td><td>4</td><td>Input</td><td rowspan="2">Interrupt 1 and 2: These lines are used respectively to request level-1 and level-2 maskable interrupts (vector mode).</td></tr>
<tr><td>$\overline{\text{INT}_2}$</td><td>16</td><td>5</td><td>Input</td></tr>
</table>

- **DMA Lines**

	Pin Number		Input/	
Symbol	**CI'-84**	**FP-80A**	**Output**	**Remarks**
$\overline{\text{DREQ}_0}$	80	66	Input	DMA request for channel 0: This line is used to request a DMA transfer using internal DMAC channel 0.
$\overline{\text{DREQ}_1}$	81	67	Input	DMA request for channel 1: This line is used to request a DMA transfer using internal DMAC channel 1.
$\overline{\text{TEND}_0}$	82	68	Output	Transfer end for channel 0: This line is used to indicate the end of a DMA transfer using internal DMAC channel 0. It is asserted synchronously with the read cycle upon the last data transfer.
$\overline{\text{TEND}_1}$	83	69	Output	Transfer end for channel 1: This line is used to indicate the end of a DMA transfer using internal DMAC channel 1. It is asserted synchronously with the read cycle upon the last data transfer.

- **Serial I/O (MSCI) Lines**

	Pin Number		**Input/**	
Symbol	**CP-84**	**FP-80A**	**Output**	**Remarks**
TXDM	71	57	Output	Transmit data from the MSCI: This line is used to output transmit data from the MSCI.
RXDM	68	54	Input	Receive data to the MSCI: This line is used to input receive data to the MSCI.
TXCM	70	56	Input/ Output	Transmit clock for the MSCI: This line is used to input/output the MSCI transmit clock. Three programmable modes: Input: • External transmit clock Output: • Transmit clock from the on-chip baud rate generator • Receive clock (used as the transmit clock)
RXCM	69	55	Input/ Output	Receive clock for the MSCI: This line is used to input/output the MSCI receive clock. This line can also be used to input the ADPLL operating clock. Four programmable modes: Input: • External receive clock • ADPLL operating clock Output: • Receive clock extracted by the ADPLL (when the on-chip baud rate generator is used as the ADPLL operating clock) • Receive clock from the on-chip baud rate generator
$\overline{\text{RTSM}}$	65	51	Output	Request to send for the MSCI: Indicates that the HD64180S has data to be output to a communications device such as modem. The output level can be automatically controlled by MSCI operation (auto-enable function). This line can also be used as a general purpose output port.
$\overline{\text{DCDM}}$	66	52	Input	Data carrier detect for the MSCI: Indicates that a communications device such as modem is receiving valid data from the communications line. MSCI receive operation can be automatically controlled by this input (auto-enable function). This line can also be used as a general purpose input port.

Symbol	Pin Number CP-84	FP-80A	Input/ Output	Remarks
$\overline{\text{CTSM}}$	67	53	Input	Clear to send for the MSCI: Indicates that a communications device such as modem is ready to send data to the communications line. MSCI transmit operation can be automatically controlled by this input (auto-enable function). This line can also be used as a general purpose input port.
$\overline{\text{SYNC}}$	64	50	Input/ Output	Synchronization for the MSCI: This line is used as an input in the external byte synchronous mode. Synchronization is established at the falling edge of $\overline{\text{SYNC}}$. This line is used as an output in the byte sync (mono- or bi-) or HDLC mode. It indicates the inverse of the SYNCD/FLGD bit in MSCI status register 1 (MST1)*. In the asynchronous mode, this line is used as an input. The input value does not affect operation.

*For details concerning MSCI status register 1 (MST1), see "MSCI Status Register 1."

- **Serial I/O (ASCI/CSIO) Lines**

Symbol	Pin Number CP-84	FP-80A	Input/ Output	Remarks
TXDA	79	65	Output	Transmit data from the ASCI/CSIO: This line is used to output transmit data from the ASCI/CSIO.
RXDA	76	62	Input	Receive data to the ASCI/CSIO: This line is used to input receive data to the ASCI/CSIO.
TXCA	78	64	Input/ Output	Transmit clock for the ASCI/CSIO: This line is used to input/output the ASCI/CSIO transmit clock. Two programmable modes: Input: • External transmit clock Output: • Transmit clock from the on-chip baud rate generator
RXCA	77	63	Input/ Output	Receive clock for the ASCI/CSIO: This line is used to input/output the ASCI/CSIO receive clock. Two programmable modes: Input: • External receive clock Output: • Receive clock from the on-chip baud rate generator

Symbol	Pin Number CP-84	FP-80A	Input/ Output	Remarks
$\overline{\text{RTSA}}$	72	58	Output	Request to send for ASCI/CSIO: Indicates that the HD64180S has data to be output to a communications device such as a modem. The output level can be automatically controlled by the ASCI/CSIO operation (auto-enable function). This line can also be used as a general purpose output port.
$\overline{\text{DCDA}}$	73	59	Input	Data carrier detect for ASCI/CSIO: Indicates that a communications device such as a modem is receiving valid signals from the communications line. ASCI/CSIO receive operation can be automatically controlled by this input (auto-enable function). This line can also be used as a general purpose input port.
$\overline{\text{CTSA}}$	75	61	Input	Clear to send for ASCI/CSIO: Indicates that a communications device such as modem is ready to send data to the communications line. ASCI/CSIO transmit operation can be controlled automatically by this input (auto-enable function). This line can also be used as a general purpose input port.

- **Timer Lines**

Symbol	Pin Number CP-84	FP-80A	Input/ Output	Remarks
TIN_0	5	75	Input	Timer inputs for channels 0 and 1: Event counter signals are input via these lines.
TIN_1	6	76	Input	
TOUT_0	7	77	Output	Timer outputs for channels 0 and 1: Timer signals are output via these lines.
TOUT_1	8	78	Output	

■ Absolute Maximum Ratings

Item	Symbol	Rating	Unit
Supply voltage	V_{cc}	–0.3 to +7.0	V
Input voltage	V_{in}	–0.3 to Vcc + 0.3	V
Operating temperature	T_{opr}	-20 to +75	°C
Storage temperature	T_{stg}	–55 to +150	°C

Caution: Permanent damage to the HD64180S may result if it is subjected to conditions that exceed the absolute maximum ratings. To assure normal operation, the following conditions should be satisfied:

$V_{ss} \leq V_{in} \leq V_{cc}$

■ Electrical Characteristics

- **DC Characteristics** (Vcc=5V ± 10%, Vss=0V, Ta=–20 to +75°C unless otherwise specified)

Item	Symbol	min	typ	max	Unit	Conditions
Input high level voltage at $\overline{\text{RESET}}$, EXTAL, and $\overline{\text{NMI}}$	V_{IH1}	V_{cc}–0.6	----	V_{cc}+0.3	V	
Input high level voltage at lines other than $\overline{\text{RESET}}$, EXTAL, and $\overline{\text{NMI}}$	V_{IH2}	2.2	----	V_{cc}+0.3	V	
Input low level voltage at $\overline{\text{RESET}}$, EXTAL, and $\overline{\text{NMI}}$	V_{IL1}	–0.3	----	0.6	V	
Input low level voltage at lines other than $\overline{\text{RESET}}$, EXTAL, and $\overline{\text{NMI}}$	V_{IL2}	–0.3	----	0.8	V	
Output high level voltage at all output lines	V_{OH}	2.4	----	----	V	I_{OH} = –200 µA
		Vcc – 1.2	----	----		I_{OH} = –20 µA
Output low level voltage at all output lines	V_{OL}	----	----	0.45	V	I_{OL} = 2.2 mA

DC Characteristics (cont.) (Vcc = 5V ± 10%, Vss = 0V, Ta = -20 to +75°C unless otherwise specified)

Item	Symbol	min	typ	max	Unit	Conditions
Input leakage current at all input lines other than XTAL and EXTAL	I_{IL}	----	----	1.0	µA	V_{in} = 0.5 to V_{CC} -0.5
Three state leakage current	I_{TL}	----	----	1.0	µA	V_{in} = 0.5 to V_{CC} -0.5
Current dissipation* (normal operation)	Icc	----	36	72	mA	f = 6 MHz
		----	48	96		f = 8 MHz
		----	60	120		f = 10 MHz
Current dissipation* (system stop mode)		----	6	12	mA	f = 6 MHz
		----	8	16		f = 8 MHz
		----	10	20		f = 10 MHz
Pin capacitance	Cp	----	----	20	pF	V_{in} = 0V, f = 1 MHz, Ta = 25°C

* Input signal
$\overline{\text{RESET}}$, EXTAL, $\overline{\text{NMI}}$: $V_{IHmin} = V_{CC}$-0.6V, V_{ILmax} = 0.6V the others: $V_{IHmin} = V_{CC}$-1.0V, V_{ILmax} = 0.8V

- **AC Characteristics** (Vcc = 5V ± 10%, Vss = 0V, Ta = -20 to +75°C unless otherwise specified)

Note that the specifications related to $\overline{\text{CS}}_2$ pin is specified only in CP-84 package version.

Bus Timing

Table .3-3. Bus Timing

		HD64180SCP6			HD64180SCP8			HD64180SCP10				
Item	Symbol	min	typ	max	min	typ	max	min	typ	max	Unit	Timing
Clock cycle time	t_{CYC}	162	–	2000	125	–	2000	100	–	2000	ns	See figures 10, 11, 12, and 13.
Clock high-level pulse width	t_{CHW}	65	–	–	50	–	–	38	–	–	ns	
Clock low-level pulse width	t_{CLW}	65	–	–	50	–	–	38	–	–	ns	
Clock fall time	t_{cf}	–	–	15	–	–	15	–	–	12	ns	
Clock rise time	t_{cr}	–	–	15	–	–	15	–	–	12	ns	
Address delay time	t_{AD}	–	–	90	–	–	80	–	–	55	ns	

Bus Timing (cont.)

		HD64180SCP6			HD64180SCP8			HD64180SCP10				
Item	**Symbol**	**min**	**typ**	**max**	**min**	**typ**	**max**	**min**	**typ**	**max**	**Unit**	**Timing**
Address set-up time (vis-a-vis falling edge of $\overline{\text{ME}}$, $\overline{\text{IOE}}$, or $\overline{\text{CS2}}$—$\overline{\text{CS0}}$)	t_{AS}	20	–	–	15	–	–	15	–	–	ns	See figures 10, 11, 12, and 13.
$\overline{\text{ME}}$ delay time 1	t_{MED1}	–	–	60	–	–	50	–	–	50	ns	
$\overline{\text{RD}}$ delay time 1	t_{RDD1}	–	–	60	–	–	50	–	–	50	ns	
$\overline{\text{LIR}}$ delay time 1	t_{LD1}	–	–	80	–	–	70	–	–	55	ns	
Address hold time (vis-a-vis rising edge of $\overline{\text{ME}}$, $\overline{\text{IOE}}$, $\overline{\text{RD}}$, $\overline{\text{WR}}$ or $\overline{\text{CS2}}$—$\overline{\text{CS0}}$)	t_{AH}	35	–	–	20	–	–	10	–	–	ns	
$\overline{\text{ME}}$ delay time 2	t_{MED2}	–	–	60	–	–	50	–	–	50	ns	
$\overline{\text{RD}}$ delay time 2	t_{RDD2}	–	–	60	–	–	50	–	–	50	ns	
$\overline{\text{RD}}$ delay time 3	t_{RDD3}	–	–	65	–	–	60	–	–	55	ns	
$\overline{\text{LIR}}$ delay time 2	t_{LD2}	–	–	80	–	–	70	–	–	55	ns	
Data read set-up time	t_{DRS}	40	–	–	30	–	–	30	–	–	ns	
Data read hold time*	t_{DRH}	0	–	–	0	–	–	0	–	–	ns	
ST delay time 1	t_{STD1}	–	–	90	–	–	70	–	–	60	ns	
ST delay time 2	t_{STD2}	–	–	90	–	–	70	–	–	60	ns	
$\overline{\text{WAIT}}$ set-up time	t_{WS}	40	–	–	40	–	–	30	–	–	ns	

* Defined against the first signal to go high level of $\overline{\text{ME}}$, $\overline{\text{RD}}$ and $\overline{\text{CS2}}$ – $\overline{\text{CS0}}$

Bus Timing (cont.)

		HD64180SCP6			HD64180SCP8			HD64180SCP10				
Item	Symbol	min	typ	max	min	typ	max	min	typ	max	Unit	Timing
$\overline{\text{WAIT}}$ hold time	t_{WH}	40	–	–	40	–	–	30	–	–	ns	See figures 10, 11, 12, and 13.
Write data floating delay time	t_{WDZ}	–	–	95	–	–	70	–	–	60	ns	
$\overline{\text{WR}}$ delay time 1	t_{WRD1}	–	–	65	–	–	60	–	–	50	ns	
Write data delay time	t_{WDD}	–	–	90	–	–	80	–	–	60	ns	
Write data set-up time (vis-a-vis falling edge of $\overline{\text{WR}}$)	t_{WDS}	40	–	–	20	–	–	15	–	–	ns	
$\overline{\text{WR}}$ delay time 2	t_{WRD2}	–	–	80	–	–	60	–	–	55	ns	
$\overline{\text{WR}}$ pulse width	t_{WRP}	170	–	–	130	–	–	110	–	–	ns	
Write data hold time (vis-a-vis rising edge of $\overline{\text{WR}}$)	t_{WDH}	40	–	–	15	–	–	10	–	–	ns	
$\overline{\text{IOE}}$ delay time 1	t_{IOD1}	–	–	60	–	–	50	–	–	50	ns	
$\overline{\text{IOE}}$ delay time 2	t_{IOD2}	–	–	60	–	–	50	–	–	50	ns	
$\overline{\text{IOE}}$ delay time 3 (from falling edge of $\overline{\text{LIR}}$)	t_{IOD3}	340	–	–	250	–	–	200	–	–	ns	
$\overline{\text{IOE}}$ delay time 4	t_{IOD4}	–	–	65	–	–	60	–	–	55	ns	
$\overline{\text{INT}}$ set-up time (vis-a-vis falling edge of ø)	t_{INTS}	40	–	–	40	–	–	30	–	–	ns	
$\overline{\text{INT}}$ hold time (vis-a-vis falling edge of ø)	t_{INTH}	40	–	–	40	–	–	30	–	–	ns	

Bus Timing (cont.)

		HD64180SCP6			HD64180SCP8			HD64180SCP10				
Item	**Symbol**	**min**	**typ**	**max**	**min**	**typ**	**max**	**min**	**typ**	**max**	**Unit**	**Timing**
$\overline{\text{NMI}}$ pulse width	t_{NMIW}	120	–	–	100	–	–	80	–	–	ns	See figures 10, 11, 12, and 13.
$\overline{\text{BUSREQ}}$ set-up time (vis-a-vis falling edge of ø)	t_{BRS}	40	–	–	40	–	–	30	–	–	ns	
$\overline{\text{BUSREQ}}$ hold time (vis-a-vis falling edge of ø)	t_{BRH}	40	–	–	40	–	–	30	–	–	ns	
$\overline{\text{BUSACK}}$ delay time 1	t_{BAD1}	–	–	95	–	–	70	–	–	60	ns	
$\overline{\text{BUSACK}}$ delay time 2	t_{BAD2}	–	–	95	–	–	70	–	–	60	ns	
Bus floating delay time	t_{BZD}	–	–	125	–	–	90	–	–	80	ns	
$\overline{\text{ME}}$ high-level pulse width	t_{MEWH}	110	–	–	90	–	–	70	–	–	ns	
$\overline{\text{ME}}$ low-level pulse width	t_{MEWL}	125	–	–	100	–	–	80	–	–	ns	
$\overline{\text{REF}}$ delay time 1	t_{RFD1}	–	–	90	–	–	80	–	–	60	ns	
$\overline{\text{REF}}$ delay time 2	t_{RFD2}	–	–	90	–	–	80	–	–	60	ns	
$\overline{\text{HALT}}$ delay time 1	t_{HAD1}	–	–	90	–	–	80	–	–	50	ns	
$\overline{\text{HALT}}$ delay time 2	t_{HAD2}	–	–	90	–	–	80	–	–	50	ns	
$\overline{\text{RESET}}$ set-up time	t_{RES}	120	–	–	100	–	–	80	–	–	ns	
$\overline{\text{RESET}}$ hold time	t_{REH}	80	–	–	80	–	–	80	–	–	ns	
Oscillator stabilize time	t_{OSC}	–	–	20	–	–	20	–	–	40	ms	

Bus Timing (cont.)

Item	Symbol	HD64180SCP6 min	HD64180SCP6 typ	HD64180SCP6 max	HD64180SCP8 min	HD64180SCP8 typ	HD64180SCP8 max	HD64180SCP10 min	HD64180SCP10 typ	HD64180SCP10 max	Unit	Timing
$\overline{\text{RESET}}$ rise time	t_{Rr}	–	–	50	–	–	50	–	–	50	ms	See figures 10, 11, 12, and 13.
$\overline{\text{RESET}}$ fall time	t_{Rf}	–	–	50	–	–	50	–	–	50	ms	
$\overline{\text{CS}}$ delay time 1	t_{CSD1}	–	–	60	–	–	55	–	–	50	ns	
$\overline{\text{CS}}$ delay time 2	t_{CSD2}	–	–	60	–	–	55	–	–	50	ns	

MSCI Timing

Item	Symbol	HD64180SCP6 min	HD64180SCP6 typ	HD64180SCP6 max	HD64180SCP8 min	HD64180SCP8 typ	HD64180SCP8 max	HD64180SCP10 min	HD64180SCP10 typ	HD64180SCP10 max	Unit	Timing
TXCM cycle time (TXCM input)	t_{TCYCM}	1.4*	–	–	1.4*	–	–	1.4*	–	–	t_{cyc}	See figures 14, 15, 16, 17, 18, 19, 20, 21 and 22.
TXCM rise time (TXCM input)	t_{TCrM}	–	–	20	–	–	15	–	–	10	ns	
TXCM fall time (TXCM input)	t_{TCfM}	–	–	20	–	–	15	–	–	10	ns	
TXCM high-level pulse width (TXCM input)	t_{TCHWM}	0.55	–	–	0.55	–	–	0.55	–	–	t_{cyc}	
TXCM low-level pulse width (TXCM input)	t_{TCLWM}	0.55	–	–	0.55	–	–	0.55	–	–	t_{cyc}	
TXDM delay time (TXCM input)	t_{TDD1M}	–	–	130	–	–	100	–	–	80	ns	
TXDM delay time (TXCM output)	t_{TDD2M}	–	–	80	–	–	65	–	–	50	ns	
RXCM cycle time	t_{RCYCM}	1.4*	–	–	1.4*	–	–	1.4*	–	–	t_{cyc}	

* In asynchronous mode, loop mode, t_{TCYCM}, t_{RCYCM} = 2.5 t_{cyc} (min).

MSCI Timing (cont.) (Vcc=5V ± 10%, Vss=0V, Ta=-20 to +75°C unless otherwise specified)

		HD64180SCP6			HD64180SCP8			HD64180SCP10				
Item	Symbol	min	typ	max	min	typ	max	min	typ	max	Unit	Timing
RXCM rise time	t_{RCrM}	–	–	20	–	–	15	–	–	10	ns	See figures 14, 15, 16, 17, 18, 19, 20, 21 and 22.
RXCM fall time	t_{RCfM}	–	–	20	–	–	15	–	–	10	ns	
RXCM high-level pulse width	t_{RCHWM}	0.55	–	–	0.55	–	–	0.55	–	–	t_{CYC}	
RXCM low-level pulse width	t_{RCLWM}	0.55	–	–	0.55	–	–	0.55	–	–	t_{CYC}	
RXDM-RXCM set-up time (RXCM input)	t_{RDS1M}	50	–	–	40	–	–	30	–	–	ns	
RXCM-RXDM hold time (RXCM input)	t_{RDH1M}	40	–	–	30	–	–	20	–	–	ns	
RXDM-RXCM set-up time (RXCM output)	t_{RDS2M}	130	–	–	100	–	–	80	–	–	ns	
RXCM-RXDM hold time (RXCM output)	t_{RDH2M}	40	–	–	30	–	–	20	–	–	ns	
ADPLL operating clock cycle time	t_{PLCYM}	120	–	–	80	–	–	57	–	–	ns	
ADPLL operating clock rise time	t_{PLrM}	–	–	15	–	–	10	–	–	8	ns	
ADPLL operating clock fall time	t_{PLfM}	–	–	15	–	–	10	–	–	8	ns	
ADPLL operating clock high-level pulse width	t_{PLHWM}	25	–	–	15	–	–	10	–	–	ns	
ADPLL operating clock low-level pulse width	t_{PLLWM}	25	–	–	15	–	–	10	–	–	ns	
ø-BRG* output delay time	t_{BGDM}	–	–	150	–	–	120	–	–	95	ns	

* $f_{BRG} \neq fø$ (f_{BRG} is the baud rate generator output frequency; fø is the CPU operating clock frequency).

MSCI Timing (cont.) (Vcc=5V ± 10%, Vss=0V, Ta=-20 to +75°C unless otherwise specified)

		HD64180SCP6			HD64180SCP8			HD64180SCP10				
Item	**Symbol**	**min**	**typ**	**max**	**min**	**typ**	**max**	**min**	**typ**	**max**	**Unit**	**Timing**
TXCM/RXCM output rise time	t_{BGMr}	–	–	50	–	–	40	–	–	30	ns	See figures 14, 15, 16, 17, 18, 19, 20, 21 and 22.
TXCM/RXCM output fall time	t_{BGMf}	–	–	50	–	–	40	–	–	30	ns	
RXCM-$\overline{\text{SYNC}}$ set-up time	t_{SYSU}	2.5	–	–	2.5	–	–	2.5	–	–	t_{cyc}	
RXCM-$\overline{\text{SYNC}}$ hold time	t_{SYHD}	2.5		–	2.5	–	–	2.5	–	–	t_{cyc}	
$\overline{\text{CTSM}}$ high-level pulse width	t_{CTSHWM}	2.0	–		2.0	–	–	2.0	–	–	t_{cyc}	
$\overline{\text{CTSM}}$ low-level pulse width	t_{CTSLWM}	2.0	–	–	2.0	–	–	2.0	–	–	t_{cyc}	
$\overline{\text{DCDM}}$ high-level pulse width	t_{DCDHWM}	2.0	–	–	2.0	–	–	2.0	–	–	t_{cyc}	
$\overline{\text{DCDM}}$ low-level pulse width	t_{DCDLWM}	2.0	–	–	2.0	–	–	2.0	–	–	t_{cyc}	
ø-$\overline{\text{RTSM}}$ delay time	t_{RTSDM}	–	–	100	–	–	85	–	–	70	ns	

ASCI/CSIO Timing (Vcc=5V ± 10%, Vss=0V, Ta=-20 to +75°C unless otherwise specified)

		HD64180SCP6			HD64180SCP8			HD64180SCP10				
Item	**Symbol**	**min**	**typ**	**max**	**min**	**typ**	**max**	**min**	**typ**	**max**	**Unit**	**Timing**
TXCA input cycle time	t_{TCYC}	2.5	–	–	2.5	–	–	2.5	–	–	t_{CYC}	See figures 23, 24, 25, 26, 27, 28 and 29.
TXCA input high-level pulse width	t_{TCHW}	0.55	–	–	0.55	–	–	0.55	–	–	t_{CYC}	
TXCA input low-level pulse width	t_{TCLW}	0.55	–	–	0.55	–	–	0.55	–	–	t_{CYC}	
TXCA input rise time	t_{TCr}	–	–	30	–	–	20	–	–	10	ns	
TXCA input fall time	t_{TCf}	–	–	30	–	–	20	–	–	10	ns	
TXDA delay time 1	t_{TDD1}	1.5	–	3.0	1.5	–	3.0	1.5	–	3.0	t_{CYC}	
TXDA delay time 2	t_{TDD2}	–	–	50	–	–	40	–	–	30	ns	
RXCA input cycle time	t_{RCYC}	2.5	–	–	2.5	–	–	2.5	–	–	t_{CYC}	
RXCA input high-level pulse width	t_{RCHW}	0.55	–	–	0.55	–	–	0.55	–	–	t_{CYC}	
RXCA input low-level pulse width	t_{RCLW}	0.55	–	–	0.55	–	–	0.55	–	–	t_{CYC}	
RXCA input rise time	t_{RCr}	–	–	30	–	–	20	–	–	10	ns	
RXCA input fall time	t_{RCf}	–	–	30	–	–	20	–	–	10	ns	

ASCI/CSIO Timing (cont.) (Vcc=5V ± 10%, Vss=0V, Ta=-20 to +75°C unless otherwise specified)

		HD64180SCP6			HD64180SCP8			HD64180SCP10				
Item	**Symbol**	**min**	**typ**	**max**	**min**	**typ**	**max**	**min**	**typ**	**max**	**Unit**	**Timing**
RXDA set-up time 1	t_{RDS1}	50	–	–	40	–	–	30	–	–	ns	See figures 23, 24, 25, 26, 27, 28 and 29.
RXDA hold time 1	t_{RDH1}	40	–	–	30	–	–	20	–	–	ns	
RXDA set-up time 2	t_{RDS2}	130	–	–	100	–	–	80	–	–	ns	
RXDA hold time 2	t_{RDH2}	40	–	–	30	–	–	20	–	–	ns	
ø-BRG output delay time	t_{BGDA}	–	–	80	–	–	70	–	–	60	ns	
TXCA/RXCA output rise time	t_{BGAr}	–	–	50	–	–	40	–	–	30	ns	
TXCA/RXCA output fall time	t_{BGAf}	–	–	50	–	–	40	–	–	30	ns	
$\overline{\text{CTSA}}$ high-level pulse width	t_{CTSHW}	2.0	–	–	2.0	–	–	2.0	–	–	t_{CYC}	
$\overline{\text{CTSA}}$ low-level pulse width	t_{CTSLW}	2.0	–	–	2.0	–	–	2.0	–	–	t_{CYC}	
$\overline{\text{DCDA}}$ high-level pulse width	t_{DCDHW}	2.0	–	–	2.0	–	–	2.0	–	–	t_{CYC}	
$\overline{\text{DCDA}}$ low-level pulse width	t_{DCDLW}	2.0	–	–	2.0	–	–	2.0	–	–	t_{CYC}	
$\overline{\text{RTSA}}$ delay time	t_{RTSD}	–	–	100	–	–	85	–	–	70	ns	

DMAC Timing (Vcc=5V ± 10%, Vss=0V, Ta=-20 to +75°C unless otherwise specified)

		HD64180SCP6			HD64180SCP8			HD64180SCP10				
Item	**Symbol**	**min**	**typ**	**max**	**min**	**typ**	**max**	**min**	**typ**	**max**	**Unit**	**Timing**
$\overline{\text{DREQ}}$ set-up time	t_{DREQS}	40	–	–	40	–	–	30	–	–	ns	See figure 30.
$\overline{\text{DREQ}}$ hold time	t_{DREQH}	40	–	–	40	–	–	30	–	–	ns	
$\overline{\text{TEND}}$ delay time 1	t_{TED1}	–	–	70	–	–	60	–	–	50	ns	
$\overline{\text{TEND}}$ delay time 2	t_{TED2}	–	–	70	–	–	60	–	–	50	ns	
ST delay time 1	t_{STD1}	–	–	90	–	–	70	–	–	60	ns	
ST delay time 2	t_{STD2}	–	–	90	–	–	70	–	–	60	ns	

Timer Timing (Vcc=5V ± 10%, Vss=0V, Ta=-20 to +75°C unless otherwise specified)

		HD64180SCP6			HD64180SCP8			HD64180SCP10				
Item	**Symbol**	**min**	**typ**	**max**	**min**	**typ**	**max**	**min**	**typ**	**max**	**Unit**	**Timing**
Timer input pulse width	t_{PWT}	2.0	–	–	2.0	–	–	2.0	–	–	t_{cyc}	See figure 31.
Timer input set-up time	t_{PDSU}	40	–	–	40	–	–	30	–	–	ns	
Timer input hold time	t_{PDH}	40	–	–	40	–	–	30	–	–	ns	
Timer output delay time	t_{TOD}	–	–	100	–	–	85	–	–	70	ns	

EXTAL Input Clock Signal Timing (Vcc=5V ± 10%, Vss=0V, Ta=–20 to +75°C unless otherwise specified)

		HD64180SCP6			HD64180SCP8			HD64180SCP10				
Item	**Symbol**	**min**	**typ**	**max**	**min**	**typ**	**max**	**min**	**typ**	**max**	**Unit**	**Timing**
External clock cycle time	t_{ECYC}	81	–	1000	62	–	1000	50	–	1000	ns	See figure 32.
External clock high-level pulse width	t_{ECHW}	20	–	–	15	–	–	10	–	–	ns	
External clock low-level pulse width	t_{ECLW}	20	–	–	15	–	–	10	–	–	ns	
External clock fall time	t_{ECf}	–	–	25	–	–	25	–	–	15	ns	
External clock rise time	t_{ECr}	–	–	25	–	–	25	–	–	15	ns	

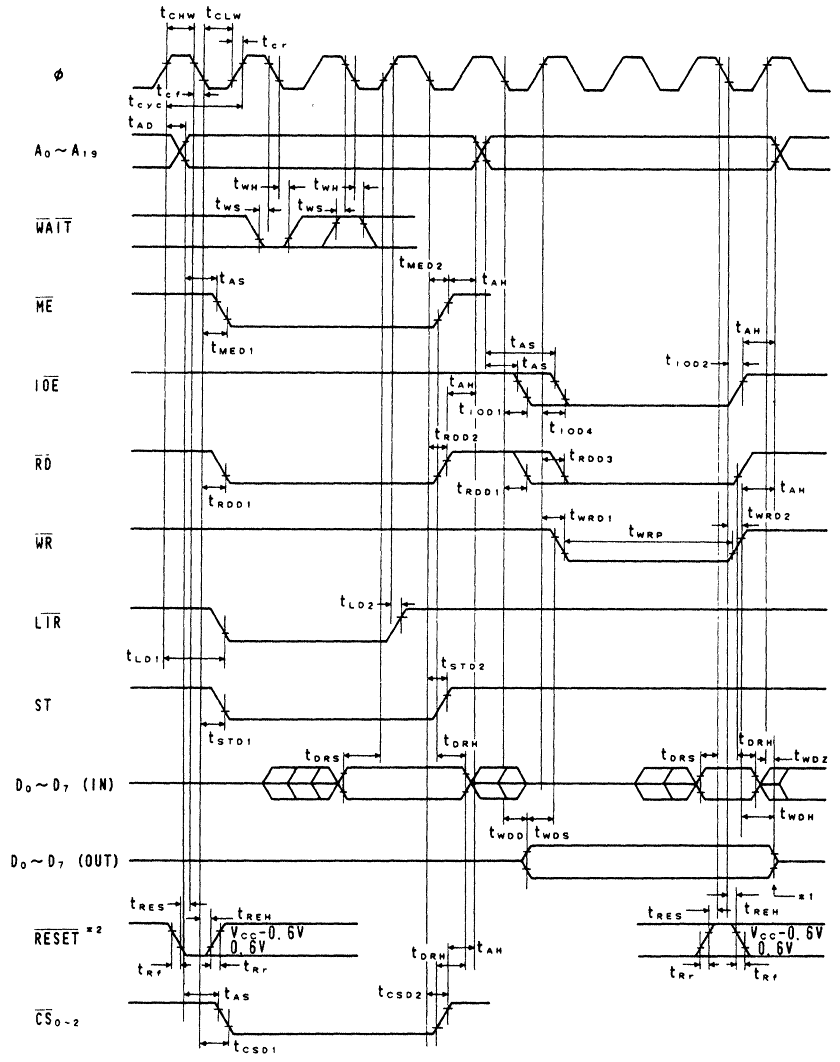
Miscellaneous

Rise and Fall Times of Input Signals with No Characteristics Specified

(Vcc = 5V ± 10%, Vss = 0V, Ta = –20 to +75°C unless otherwise specified)

		HD64180SCP6			HD64180SCP8			HD64180SCP10				
Item	**Symbol**	**min**	**typ**	**max**	**min**	**typ**	**max**	**min**	**typ**	**max**	**Unit**	**Timing**
Input line rise time (no characteristics specified)	t_{ir}	–	–	100	–	–	100	–	–	100	ns	See figure 33.
Input line fall time (no characteristics specified)	t_{if}	–	–	100	–	–	100	–	–	100	ns	

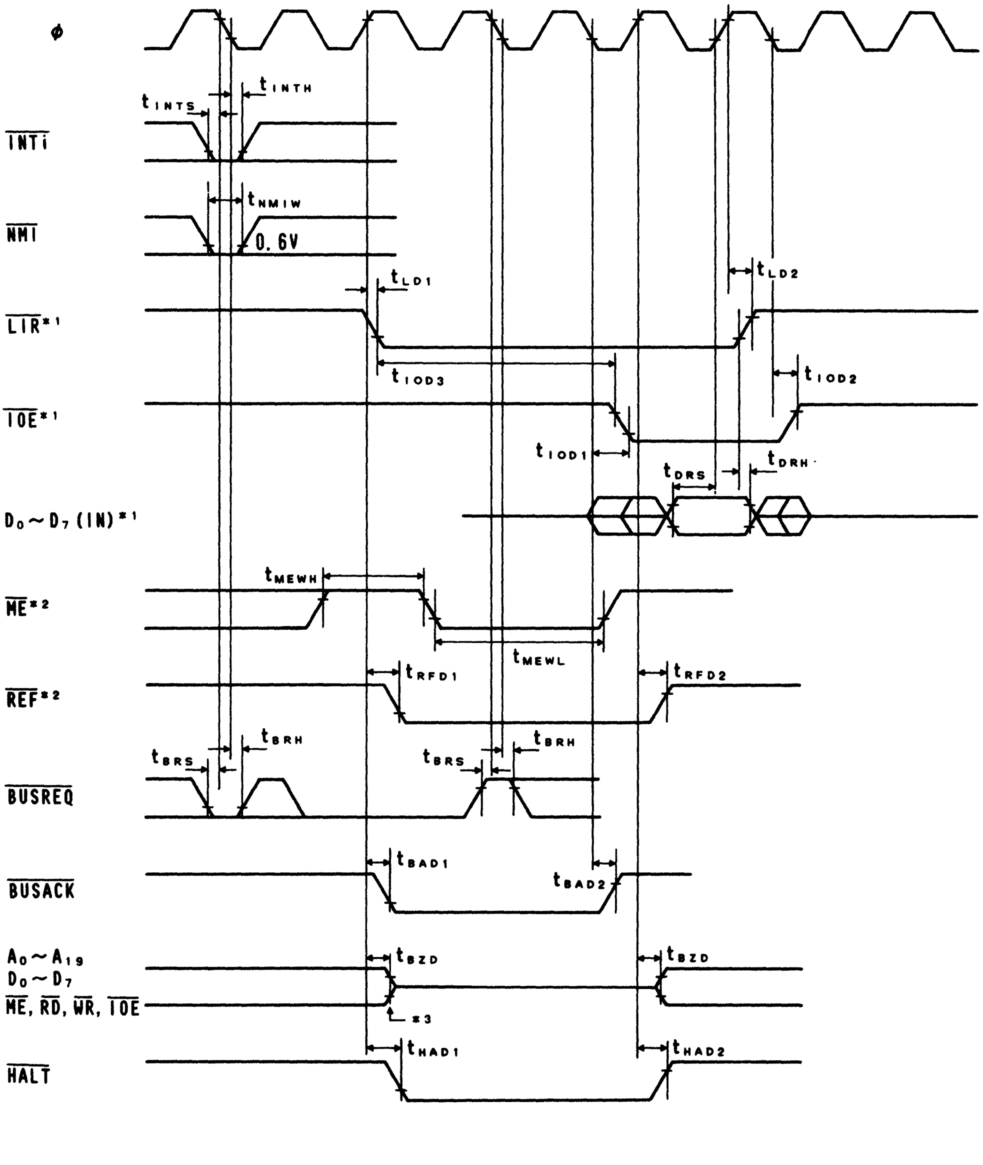
Bus Timing



*1 Output buffer inactivation timing.

*2 A low-level signal should be input to the RESET pin such that it is sampled at low level for at least six successive ø clock falling edges. After the reset mode is entered, it may require up to 10 clock cycles before all of the output lines are set to their initialize conditions. See "Reset Mode," for details.

Figure 10. Bus Timing (1)



*1 During INTo acknowledge cycle

*2 During refresh cycle

*3 Output buffer inactivation timing

Figure 11. Bus Timing (2)

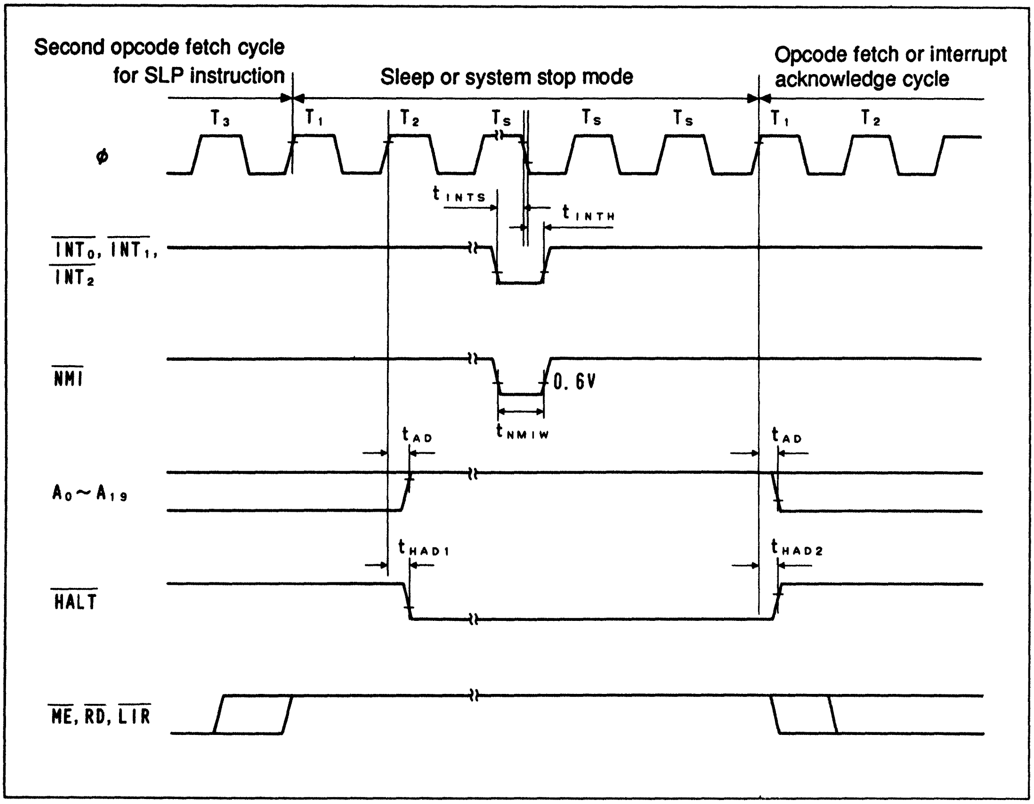


Figure 12. Bus Timing (3) (sleep or system stop mode)

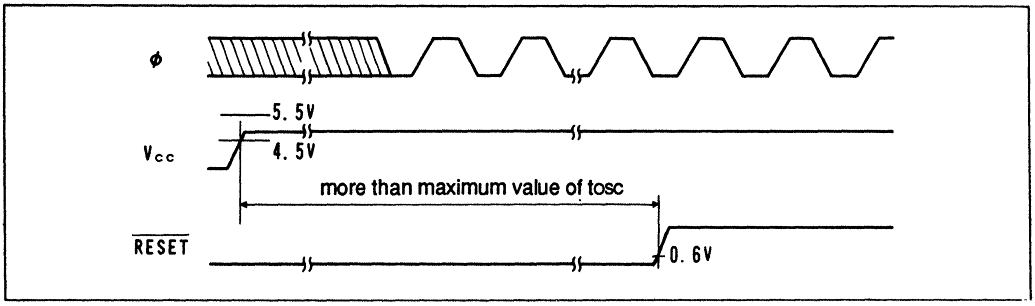
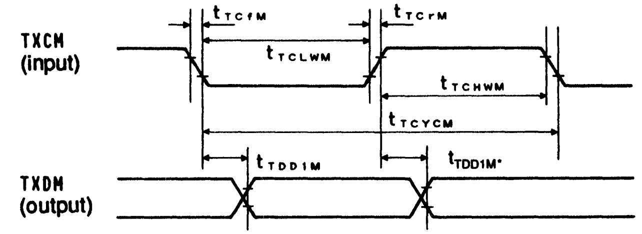


Figure 13. Bus Timing (4)

MSCI Timing

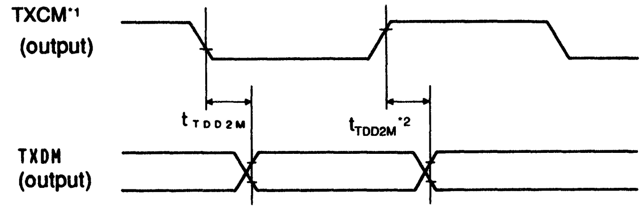
(1) Transmit timing (TXCM input)



* Defines in the FM type codes.

Figure 14. Transmit Timing (TXCM input)

(2) Transmit timing (TXCM output)



*1 For details of the TXCM waveform, see figure 13-10.

*2 Defines in the FM type codes.

Figure 15. Transmit Timing (TXCM output)

(3) Receive Timing (RXCM input)

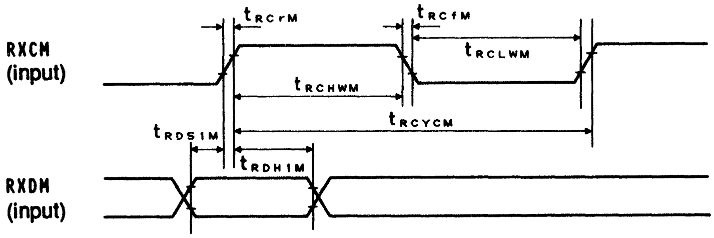
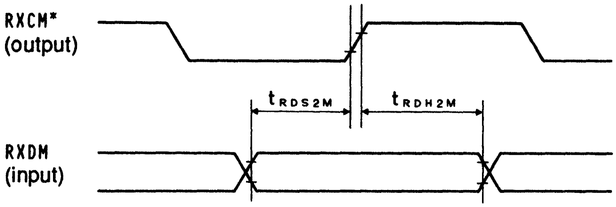


Figure 16. Receive Timing (RXCM input)

(4) Receive Timing (RXCM output)



*For details of the RXCA waveform, see figure 19.

Figure 17. Receive Timing (RXCM output)

(5) ADPLL Operating Clock Timing

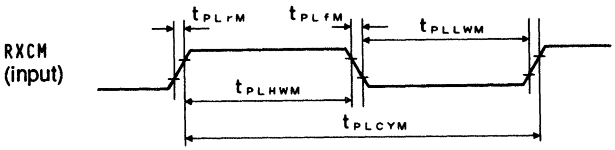


Figure 18. ADPLL Operating Clock Timing

(6) Baud Rate Generator Output Timing

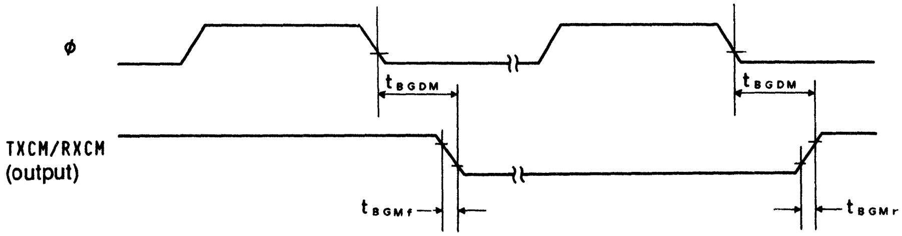


Figure 19. Baud Rate Generator Output Timing ($f_{BRG}=f_{\phi}$)

(7) $\overline{\text{SYNC}}$ Timing

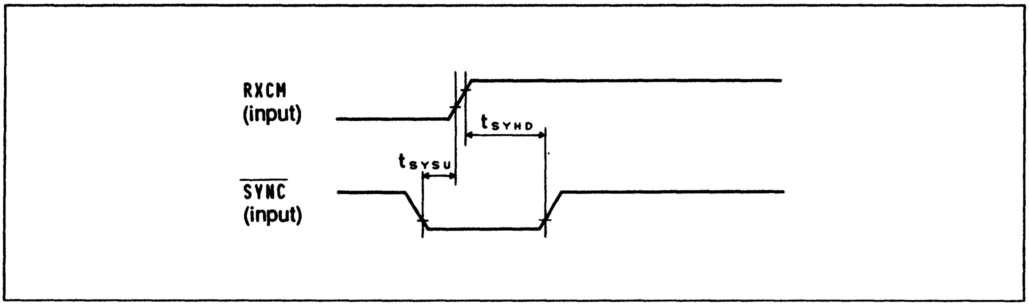


Figure 20. $\overline{\text{SYNC}}$ Timing

(8) $\overline{\text{CTSM}}$ and $\overline{\text{DCDM}}$ Timing

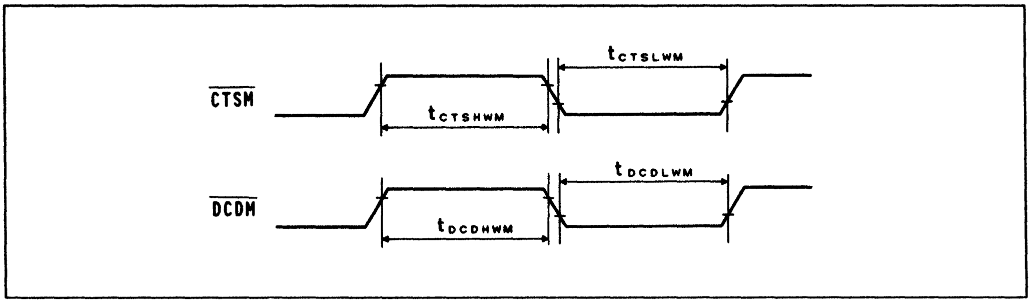


Figure 21. $\overline{\text{CTSM}}$ and $\overline{\text{DCDM}}$ Timing

(9) $\overline{\text{RTSM}}$ Timing

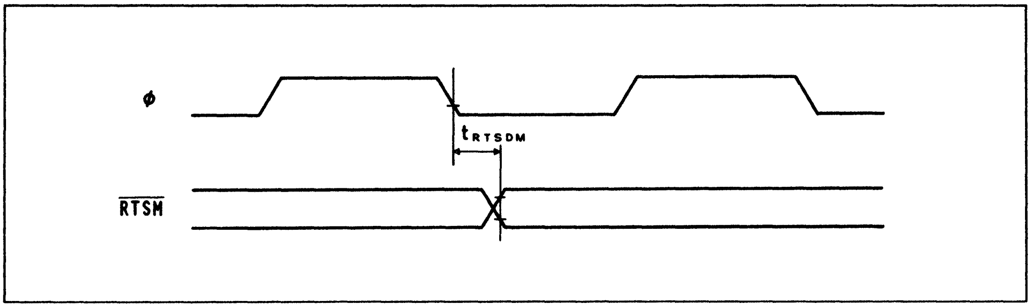


Figure 22. $\overline{\text{RTSM}}$ Timing

ASCI/CSIO Timing

(1) Transmit Timing (TXCA input)

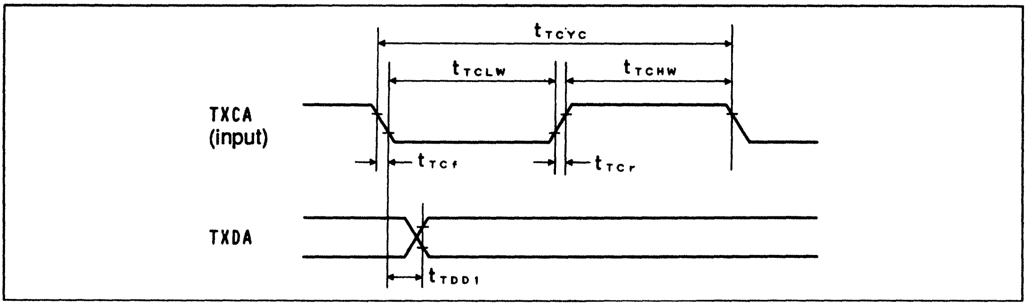
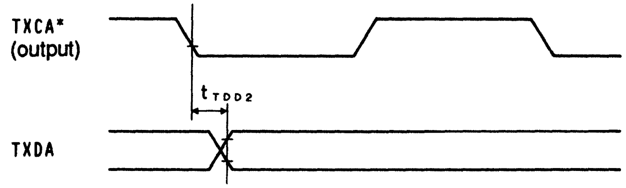


Figure 23. Transmit Timing (TXCA input)

(2) Transmit Timing (TXCA output)



* For details of the TXCA waveform, see figure 3-18.

Figure 24. Transmit Timing (TXCA output)

(3) Receive Timing (RXCA input)

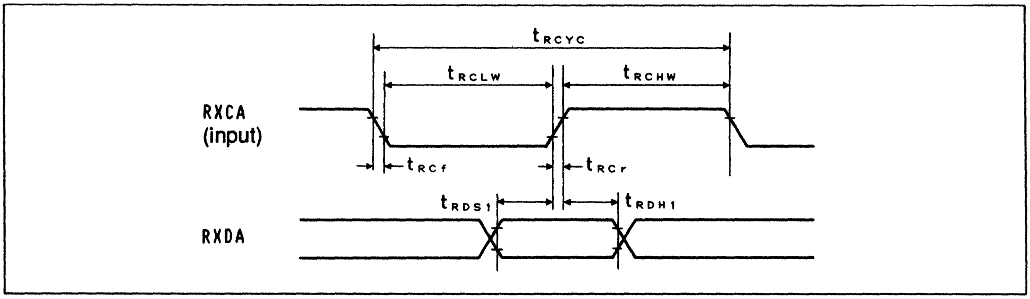
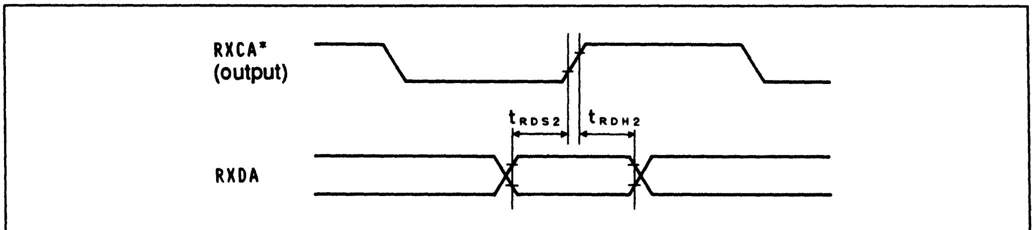


Figure 25. Receive Timing (RXCA input)

(4) Receive Timing (RXCA output)



*For details of the RXCA waveform, see figure 27.

Figure 26. Receive Timing (RXCA output)

(5) Baud Rate Generator Timing

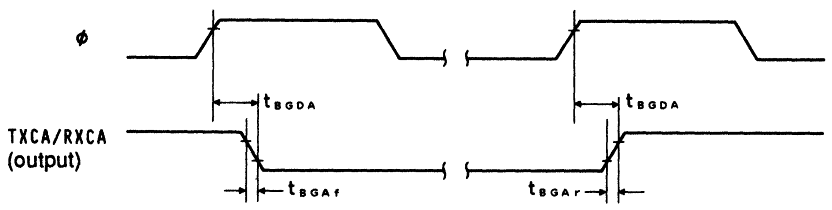


Figure 27. Baud Rate Generator Timing

(6) $\overline{\text{CTSA}}$ and $\overline{\text{DCDA}}$ Timing

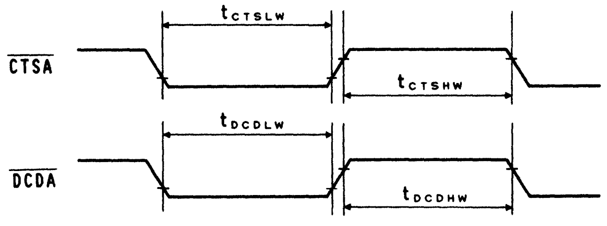


Figure 28. $\overline{\text{CTSA}}$ and $\overline{\text{DCDA}}$ Timing

(7) $\overline{\text{RTSA}}$ Timing

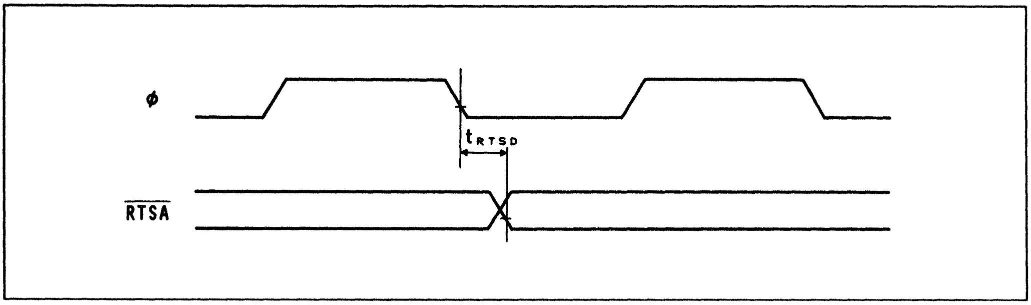
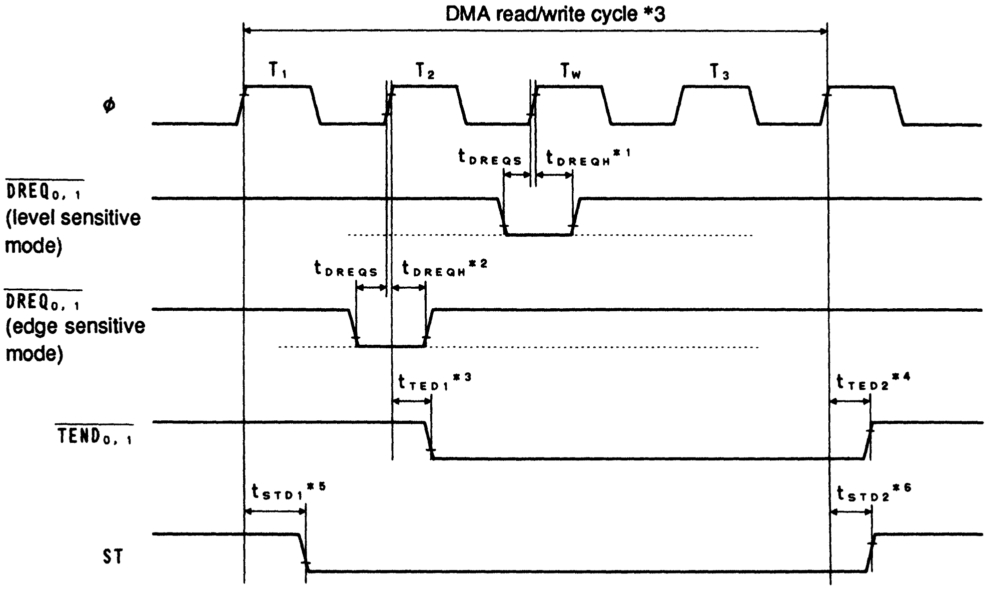


Figure 29. $\overline{\text{RTSA}}$ Timing

DMAC Timing



*1 Defines the rising edge of the clock immediately preceding T3 in single-block transfer mode (dual address type) in write cycle

*2 Defines the rising edge of the clock

*3 Defines a read cycle in single-block transfer mode (dual address type)

*4 Defines the clock after T3 of write cycle in single-block transfer mode (dual address type)

*5 Defines the rising edge of the clock at the beginning of a DMA cycle

*6 Defines the rising edge of the clock at the beginning of a CPU cycle

Note: The timing for $\overline{ME}$, $\overline{IOE}$, $\overline{RD}$, $\overline{WR}$, D_0-D_7, and A_0-A_{19} during read/write operations is the same during CPU operation.

Figure 30. DMAC Timing

Timer Timing

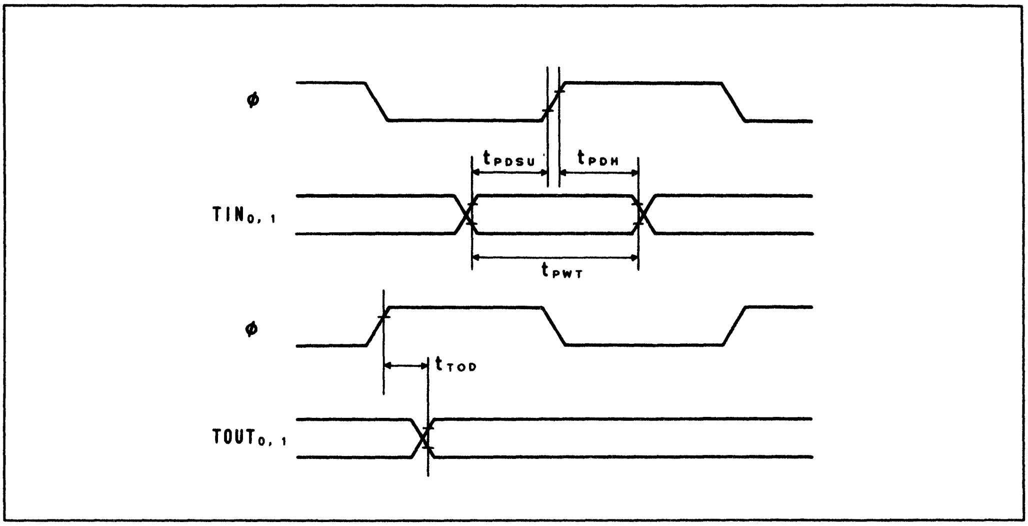


Figure 31. Timer Timing

EXTAL Input Clock Signal Timing

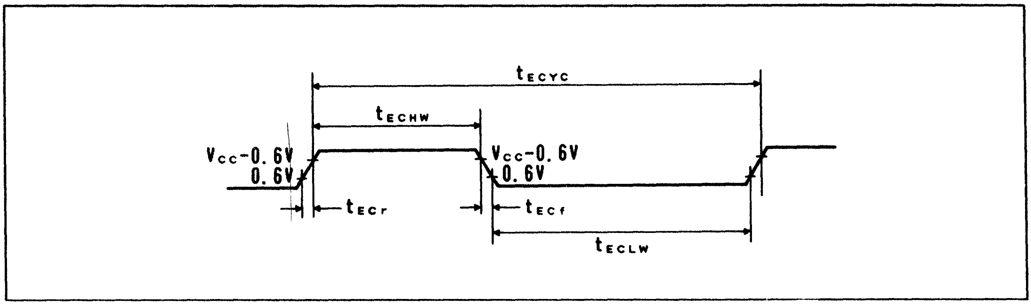


Figure 32. EXTAL Input Clock Signal Timing

Miscellaneous

(1) Rise and Fall Times of Input Signals with No Characteristics Specified

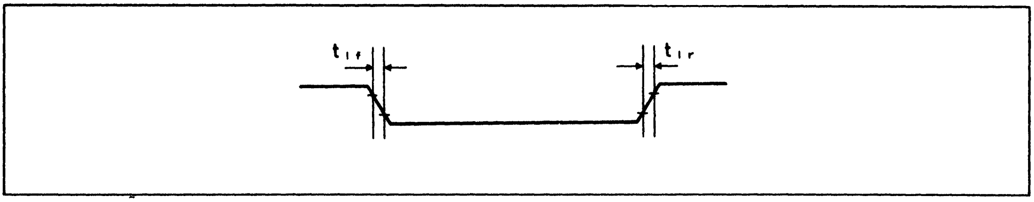


Figure 33. Rise and Fall Times of Input Signals with No Characteristics Specified

(2) Reference Levels (when not otherwise specified)

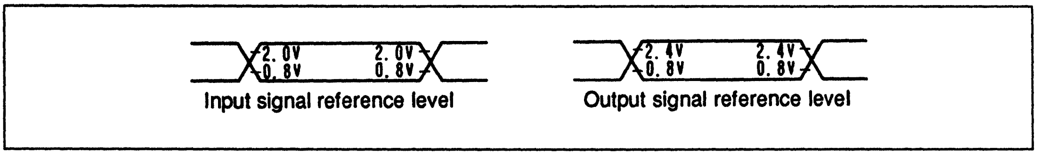


Figure 34. Reference Levels

(3) Bus Timing Load (TTL load)

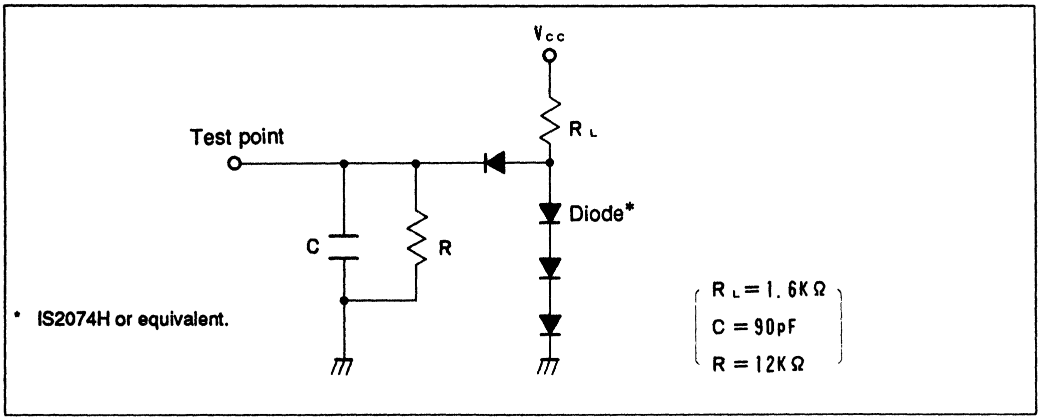


Figure 35. Bus Timing Load

■ Instruction Set

In the instruction set, the following conventions are used:

(1) Register specification

g, g' represents a 8-bit register, while ww, xx, yy, or zz represents a pair of 8-bit registers. The corresponding registers are listed below.

g,g'	Register	ww	Register	xx	Register	yy	Register	zz	Register
000	B	00	BC	00	BC	00	BC	00	BC
001	C	01	DE	01	DE	01	DE	01	DE
010	D	10	HL	10	IX	10	IY	10	HL
011	E	11	SP	11	SP	11	SP	11	AF
100	H								
101	L								
111	A								

Note: ww, xx, yy, or xx plus H or L (eg, wwH, IXL) indicates the high or low order byte of a 16-bit register.

(2) Bit specification.

'b' indicates the bit number of the bit operand in a bit manipulation instruction. The corresponding bits are listed below.

B	Bit
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

(3) Condition specification

'f' indicates the condition for executing an instruction, based on the arithmetic result. The corresponding conditions are listed below.

f	Condition	
000	NZ	non zero
001	Z	zero
010	NC	non carry
011	C	carry
100	PO	parity odd
101	PE	parity even
110	P	sign plus
111	M	sign minus

(4) Restart address

'v' indicates the restart address of a restart instruction. The corresponding addresses are listed below.

v	Address
000	00H
001	08H
010	10H
011	18H
100	20H
101	28H
110	30H
111	38H

(5) Flag

Flag changes are indicated by the following symbols:

•: The flag is not changed by the instruction.
X: Flag change by this instruction is undefined.

↕: The flag is changed according to the arithmetic result of the instruction.

S: The flag is set to 1 by the instruction.

R: The flag is reset to 0 by the instruction.

P: The flag is changed as a parity flag by the instruction.

V: The flag is changed as an overflow flag by the instruction.

(6) Others

( )M:	Indicates the memory at the address indicated in parentheses.
( )I:	Indicates the I/O at the address indicated in parentheses.
m or n:	8-bit value
mn:	16-bit value
r:	Subscript r indicates a 8-bit register.
R:	Subscript R indicates a 16-bit register.
b·( )M:	Indicates the memory bit specified by b at the address indicated in parentheses.
b·gr:	Indicates the register bit specified by b in the register specified by gr.
d or j:	Signed 8-bit displacement
S:	Source addressing mode
D:	Destination addressing mode
·:	AND
+:	OR
⊕:	Exclusive OR

• Data Manipulation Instructions

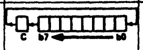
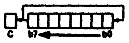
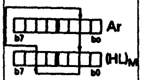
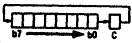
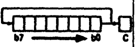
(1) Arithmetic and logic instructions (8bits)

Operation name	MNEMONICS	OP code	Addressing							Bytes	States	Operation	Flag					
													7	6	4	2	1	0
			IMMED	EXT	IND	REG	REGI	IMP	REL				S	Z	H	P/V	N	C
ADD	ADD A,g	10 000 g				S		D		1	4	Ar+gr→Ar	↕	↕	↕	V	R	↕
	ADD A,(HL)	10 000 110					S	D		1	6	Ar+(HL)$_M$→Ar	↕	↕	↕	V	R	↕
	ADD A,m	11 000 110 〈 m 〉	S					D		2	6	Ar+m→Ar	↕	↕	↕	V	R	↕
	ADD A,(IX+d)	11 011 101 10 000 110 〈 d 〉			S			D		3	14	Ar+(IX+d)$_M$→Ar	↕	↕	↕	V	R	↕
	ADD A,(IY+d)	11 111 101 10 000 110 〈 d 〉			S			D		3	14	Ar+(IY+d)$_M$→Ar	↕	↕	↕	V	R	↕
ADC	ADC A,g	10 001 g				S		D		1	4	Ar+gr+c→Ar	↕	↕	↕	V	R	↕
	ADC A,(HL)	10 001 110					S	D		1	6	Ar+(HL)$_M$+c→Ar	↕	↕	↕	V	R	↕
	ADC A,m	11 001 110 〈 m 〉	S					D		2	6	Ar+m+c→Ar	↕	↕	↕	V	R	↕
	ADC A,(IX+d)	11 011 101 10 001 110 〈 d 〉			S			D		3	14	Ar+(IX+d)$_M$+c→Ar	↕	↕	↕	V	R	↕
	ADC A,(IY+d)	11 111 101 10 001 110 〈 d 〉			S			D		3	14	Ar+(IY+d)$_M$+c→Ar	↕	↕	↕	V	R	↕
AND	AND g	10 100 g				S		D		1	4	Ar·gr→Ar	↕	↕	S	P	R	R
	AND (HL)	10 100 110					S	D		1	6	Ar·(HL)$_M$→Ar	↕	↕	S	P	R	R
	AND m	11 100 110 〈 m 〉	S					D		2	6	Ar·m→Ar	↕	↕	S	P	R	R
	AND (IX+d)	11 011 101 10 100 110 〈 d 〉			S			D		3	14	Ar·(IX+d)$_M$→Ar	↕	↕	S	P	R	R
	AND (IY+d)	11 111 101 10 100 110 〈 d 〉			S			D		3	14	Ar·(IY+d)$_M$→Ar	↕	↕	S	P	R	R
Compare	CP g	10 111 g				S		D		1	4	Ar−gr	↕	↕	↕	V	S	↕
	CP (HL)	10 111 110					S	D		1	6	Ar−(HL)$_M$	↕	↕	↕	V	S	↕
	CP m	11 111 110 〈 m 〉	S					D		2	6	Ar−m	↕	↕	↕	V	S	↕
	CP (IX+d)	11 011 101 10 111 110 〈 d 〉			S			D		3	14	Ar−(IX+d)$_M$	↕	↕	↕	V	S	↕
	CP (IY+d)	11 111 101 10 111 110 〈 d 〉			S			D		3	14	Ar−(IY+d)$_M$	↕	↕	↕	V	S	↕
COMPLE-MENT	CPL	00 101 111						S/D		1	3	$\overline{\text{Ar}}$→Ar	·	·	S	·	S	·
DEC	DEC g	00 g 101				S/D				1	4	gr−1→gr	↕	↕	↕	V	S	·
	DEC (HL)	00 110 101					S/D			1	10	(HL)$_M$−1→(HL)$_M$	↕	↕	↕	V	S	·
	DEC (IX+d)	11 011 101 00 110 101 〈 d 〉			S/D					3	18	(IX+d)$_M$−1→ (IX+d)$_M$	↕	↕	↕	V	S	·
	DEC (IY+d)	11 111 101 00 110 101 〈 d 〉			S/D					3	18	(IY+d)$_M$−1→ (IY+d)$_M$	↕	↕	↕	V	S	·
INC	INC g	00 g 100				S/D				1	4	gr+1→gr	↕	↕	↕	V	R	·
	INC (HL)	00 110 100					S/D			1	10	(HL)$_M$+1→(HL)$_M$	↕	↕	↕	V	R	·
	INC (IX+d)	11 011 101 00 110 100 〈 d 〉			S/D					3	18	(IX+d)$_M$+1→ (IX+d)$_M$	↕	↕	↕	V	R	·
	INC (IY+d)	11 111 101 00 110 100 〈 d 〉			S/D					3	18	(IY+d)$_M$+1→ (IY+d)$_M$	↕	↕	↕	V	R	·

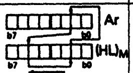
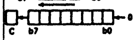
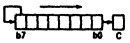
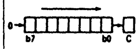
(Continued)

Operation name	MNEMONICS	OP code	Addressing							Bytes	·States	Operation	Flag					
													7	6	4	2	1	0
			IMMED	EXT	IND	REG	REGI	IMP	REL				S	Z	H	P/V	N	C
MULT	MLT ww	11 101 101 01 ww1 100				S/D				2	17	wwHr×wwLr→ww$_R$	·	·	·	·	·	·
NEGATE	NEG	11 101 101 01 000 100						S/D		2	6	0−Ar→Ar	↕	↕	↕	V	S	↕
OR	OR g	10 110 g				S		D		1	4	Ar+gr→Ar	↕	↕	R	P	R	R
	OR (HL)	10 110 110					S	D		1	6	Ar+(HL)$_M$→Ar	↕	↕	R	P	R	R
	OR m	11 110 110 ⟨ m ⟩	S					D		2	6	Ar+m→Ar	↕	↕	R	P	R	R
	OR (IX+d)	11 011 101 10 110 110 ⟨ d ⟩			S			D		3	14	Ar+(IX+d)$_M$→Ar	↕	↕	R	P	R	R
	OR (IY+d)	11 111 101 10 110 110 ⟨ d ⟩			S			D		3	14	Ar+(IY+d)$_M$→Ar	↕	↕	R	P	R	R
SUB	SUB g	10 010 g				S		D		1	4	Ar−gr→Ar	↕	↕	↕	V	S	↕
	SUB (HL)	10 010 110					S	D		1	6	Ar−(HL)$_M$→Ar	↕	↕	↕	V	S	↕
	SUB m	11 010 110 ⟨ m ⟩	S					D		2	6	Ar−m→Ar	↕	↕	↕	V	S	↕
	SUB (IX+d)	11 011 101 10 010 110 ⟨ d ⟩			S			D		3	14	Ar−(IX+d)$_M$→Ar	↕	↕	↕	V	S	↕
	SUB (IY+d)	11 111 101 10 010 110 ⟨ d ⟩			S			D		3	14	Ar−(IY+d)$_M$→Ar	↕	↕	↕	V	S	↕
SUBC	SBC A,g	10 011 g				S		D		1	4	Ar−gr−c→Ar	↕	↕	↕	V	S	↕
	SBC A,(HL)	10 011 110					S	D		1	6	Ar−(HL)$_M$−c→Ar	↕	↕	↕	V	S	↕
	SBC A,m	11 011 110 ⟨ m ⟩	S					D		2	6	Ar−m−c→Ar	↕	↕	↕	V	S	↕
	SBC A,(IX+d)	11 011 101 10 011 110 ⟨ d ⟩			S			D		3	14	Ar−(IX+d)$_M$−c→Ar	↕	↕	↕	V	S	↕
	SBC A,(IY+d)	11 111 101 10 011 110 ⟨ d ⟩			S			D		3	14	Ar−(IY+d)$_M$−c→Ar	↕	↕	↕	V	S	↕
TEST	TST g	11 101 101 00 g 100				S				2	7	Ar·gr	↕	↕	S	P	R	R
	TST (HL)	11 101 101 00 110 100					S			2	10	Ar·(HL)$_M$	↕	↕	S	P	R	R
	TST m	11 101 101 01 100 100 ⟨ m ⟩	S							3	9	Ar·m	↕	↕	S	P	R	R
XOR	XOR g	10 101 g				S		D		1	4	Ar⊕gr→Ar	↕	↕	R	P	R	R
	XOR (HL)	10 101 110					S	D		1	6	Ar⊕(HL)$_M$→Ar	↕	↕	R	P	R	R
	XOR m	11 101 110 ⟨ m ⟩	S					D		2	6	Ar⊕m→Ar	↕	↕	R	P	R	R
	XOR (IX+d)	11 011 101 10 101 110 ⟨ d ⟩			S			D		3	14	Ar⊕(IX+d)$_M$→Ar	↕	↕	R	P	R	R
	XOR (IY+d)	11 111 101 10 101 110 ⟨ d ⟩			S			D		3	14	Ar⊕(IY+d)$_M$→Ar	↕	↕	R	P	R	R

(2) Rotate/shift instructions

Operation name	MNEMONICS	OP code	Addressing							Bytes	States	Operation	Flag					
													7	6	4	2	1	0
			IMMED	EXT	IND	REG	REGI	IMP	REL				S	Z	H	P/V	N	C
Rotate and Shift Data	RLA	00 010 111						S/D		1	3		·	·	R	·	R	↕
	RL g	11 001 011				S/D				2	7		↕	↕	R	P	R	↕
		00 010 g																
	RL (HL)	11 001 011					S/D			2	13		↕	↕	R	P	R	↕
		00 010 110																
	RL (IX+d)	11 011 101			S/D					4	19		↕	↕	R	P	R	↕
		11 001 011																
		〈 d 〉																
		00 010 110																
	RL (IY+d)	11 111 101			S/D					4	19		↕	↕	R	P	R	↕
		11 001 011																
		〈 d 〉																
		00 010 110																
	RLCA	00 000 111						S/D		1	3		·	·	R	·	R	↕
	RLC g	11 001 011				S/D				2	7		↕	↕	R	P	R	↕
		00 000 g																
	RLC (HL)	11 001 011					S/D			2	13		↕	↕	R	P	R	↕
		00 000 110																
	RLC (IX+d)	11 011 101			S/D					4	19		↕	↕	R	P	R	↕
		11 001 011																
		〈 d 〉																
		00 000 110																
	RLC (IY+d)	11 111 101			S/D					4	19		↕	↕	R	P	R	↕
		11 001 011																
		〈 d 〉																
		00 000 110																
	RLD	11 101 101						S/D		2	16		↕	↕	R	P	R	·
		01 101 111																
	RRA	00 011 111						S/D		1	3		·	·	R	·	R	↕
	RR g	11 001 011				S/D				2	7		↕	↕	R	P	R	↕
		00 011 g																
	RR (HL)	11 001 011					S/D			2	13		↕	↕	R	P	R	↕
		00 011 110																
	RR (IX+d)	11 011 101			S/D					4	19		↕	↕	R	P	R	↕
		11 001 011																
		〈 d 〉																
		00 011 110																
	RR (IY+d)	11 111 101			S/D					4	19		↕	↕	R	P	R	↕
		11 001 011																
		〈 d 〉																
		00 011 110																
	RRCA	00 001 111						S/D		1	3		·	·	R	·	R	↕
	RRC g	11 001 011				S/D				2	7		↕	↕	R	P	R	↕
		00 001 g																
	RRC (HL)	11 001 011					S/D			2	13		↕	↕	R	P	R	↕
		00 001 110																
	RRC (IX+d)	11 011 101			S/D					4	19		↕	↕	R	P	R	↕
		11 001 011																
		〈 d 〉																
		00 001 110																
	RRC (IY+d)	11 111 101			S/D					4	19		↕	↕	R	P	R	↕
		11 001 011																
		〈 d 〉																
		00 001 110																

(Continued)

Operation name	MNEMONICS	OP code	Addressing							Bytes	States	Operation	Flag					
													7	6	4	2	1	0
			IMMED	EXT	IND	REG	REGI	IMP	REL				S	Z	H	P/V	N	C
Rotate and Shift Data	RRD	11 101 101 01 100 111						S/D		2	16		↕	↕	R	P	R	•
	SLA g	11 001 011 00 100 g				S/D				2	7		↕	↕	R	P	R	↕
	SLA (HL)	11 001 011 00 100 110					S/D			2	13		↕	↕	R	P	R	↕
	SLA (IX+d)	11 011 101 11 001 011 < d > 00 100 110			S/D					4	19		↕	↕	R	P	R	↕
	SLA (IY+d)	11 111 101 11 001 011 < d > 00 100 110			S/D					4	19		↕	↕	R	P	R	↕
	SRA g	11 001 011 00 101 g				S/D				2	7		↕	↕	R	P	R	↕
	SRA (HL)	11 001 011 00 101 110					S/D			2	13		↕	↕	R	P	R	↕
	SRA (IX+d)	11 011 101 11 001 011 < d > 00 101 110			S/D					4	19		↕	↕	R	P	R	↕
	SRA (IY+d)	11 111 101 11 001 011 < d > 00 101 110			S/D					4	19		↕	↕	R	P	R	↕
	SRL g	11 001 011 00 111 g				S/D				2	7		↕	↕	R	P	R	↕
	SRL (HL)	11 001 011 00 111 110					S/D			2	3		↕	↕	R	P	R	↕
	SRL (IX+d)	11 011 101 11 001 011 < d > 00 111 110			S/D					4	19		↕	↕	R	P	R	↕
	SRL (IY+d)	11 111 101 11 001 011 < d > 00 111 110			S/D					4	19		↕	↕	R	P	R	↕

(3) Bit manipulation instructions

Operation name	MNEMONICS	OP code	Addressing							Bytes	States	Operation	Flag					
			IMMED	EXT	IND	REG	REGI	IMP	REL				7	6	4	2	1	0
													S	Z	H	P/V	N	C
Bit Set	SET b,g	11 001 011 11 b g				S/D				2	7	$1 \to b \cdot gr$	·	·	·	·	·	·
	SET b,(HL)	11 001 011 11 b 110					S/D			2	13	$1 \to b \cdot (HL)_M$	·	·	·	·	·	·
	SET b,(IX+d)	11 011 101 11 001 011 < d > 11 b 110			S/D					4	19	$1 \to b \cdot (IX+d)_M$	·	·	·	·	·	·
	SET b,(IY+d)	11 111 101 11 001 011 < d > 11 b 110			S/D					4	19	$1 \to b \cdot (IY+d)_M$	·	·	·	·	·	·
Bit Reset	RES b,g	11 001 011 10 b g				S/D				2	7	$0 \to b \cdot gr$	·	·	·	·	·	·
	RES b,(HL)	11 001 011 10 b 110					S/D			2	13	$0 \to b \cdot (HL)_M$	·	·	·	·	·	·
	RES b,(IX+d)	11 011 101 11 001 011 < d > 10 b 110			S/D					4	19	$0 \to b \cdot (IX+d)_M$	·	·	·	·	·	·
	RES b,(IY+d)	11 111 101 11 001 011 < d > 10 b 110			S/D					4	19	$0 \to b \cdot (IY+d)_M$	·	·	·	·	·	·
Bit Test	BIT b,g	11 001 011 01 b g				S				2	6	$\overline{b \cdot gr} \to z$	X	↕	S	X	R	·
	BIT b,(HL)	11 001 011 01 b 110					S			2	9	$\overline{b \cdot (HL)_M} \to z$	X	↕	S	X	R	·
	BIT b,(IX+d)	11 011 101 11 001 011 < d > 01 b 110			S					4	15	$\overline{b \cdot (IX+d)_M} \to z$	X	↕	S	X	R	·
	BIT b,(IY+d)	11 111 101 11 001 011 < d > 01 b 110			S					4	15	$\overline{b \cdot (IY+d)_M} \to z$	X	↕	S	X	R	·

3

(4) Arithmetic instructions (16 bits)

Operation name	MNEMONICS	OP code	Addressing							Bytes	States	Operation	Flag					
													7	6	4	2	1	0
			IMMED	EXT	IND	REG	REGI	IMP	REL				S	Z	H	P/V	N	C
ADD	ADD HL,ww	00 ww1 001				S		D		1	7	$HL_R+ww_R \rightarrow HL_R$	·	·	X	·	R	↕
	ADD IX,xx	11 011 101				S		D		2	10	$IX_R+xx_R \rightarrow IX_R$	·	·	X	·	R	↕
		00 xx1 001																
	ADD IY,yy	11 111 101				S		D		2	10	$IY_R+yy_R \rightarrow IY_R$	·	·	X	·	R	↕
		00 yy1 001																
ADC	ADC HL,ww	11 101 101				S		D		2	10	$HL_R+ww_R+c \rightarrow HL_R$	↕	↕	X	V	R	↕
		01 ww1 010																
DEC	DEC ww	00 ww1 011				S/D				1	4	$ww_R-1 \rightarrow ww_R$	·	·	·	·	·	·
	DEC IX	11 011 101						S/D		2	7	$IX_R-1 \rightarrow IX_R$	·	·	·	·	·	·
		00 101 011																
	DEC IY	11 111 101						S/D		2	7	$IY_R-1 \rightarrow IY_R$	·	·	·	·	·	·
		00 101 011																
INC	INC ww	00 ww0 011				S/D				1	4	$ww_R+1 \rightarrow ww_R$	·	·	·	·	·	·
	INC IX	11 011 101						S/D		2	7	$IX_R+1 \rightarrow IX_R$	·	·	·	·	·	·
		00 100 011																
	INC IY	11 111 101						S/D		2	7	$IY_R+1 \rightarrow IY_R$	·	·	·	·	·	·
		00 100 011																
SBC	SBC HL,ww	11 101 101				S		D		2	10	$HL_R-ww_R-c \rightarrow HL_R$	↕	↕	X	V	S	↕
		01 ww0 010																

• Data Transfer Instructions

(1) 8-bit transfer instructions

Operation name	MNEMONICS	OP code	Addressing							Bytes	States	Operation	Flag					
													7	6	4	2	1	0
			IMMED	EXT	IND	REG	REGI	IMP	REL				S	Z	H	P/V	N	C
Load 8-bit Data	LD A,I	11 101 101						S/D		2	6	Ir→Ar *1	↕	↕	R	IEF$_2$	R	·
		01 010 111																
	LD A,R	11 101 101						S/D		2	6	Rr→Ar *1	↕	↕	R	IEF$_2$	R	·
		01 011 111																
	LD A,(BC)	00 001 010					S	D		1	6	(BC)$_M$→Ar	·	·	·	·	·	·
	LD A,(DE)	00 011 010					S	D		1	6	(DE)$_M$→Ar	·	·	·	·	·	·
	LD A,(mn)	00 111 010		S				D		3	12	(mn)$_M$→Ar	·	·	·	·	·	·
		< n >																
		< m >																
	LD I,A	11 101 101						S/D		2	6	Ar→Ir	·	·	·	·	·	·
		01 000 111																
	LD R,A	11 101 101						S/D		2	6	Ar→Rr	·	·	·	·	·	·
		01 001 111																
	LD (BC),A	00 000 010					D	S		1	7	Ar→(BC)$_M$	·	·	·	·	·	·
	LD (DE),A	00 010 010					D	S		1	7	Ar→(DE)$_M$	·	·	·	·	·	·
	LD (mn),A	00 110 010		D				S		3	13	Ar→(mn)$_M$	·	·	·	·	·	·
		< n >																
		< m >																
	LD g,g'	01 g g'				S/D				1	4	gr'→gr	·	·	·	·	·	·
	LD g,(HL)	01 g 110				D	S			1	6	(HL)$_M$→gr	·	·	·	·	·	·
	LD g,m	00 g 110	S			D				2	6	m→gr	·	·	·	·	·	·
		< m >																
	LD g,(IX+d)	11 011 101			S	D				3	14	(IX+d)$_M$→gr	·	·	·	·	·	·
		01 g 110																
		< d >																
	LD g,(IY+d)	11 111 101			S	D				3	14	(IY+d)$_M$→gr	·	·	·	·	·	·
		01 g 110																
		< d >																
	LD (HL),m	00 110 110	S				D			2	9	m→(HL)$_M$	·	·	·	·	·	·
		< m >																
	LD (IX+d),m	11 011 101	S		D					4	15	m→(IX+d)$_M$	·	·	·	·	·	·
		00 110 110																
		< d >																
		< m >																
	LD (IY+d),m	11 111 101	S		D					4	15	m→(IY+d)$_M$	·	·	·	·	·	·
		00 110 110																
		< d >																
		< m >																
	LD (HL),g	01 110 g				S	D			1	7	gr→(HL)$_M$	·	·	·	·	·	·
	LD (IX+d),g	11 011 101			D	S				3	15	gr→(IX+d)$_M$	·	·	·	·	·	·
		01 110 g																
		< d >																
	LD (IY+d),g	11 111 101			D	S				3	15	gr→(IY+d)$_M$	·	·	·	·	·	·
		01 110 g																
		< d >																

*1 No interrupts are sampled at the end of LD A,I or LD A,R instruction.

(2) 16-bit transfer instructions

Operation name	MNEMONICS	OP code	Addressing							Bytes	States	Operation	Flag					
													7	6	4	2	1	0
			IMMED	EXT	IND	REG	REGI	IMP	REL				S	Z	H	P/V	N	C
Load 16-bit Data	LD ww,mn	00 ww0 001 〈 n 〉 〈 m 〉	S			D				3	9	$mn \rightarrow ww_R$	·	·	·	·	·	·
	LD IX,mn	11 011 101 00 100 001 〈 n 〉 〈 m 〉	S					D		4	12	$mn \rightarrow IX_R$	·	·	·	·	·	·
	LD IY,mn	11 111 101 00 100 001 〈 n 〉 〈 m 〉	S					D		4	12	$mn \rightarrow IY_R$	·	·	·	·	·	·
	LD SP,HL	11 111 001						S/D		1	4	$HL_R \rightarrow SP_R$	·	·	·	·	·	·
	LD SP,IX	11 011 101 11 111 001						S/D		2	7	$IX_R \rightarrow SP_R$	·	·	·	·	·	·
	LD SP,IY	11 111 101 11 111 001						S/D		2	7	$IY_R \rightarrow SP_R$	·	·	·	·	·	·
	LD ww,(mn)	11 101 101 01 ww1 011 〈 n 〉 〈 m 〉		S		D				4	18	$(mn+1)_M \rightarrow wwHr$ $(mn)_M \rightarrow wwLr$	·	·	·	·	·	·
	LD HL,(mn)	00 101 010 〈 n 〉 〈 m 〉		S				D		3	15	$(mn+1)_M \rightarrow Hr$ $(mn)_M \rightarrow Lr$	·	·	·	·	·	·
	LD IX,(mn)	11 011 101 00 101 010 〈 n 〉 〈 m 〉		S				D		4	18	$(mn+1)_M \rightarrow IXHr$ $(mn)_M \rightarrow IXLr$	·	·	·	·	·	·
	LD IY,(mn)	11 111 101 00 101 010 〈 n 〉 〈 m 〉		S				D		4	18	$(mn+1)_M \rightarrow IYHr$ $(mn)_M \rightarrow IYLr$	·	·	·	·	·	·
	LD (mn),ww	11 101 101 01 ww0 011 〈 n 〉 〈 m 〉		D		S				4	19	$wwHr \rightarrow (mn+1)_M$ $wwLr \rightarrow (mn)_M$	·	·	·	·	·	·
	LD (mn),HL	00 100 010 〈 n 〉 〈 m 〉		D				S		3	16	$Hr \rightarrow (mn+1)_M$ $Lr \rightarrow (mn)_M$	·	·	·	·	·	·
	LD (mn),IX	11 011 101 00 100 010 〈 n 〉 〈 m 〉		D				S		4	19	$IXHr \rightarrow (mn+1)_M$ $IXLr \rightarrow (mn)_M$	·	·	·	·	·	·
	LD (mn),IY	11 111 101 00 100 010 〈 n 〉 〈 m 〉		D				S		4	19	$IYHr \rightarrow (mn+1)_M$ $IYLr \rightarrow (mn)_M$	·	·	·	·	·	·

(3) Block transfer instructions

Operation name	MNEMONICS	OP code	Addressing							Bytes	States	Operation	Flag					
			IMMED	EXT	IND	REG	REGI	IMP	REL				7	6	4	2	1	0
													S	Z	H	P/V	N	C
Block Transfer Search Data	CPD	11 101 101 10 101 001					S	S		2	12	$Ar-(HL)_M$ $BC_R-1\rightarrow BC_R$ $HL_R-1\rightarrow HL_R$	↕	*3 ↕	↕	*2 ↕	S	·
	CPDR	11 101 101 10 111 001					S	S		2	14 12	$BC_R\neq 0$ $Ar\neq(HL)_M$ $BC_R=0$ or $Ar=(HL)_M$ $Q\begin{cases}Ar-(HL)_M\\BC_R-1\rightarrow BC_R\\HL_R-1\rightarrow HL_R\end{cases}$ Repeat Q until $Ar=(HL)_M$ or $BC_R=0$	↕	*3 ↕	↕	*2 ↕	S	·
	CPI	11 101 101 10 100 001					S	S		2	12	$Ar-(HL)_M$ $BC_R-1\rightarrow BC_R$ $HL_R+1\rightarrow HL_R$	↕	*3 ↕	↕	*2 ↕	S	·
	CPIR	11 101 101 10 110 001					S	S		2	14 12	$BC_R\neq 0$ $Ar\neq(HL)_M$ $BC_R=0$ or $Ar=(HL)_M$ $Q\begin{cases}Ar-(HL)_M\\BC_R-1\rightarrow BC_R\\HL_R+1\rightarrow HL_R\end{cases}$ Repeat Q until $Ar=(HL)_M$ or $BC_R=0$	↕	*3 ↕	↕	*2 ↕	S	·
	LDD	11 101 101 10 101 000					S/D			2	12	$(HL)_M\rightarrow(DE)_M$ $BC_R-1\rightarrow BC_R$ $DE_R-1\rightarrow DE_R$ $HL_R-1\rightarrow HL_R$	·	·	R	*2 ↕	$\overline{R}$	·
	LDDR	11 101 101 10 111 000					S/D			2	14 $(BC_R\neq 0)$ 12 $(BC_R=0)$	$Q\begin{cases}(HL)_M\rightarrow(DE)_M\\BC_R-1\rightarrow BC_R\\DE_R-1\rightarrow DE_R\\HL_R-1\rightarrow HL_R\end{cases}$ Repeat Q until $BC_R=0$	·	·	R	R	R	·
	LDI	11 101 101 10 100 000					S/D			2	12	$(HL)_M\rightarrow(DE)_M$ $BC_R-1\rightarrow BC_R$ $DE_R+1\rightarrow DE_R$ $HL_R+1\rightarrow HL_R$	·	·	R	*2 ↕	R	·
	LDIR	11 101 101 10 110 000					S/D			2	14 $(BC_R\neq 0)$ 12 $(BC_R=0)$	$Q\begin{cases}(HL)_M\rightarrow(DE)_M\\BC_R-1\rightarrow BC_R\\DE_R+1\rightarrow DE_R\\HL_R+1\rightarrow HL_R\end{cases}$ Repeat Q until $BC_R=0$	·	·	R	R	R	·

*2 P/V = 0 : $BC_R-1=0$
P/V = 1 : $BC_R-1\neq 0$

*3 Z = 1 : $Ar=(HL)_M$
Z = 0 : $Ar\neq(HL)_M$

(4) Stack/exchange instructions

Operation name	MNEMONICS	OP code	Addressing							Bytes	States	Operation	Flag					
													7	6	4	2	1	0
			IMMED	EXT	IND	REG	REGI	IMP	REL				S	Z	H	P/V	N	C
PUSH	PUSH zz	11 zz0 101				S		D		1	11	$zzLr\rightarrow(SP-2)_M$ $zzHr\rightarrow(SP-1)_M$ $SP_R-2\rightarrow SP_R$	·	·	·	·	·	·
	PUSH IX	11 011 101 11 100 101						S/D		2	14	$IXLr\rightarrow(SP-2)_M$ $IXHr\rightarrow(SP-1)_M$ $SP_R-2\rightarrow SP_R$	·	·	·	·	·	·
	PUSH IY	11 111 101 11 100 101						S/D		2	14	$IYLr\rightarrow(SP-2)_M$ $IYHr\rightarrow(SP-1)_M$ $SP_R-2\rightarrow SP_R$	·	·	·	·	·	·
POP	POP zz	11 zz0 001				D		S		1	9	$(SP+1)_M\rightarrow zzHr$ *4 $(SP)_M\rightarrow zzLr$ $SP_R+2\rightarrow SP_R$	·	·	·	·	·	·
	POP IX	11 011 101 11 100 001						S/D		2	12	$(SP+1)_M\rightarrow IXHr$ $(SP)_M\rightarrow IXLr$ $SP_R+2\rightarrow SP_R$	·	·	·	·	·	·
	POP IY	11 111 101 11 100 001						S/D		2	12	$(SP+1)_M\rightarrow IYHr$ $(SP)_M\rightarrow IYLr$ $SP_R+2\rightarrow SP_R$	·	·	·	·	·	·
Exchange	EX AF,AF'	00 001 000						S/D		1	4	$AF_R\leftrightarrow AF_R'$	·	·	·	·	·	·
	EX DE,HL	11 101 011						S/D		1	3	$DE_R\leftrightarrow HL_R$	·	·	·	·	·	·
	EXX	11 011 001						S/D		1	3	$BC_R\leftrightarrow BC_R'$ $DE_R\leftrightarrow DE_R'$ $HL_R\leftrightarrow HL_R'$	·	·	·	·	·	·
	EX (SP),HL	11 100 011						S/D		1	16	$Hr\leftrightarrow(SP+1)_M$ $Lr\leftrightarrow(SP)_M$	·	·	·	·	·	·
	EX (SP),IX	11 011 101 11 100 011						S/D		2	19	$IXHr\leftrightarrow(SP+1)_M$ $IXLr\leftrightarrow(SP)_M$	·	·	·	·	·	·
	EX (SP),IY	11 111 101 11 100 011						S/D		2	19	$IYHr\leftrightarrow(SP+1)_M$ $IYLr\leftrightarrow(SP)_M$	·	·	·	·	·	·

*4 POP AF writes the stack contents to the flag.

• Program Control Instructions

Operation name	MNEMONICS	OP code	Addressing							Bytes	States	Operation	Flag					
													7	6	4	2	1	0
			IMMED	EXT	IND	REG	REGI	IMP	REL				S	Z	H	P/V	N	C
Call	CALL mn	11 001 101 〈 n 〉 〈 m 〉		D						3	16	PCHr→$(SP-1)_M$ PCLr→$(SP-2)_M$ mn→PC_R SP_R-2→SP_R	·	·	·	·	·	·
	CALL f,mn	11 f 100 〈 n 〉 〈 m 〉		D						3	6 (f false) 16 (f true)	continue f ıs false CALL mn f ıs true	·	·	·	·	·	·
Jump	DJNZ j	00 010 000 〈j-2〉							D	2 2	9 (Br≠0) 7 (Br=0)	Br−1→Br continue ˙ Br=0 PC_R+j→PC_R Br≠0	·	·	·	·	·	·
	JP f,mn	11 f 010 〈 n 〉 〈 m 〉		D						3 3	6 (f false) 9 (f true)	mn→PC_R f ıs true continue f is false	·	·	·	·	·	·
	JP mn	11 000 011 〈 n 〉 〈 m 〉		D						3	9	mn→PC_R	·	·	·	·	·	·
	JP (HL)	11 101 001					D			1	3	HL_R→PC_R	·	·	·	·	·	·
	JP (IX)	11 011 101 11 101 001					D			2	6	IX_R→PC_R	·	·	·	·	·	·
	JP (IY)	11 111 101 11 101 001					D			2	6	IY_R→PC_R	·	·	·	·	·	·
	JR ȷ	00 011 000 〈j-2〉							D	2	8	PC_R+ȷ→PC_R	·	·	·	·	·	·
	JR C,ȷ	00 111 000 〈ȷ-2〉							D	2 2	6 8	continue . C=0 PC_R+ȷ→PC_R C=1	·	·	·	·	·	·
	JR NC,j	00 110 000 〈ȷ-2〉							D	2 2	6 8	contınue C=1 PC_R+ȷ→PC_R C=0	·	·	·	·	·	·
	JR Z,j	00 101 000 〈j-2〉							D	2 2	6 8	contınue Z=0 PC_R+j→PC_R Z=1	·	·	·	·	·	·
	JR NZ,j	00 100 000 〈j-2〉							D	2 2	6 8	continue . Z=1 PC_R+j→PC_R Z=0	·	·	·	·	·	·
Return	RET	11 001 001						D		1	9	$(SP)_M$→PCLr $(SP+1)_M$→PCHr SP_R+2→SP_R	·	·	·	·	·	·
	RET f	11 f 000							D	1 1	5 (f false) 10 (f true)	continue ˙ f ıs false RET f ıs true	·	·	·	·	·	·
	RETI	11 101 101 01 001 101						D		2	22	$(SP)_M$→PCLr $(SP+1)_M$→PCHr SP_R+2→SP_R	·	·	·	·	·	·
	RETN	11 101 101 01 000 101						D		2	12	$(SP)_M$→PCLr $(SP+1)_M$→PCHr SP_R+2→SP_R IEF_2→IEF_1	·	·	·	·	·	·
Restart	RST v	11 v 111						D		1	11	PCHr→$(SP-1)_M$ PCLr→$(SP-2)_M$ 0→PCHr v→PCLr SP_R 2→SP_R	·	·	·	·	·	·

3

- **I/O Instructions**

Operation name	MNEMONICS	OP code	Addressing							Bytes	States	Operation	Flag					
													7	6	4	2	1	0
			IMMED	EXT	IND	REG	REGI	IMP	I/O				S	Z	H	P/V	N	C
INPUT	IN A,(m)	11 011 011 ⟨ m ⟩						D	S	2	9	$(Am)_I \to Ar$ $m \to A_0 \sim A_7$ $Ar \to A_8 \sim A_{15}$	•	•	•	•	•	•
	IN g,(C)	11 101 101 01 g 000				D			S	2	9	$(BC)_I \to gr$ g=110 : Only the flags will change. $Cr \to A_0 \sim A_7$ $Br \to A_8 \sim A_{15}$	↕	↕	R	P	R	•
	IN0 g,(m)	11 101 101 00 g 000 ⟨ m ⟩				D			S	3	12	$(00m)_I \to gr$ g=110 Only the flags will change. $m \to A_0 \sim A_7$ $00 \to A_8 \sim A_{15}$	↕	↕	R	P	R	•
	IND	11 101 101 10 101 010					D		S	2	12	$(BC)_I \to (HL)_M$ $HL_R - 1 \to HL_R$ $Br - 1 \to Br$ $Cr \to A_0 \sim A_7$ $Br \to A_8 \sim A_{15}$	X	*5 ↕	X	X	*6 ↕	X
	INDR	11 101 101 10 111 010					D		S	2	14(Br≠0) 12(Br=0)	$Q\left\{\begin{array}{l}(BC)_I \to (HL)_M\\ HL_R - 1 \to HL_R\\ Br - 1 \to Br\end{array}\right.$ Repeat Q until Br=0 $Cr \to A_0 \sim A_7$ $Br \to A_8 \sim A_{15}$	X	S	X	X	*6 ↕	X
	INI	11 101 101 10 100 010					D		S	2	12	$(BC)_I \to (HL)_M$ $HL_R + 1 \to HL_R$ $Br - 1 \to Br$ $Cr \to A_0 \sim A_7$ $Br \to A_8 \sim A_{15}$	X	*5 ↕	X	X	*6 ↕	X
	INIR	11 101 101 10 110 010					D		S	2	14(Br≠0) 12(Br=0)	$Q\left\{\begin{array}{l}(BC)_I \to (HL)_M\\ HL_R + 1 \to HL_R\\ Br - 1 \to Br\end{array}\right.$ Repeat Q until Br=0 $Cr \to A_0 \sim A_7$ $Br \to A_8 \sim A_{15}$	X	S	X	X	*6 ↕	X

*5 $Z = 1$: $Br - 1 = 0$
$Z = 0$: $Br - 1 \neq 0$
*6 $N = 1$: MSB of Data = 1
$N = 0$: MSB of Data = 0

(Continued)

Operation name	MNEMONICS	OP code	Addressing							Bytes	States	Operation	Flag					
													7	6	4	2	1	0
			IMMED	EXT	IND	REG	REGI	IMP	I/O				S	Z	H	P/V	N	C
OUTPUT	OUT (m),A	11 010 011 ⟨ m ⟩						S	D	2	10	$Ar \to (Am)_1$ $m \to A_0 \sim A_7$ $Ar \to A_8 \sim A_{15}$	•	•	•	•	•	•
	OUT (C),g	11 101 101 01 g 001				S			D	2	10	$gr \to (BC)_1$ $Cr \to A_0 \sim A_7$ $Br \to A_8 \sim A_{15}$	•	•	•	•	•	•
	OUT0 (m),g	11 101 101 00 g 001 ⟨ m ⟩				S			D	3	13	$gr \to (00m)_1$ $m \to A_0 \sim A_7$ $00 \to A_8 \sim A_{15}$	•	•	•	•	•	•
	OTDM	11 101 101 10 001 011					S		D	2	14	$(HL)_M \to (00C)_1$ $HL_R - 1 \to HL_R$ $Cr - 1 \to Cr$ $Br - 1 \to Br$ $Cr \to A_0 \sim A_7$ $00 \to A_8 \sim A_{15}$	↕	*7 ↕	↕	P	*8 ↕	↕
	OTDMR	11 101 101 10 011 011					S		D	2	16(Br≠0) 14(Br=0)	$Q \begin{cases} (HL)_M \to (00C)_1 \\ HL_R - 1 \to HL_R \\ Cr - 1 \to Cr \\ Br - 1 \to Br \end{cases}$ Repeat Q until Br=0 $Cr \to A_0 \sim A_7$ $00 \to A_8 \sim A_{15}$	R	S	R	S	*8 ↕	R
	OTDR	11 101 101 10 111 011					S		D	2	14(Br≠0) 12(Br=0)	$Q \begin{cases} (HL)_M \to (BC)_1 \\ HL_R - 1 \to HL_R \\ Br - 1 \to Br \end{cases}$ Repeat Q until Br=0 $Cr \to A_0 \sim A_7$ $Br \to A_8 \sim A_{15}$	X	S	X	X	*8 ↕	X
	OUTI	11 101 101 10 100 011					S		D	2	12	$(HL)_M \to (BC)_1$ $HL_R + 1 \to HL_R$ $Br - 1 \to Br$ $Cr \to A_0 \sim A_7$ $Br \to A_8 \sim A_{15}$	X	*7 ↕	X	X	*8 ↕	X
	OTIR	11 101 101 10 110 011					S		D	2	14(Br≠0) 12(Br=0)	$Q \begin{cases} (HL)_M \to (BC)_1 \\ HL_R + 1 \to HL_R \\ Br - 1 \to Br \end{cases}$ Repeat Q until Br=0 $Cr \to A_0 \sim A_7$ $Br \to A_8 \sim A_{15}$	X	S	X	X	*8 ↕	X
	TSTIO m	11 101 101 01 110 100 ⟨ m ⟩	S						S	3	12	$(00C)_1 \cdot m$ $Cr \to A_0 \sim A_7$ $00 \to A_8 \sim A_{15}$	↕	↕	S	P	R	R
	OTIM	11 101 101 10 000 011					S		D	2	14	$(HL)_M \to (00C)_1$ $HL_R + 1 \to HL_R$ $Cr + 1 \to Cr$ $Br - 1 \to Br$ $Cr \to A_0 \sim A_7$ $00 \to A_8 \sim A_{15}$	↕	*7 ↕	↕	P	*8 ↕	↕
	OTIMR	11 101 101 10 010 011					S		D	2	16(Br≠0) 14(Br=0)	$Q \begin{cases} (HL)_M \to (00C)_1 \\ HL_R + 1 \to HL_R \\ Cr + 1 \to Cr \\ Br - 1 \to Br \end{cases}$ Repeat Q until Br=0 $Cr \to A_0 \sim A_7$ $00 \to A_8 \sim A_{15}$	R	S	R	S	*8 ↕	R
	OUTD	11 101 101 10 101 011					S		D	2	12	$(HL)_M \to (BC)_1$ $HL_R - 1 \to HL_R$ $Br - 1 \to Br$ $Cr \to A_0 \sim A_7$ $Br \to A_8 \sim A_{15}$	X	*7 ↕	X	X	*8 ↕	X

*7 Z = 1 : Br − 1 = 0
Z = 0 : Br − 1 ≠ 0
*8 N = 1 : MSB of Data = 1
N = 0 : MSB of Data = 0

• Special Control Instructions

Operation name	MNEMONICS	OP code	Addressing							Bytes	States	Operation	Flag					
													7	6	4	2	1	0
			IMMED	EXT	IND	REG	REGI	IMP	REL				S	Z	H	P/V	N	C
Special Function	DAA	00 100 111						S/D		1	4	Decimal Adjust Accumulator	↕	↕	↕	P	·	↕
Carry Control	CCF	00 111 111								1	3	$\overline{C}$→C	·	·	R	·	R	↕
	SCF	00 110 111								1	3	1→C	·	·	R	·	R	S
CPU Control	DI	11 110 011								1	3	0→IEF_1, 0→IEF_2 *9	·	·	·	·	·	·
	EI	11 111 011								1	3	1→IEF_1, 1→IEF_2 *9	·	·	·	·	·	·
	HALT	01 110 110								1	3	CPU halted	·	·	·	·	·	·
	IM 0	11 101 101 01 000 110								2	6	Interrupt mode 0	·	·	·	·	·	·
	IM 1	11 101 101 01 010 110								2	6	Interrupt mode 1	·	·	·	·	·	·
	IM 2	11 101 101 01 011 110								2	6	Interrupt mode 2	·	·	·	·	·	·
	NOP	00 000 000								1	3	No operation	·	·	·	·	·	·
	SLP	11 101 101 01 110 110								2	8	Sleep	·	·	·	·	·	·

*9 No interrupts are sampled at the end of a DI or EI instruction.

Alphabetical List of Instructions

MNEMONICS	Bytes	Machine Cycles	States
ADC A, m	2	2	6
ADC A, g	1	2	4
ADC A, (HL)	1	2	6
ADC A, (IX+d)	3	6	14
ADC A, (IY+d)	3	6	14
ADD A, m	2	2	6
ADD A, g	1	2	4
ADD A, (HL)	1	2	6
ADD A, (IX+d)	3	6	14
ADD A, (IY+d)	3	6	14
ADC HL, ww	2	6	10
ADD HL, ww	1	5	7
ADD IX, xx	2	6	10
ADD IY, yy	2	6	10
AND m	2	2	6
AND g	1	2	4
AND (HL)	1	2	6
AND (IX+d)	3	6	14
AND (IY+d)	3	6	14
BIT b, (HL)	2	3	9
BIT b, (IX+d)	4	5	15
BIT b, (IY+d)	4	5	15
BIT b, g	2	2	6
CALL f, mn	3	2	6
			(If condition is false)
	3	6	16
			(If condition is true)
CALL mn	3	6	16
CCF	1	1	3
CPD	2	6	12
CPDR	2	8	14
			(If $BC_R \neq 0$ and $Ar \neq (HL)_M$)
	2	6	12
			(If $BC_R = 0$ or $Ar = (HL)_M$)
CP(HL)	1	2	6
CPI	2	6	12
CPIR	2	8	14
			(If $BC_R \neq 0$ and $Ar \neq (HL)_M$)

(Continued)

MNEMONICS	Bytes	Machine Cycles	States
CPIR	2	6	12
			(If $BC_R=0$ or $Ar=(HL)_M$)
CP (IX+d)	3	6	14
CP (IY+d)	3	6	14
CPL	1	1	3
CP m	2	2	6
CP g	1	2	4
DAA	1	2	4
DEC (HL)	1	4	10
DEC IX	2	3	7
DEC IY	2	3	7
DEC (IX+d)	3	8	18
DEC (IY+d)	3	8	18
DEC g	1	2	4
DEC ww	1	2	4
DI	1	1	3
DJNZ j	2	5	9(If Br≠0)
	2	3	7(If Br=0)
EI	1	1	3
EX AF, AF′	1	2	4
EX DE, HL	1	1	3
EX (SP), HL	1	6	16
EX (SP), IX	2	7	19
EX (SP), IY	2	7	19
EXX	1	1	3
HALT	1	1	3
IM 0	2	2	6
IM 1	2	2	6
IM 2	2	2	6
INC g	1	2	4
INC (HL)	1	4	10
INC (IX+d)	3	8	18
INC (IY+d)	3	8	18
INC ww	1	2	4
INC IX	2	3	7
INC IY	2	3	7
IN A, (m)	2	3	9
IN g, (C)	2	3	9
INI	2	4	12
INIR	2	6	14(If Br≠0)

(Continued)

MNEMONICS	Bytes	Machine Cycles	States
INIR	2	4	12(If Br=0)
IND	2	4	12
INDR	2	6	14(If Br≠0)
	2	4	12(If Br=0)
IN0 g, (m)	3	4	12
JP f, mn	3	2	6
			(If f is false)
	3	3	9
			(If f is true)
JP (HL)	1	1	3
JP (IX)	2	2	6
JP (IY)	2	2	6
JP mn	3	3	9
JR j	2	4	8
JR C, j	2	2	6
			(If condition is false)
	2	4	8
			(If condition is true)
JR NC, j	2	2	6
			(If condition is false)
	2	4	8
			(If condition is true)
JR Z, j	2	2	6
			(If condition is false)
	2	4	8
			(If condition is true)
JR NZ, j	2	2	6
			(If condition is false)
	2	4	8
			(If condition is true)
LD A, (BC)	1	2	6
LD A, (DE)	1	2	6
LD A, I	2	2	6
LD A, (mn)	3	4	12
LD A, R	2	2	6
LD (BC), A	1	3	7
LDD	2	4	12
LD (DE), A	1	3	7
LD ww, mn	3	3	9
LD ww, (mn)	4	6	18

(Continued)

MNEMONICS	Bytes	Machine Cycles	States
LDDR	2	6	14 (If $BC_R \neq 0$)
	2	4	12 (If $BC_R = 0$)
LD (HL), m	2	3	9
LD HL, (mn)	3	5	15
LD (HL), g	1	3	7
LDI	2	4	12
LD I, A	2	2	6
LDIR	2	6	14 (If $BC_R \neq 0$)
	2	4	12 (If $BC_R = 0$)
LD IX, mn	4	4	12
LD IX, (mn)	4	6	18
LD (IX+d), m	4	5	15
LD (IX+d), g	3	7	15
LD IY, mn	4	4	12
LD IY, (mn)	4	6	18
LD (IY+d), m	4	5	15
LD (IY+d), g	3	7	15
LD (mn), A	3	5	13
LD (mn), ww	4	7	19
LD (mn), HL	3	6	16
LD (mn), IX	4	7	19
LD (mn), IY	4	7	19
LD R, A	2	2	6
LD g, (HL)	1	2	6
LD g, (IX+d)	3	6	14
LD g, (IY+d)	3	6	14
LD g, m	2	2	6
LD g, g'	1	2	4
LD SP, HL	1	2	4
LD SP, IX	2	3	7
LD SP, IY	2	3	7
MLT ww	2	13	17
NEG	2	2	6
NOP	1	1	3
OR (HL)	1	2	6
OR (IX+d)	3	6	14
OR (IY+d)	3	6	14
OR m	2	2	6
OR g	1	2	4
OTDM	2	6	14

(Continued)

MNEMONICS	Bytes	Machine Cycles	States
OTDMR	2	8	16(If $B_r \neq 0$)
	2	6	14(If $B_r = 0$)
OTDR	2	6	14(If $B_r \neq 0$)
	2	4	12(If $B_r = 0$)
OTIM	2	6	14
OTIMR	2	8	16(If $B_r \neq 0$)
	2	6	14(If $B_r = 0$)
OTIR	2	6	14(If $B_r \neq 0$)
	2	4	12(If $B_r = 0$)
OUTD	2	4	12
OUTI	2	4	12
OUT (m), A	2	4	10
OUT (C), g	2	4	10
OUT0 (m), g	3	5	13
POP IX	2	4	12
POP IY	2	4	12
POP zz	1	3	9
PUSH IX	2	6	14
PUSH IY	2	6	14
PUSH zz	1	5	11
RES b, (HL)	2	5	13
RES b, (IX+d)	4	7	19
RES b, (IY+d)	4	7	19
RES b, g	2	3	7
RET	1	3	9
RET f	1	3	5 (If condition is false)
	1	4	10 (If condition is true)
RETI	2	10	22
RETN	2	4	12
RLA	1	1	3
RLCA	1	1	3
RLC (HL)	2	5	13
RLC (IX+d)	4	7	19
RLC (IY+d)	4	7	19
RLC g	2	3	7
RLD	2	8	16
RL (HL)	2	5	13

(Continued)

MNEMONICS	Bytes	Machine Cycles	States
RL (IX+d)	4	7	19
RL (IY+d)	4	7	19
RL g	2	3	7
RRA	1	1	3
RRCA	1	1	3
RRC (HL)	2	5	13
RRC (IX+d)	4	7	19
RRC (IY+d)	4	7	19
RRC g	2	3	7
RRD	2	8	16
RR (HL)	2	5	13
RR (IX+d)	4	7	19
RR (IY+d)	4	7	19
RR g	2	3	7
RST v	1	5	11
SBC A, (HL)	1	2	6
SBC A, (IX+d)	3	6	14
SBC A, (IY+d)	3	6	14
SBC A, m	2	2	6
SBC A, g	1	2	4
SBC HL, ww	2	6	10
SCF	1	1	3
SET b, (HL)	2	5	13
SET b, (IX+d)	4	7	19
SET b, (IY+d)	4	7	19
SET b, g	2	3	7
SLA (HL)	2	5	13
SLA (IX+d)	4	7	19
SLA (IY+d)	4	7	19
SLA g	2	3	7
SLP	2	2	8
SRA (HL)	2	5	13
SRA (IX+d)	4	7	19
SRA (IY+d)	4	7	19
SRA g	2	3	7
SRL (HL)	2	5	13
SRL (IX+d)	4	7	19
SRL (IY+d)	4	7	19
SRL g	2	3	7
SUB (HL)	1	2	6

(Continued)

MNEMONICS	Bytes	Machine Cycles	States
SUB (IX+d)	3	6	14
SUB (IY+d)	3	6	14
SUB m	2	2	6
SUB g	1	2	4
TSTIO m	3	4	12
TST g	2	3	7
TST m	3	3	9
TST (HL)	2	4	10
XOR (HL)	1	2	6
XOR (IX+d)	3	6	14
XOR (IY+d)	3	6	14
XOR m	2	2	6
XOR g	1	2	4

■ Opcode Maps

Table 1. Opcode Map (1)

First opcode

Instruction format: XX

				ww(LO=ALL)												LO=0~7				
				BC	DE	HL	SP									BC	DE	HL	AF	zz
				g (LO=0~7)												NZ	NC	PO	P	f
				B	D	H	(HL)	B	D	H	(HL)					00H	10H	20H	30H	v
			HI	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111	
		LO		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
S (HI=ALL)	B	0000	0	NOP	DJNZ j	JR NZ,j	JR NC,j	LD g, s			*1	ADD A ,s	SUB s	AND s	OR s	RET f				0
	C	0001	1	LD ww, mn												POP zz				1
	D	0010	2	LD (ww), A		LD (mn) ,HL	LD (mn) ,A									JP f, mn				2
	E	0011	3	INC ww												JP mn	OUT (m) ,A	EX (SP) ,HL	DI	3
	H	0100	4	INC g			*1									CALL f, mn				4
	L	0101	5	DEC g			*1									PUSH zz				5
	(HL)	0110	6	LD g, m			*1	*2			HALT	*2	*2	*2	*2	ADD A,m	SUB m	AND m	OR m	6
	A	0111	7	RLCA	RLA	DAA	SCF									RST v				7
	B	1000	8	EXAF,AF'	JR j	JR Z, j	JR C, j	LD g, s				ADC A ,s	SBC A ,s	XOR s	CP s	RET f				8
	C	1001	9	ADD HL, ww												RET	EXX	JP (HL)	LD SP, HL	9
	D	1010	A	LD A, (ww)		LD HL, (mn)	LD A, (mn)									JP f, mn				A
	E	1011	B	DEC ww												Table 2	IN A, (m)	EXDE, HL	EI	B
	H	1100	C	INC g												CALL f, mn				C
	L	1101	D	DEC g												CALL mn	*3	Table 3	*3	D
	(HL)	1110	E	LD g, m				*2				*2	*2	*2	*2	ADC A,m	SBC A,m	XOR m	CP m	E
	A	1111	F	RRCA	RRA	CPL	CCF									RST v				F
				0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
				C	E	L	A	C	E	L	A					Z	C	PE	M	f
				g(LO=8~F)												08H	18H	28H	38H	v
																LO=8~F				

*1 g is replaced by (HL).

*2 s is replaced by (HL).

*3 If DD is added to the beginning of an opcode (DD XX), only the instructions having HL, (HL) as an operand are replaced with

$$\left\{\begin{matrix} \text{HL} & \rightarrow & \text{IX} \\ \text{(HL)} & \rightarrow & \text{(IX + d)} \end{matrix}\right\}$$

to perform the same operation.

(Example)

22H; LD (mn), HL
↓
DDH 22H; LD (mn), IX

If FD is added to the beginning of the opcode (FD XX), it is replaced by

$$\left\{\begin{matrix} \text{HL} & \rightarrow & \text{IY} \\ \text{(HL)} & \rightarrow & \text{(IY + d)} \end{matrix}\right\}$$

to perform the same operation.

(Example)

34H; INC (HL)
↓
FDH 34H; INC (IY + d)

As an exception, when DDH, FDH is added to the beginning of JP (HL) of E9H, (HL) is replaced by (IX), (IY). If DDH, FDH is added to the beginning of EX DE, HL of EBH, HL is not replaced. It becomes an undefined instruction.

3

Table 2. Opcode Map (2)

Second opcode
Instruction format: CB XX

								b (LO=0~7)												
								0	2	4	6	0	2	4	6	0	2	4	6	
			HI	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111	
		LO		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
g (HI=ALL)	B	0000	0	RLC g	RL g	SLA g		BIT b,g				RES b,g				SET b,g				0
	C	0001	1																	1
	D	0010	2																	2
	E	0011	3																	3
	H	0100	4																	4
	L	0101	5																	5
	(HL)	0110	6	*	*	*		*				*				*				6
	A	0111	7																	7
	B	1000	8	RRC g	RR g	SRA g	SRL g	BIT b,g				RES b,g				SET b,g				8
	C	1001	9																	9
	D	1010	A																	A
	E	1011	B																	B
	H	1100	C																	C
	L	1101	D																	D
	(HL)	1110	E	*	*	*	*	*				*				*				E
	A	1111	F																	F
				0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
								1	3	5	7	1	3	5	7	1	3	5	7	
								b (LO=8~F)												

* In the instruction to be executed, DDH can be added to the beginning of the opcode and (HL) is replaced by (IX + d) in opcode DD CB d XX. In the same way, FDH can be added to the beginning of the opcode. In the instruction to be executed, (HL) is replaced by (IY+ d) in opcode FD CB d XX.

Table 3. Opcode Map (3)

Second opcode

Instruction format: ED XX

						ww (LO=ALL)												
						BC	DE	HL	SP									
		g (LO=0~7)																
		B	D	H		B	D	H										
	HI	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111	
LO		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0000	0	IN0 g, (m)				IN g, (C)						LDI	LDIR					0
0001	1	OUT0 (m),g				OUT (C),g						CPI	CPIR					1
0010	2					SBC HL, ww						INI	INIR					2
0011	3					LD (mn), ww				OTIM	OTIMR	OUTI	OTIR					3
0100	4	TST g			TST (HL)	NEG		TST m	TSTIO m									4
0101	5					RETN												5
0110	6					IM 0	IM 1		SLP									6
0111	7					LD I,A	LD A,I	RRD										7
1000	8	IN0 g, (m)				IN g, (C)						LDD	LDDR					8
1001	9	OUT0 (m),g				OUT (C),g						CPD	CPDR					9
1010	A					ADC HL,ww						IND	INDR					A
1011	B					LD ww, (mn)				OTDM	OTDMR	OUTD	OTDR					B
1100	C	TST g				MLT ww												C
1101	D					RETI												D
1110	E						IM 2											E
1111	F					LD R,A	LD A,R	RLD										F
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
		C	E	L	A	C	E	L	A									
		g (LO=8~F)																

HITACHI

■ Bus Cycle States

'*' in the ADDRESS column indicates that the address output is undefined and 'Z' in the DATA column indicates that the data pin is in the high-impedance state. The $\overline{\text{LIR}}$ pin output value is obtained when the LIRE bit in the operation mode control register is 1.

Instruction	Machine Cycle	States	ADDRESS	DATA	$\overline{\text{RD}}$	$\overline{\text{WR}}$	$\overline{\text{ME}}$	$\overline{\text{IOE}}$	$\overline{\text{LIR}}$	$\overline{\text{HALT}}$	ST
ADD HL,ww	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2 ~MC_5	TiTiTiTi	*	Z	1	1	1	1	1	1	1
ADD IX,xx ADD IY,yy	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC_3 ~MC_6	TiTiTiTi	*	Z	1	1	1	1	1	1	1
ADC HL,ww SBC HL,ww	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC_3 ~MC_6	TiTiTiTi	*	Z	1	1	1	1	1	1	1
ADD A,g ADC A,g SUB g SBC A,g AND g OR g XOR g CP g	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	Ti	*	Z	1	1	1	1	1	1	1
ADD A,m ADC A,m SUB m SBC A,m AND m OR m XOR m CP m	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	1st operand Address	m	0	1	0	1	1	1	1
ADD A, (HL) ADC A, (HL) SUB (HL) SBC A, (HL) AND (HL) OR (HL) XOR (HL) CP (HL)	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	HL	DATA	0	1	0	1	1	1	1
ADD A, (IX+d) ADD A, (IY+d) ADC A, (IX+d) ADC A, (IY+d) SUB (IX+d) SUB (IY+d) SBC A, (IX+d)	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1

(Continued)

Instruction	Machine Cycle	States	ADDRESS	DATA	$\overline{\text{RD}}$	$\overline{\text{WR}}$	$\overline{\text{ME}}$	$\overline{\text{IOE}}$	$\overline{\text{LIR}}$	$\overline{\text{HALT}}$	ST
SBC A, (IY+d) AND (IX+d) AND (IY+d) OR (IX+d) OR (IY+d) XOR (IX+d) XOR (IY+d) CP (IX+d) CP (IY+d)	MC_3	$T_1T_2T_3$	1st operand Address	d	0	1	0	1	1	1	1
	MC_4 ~MC_5	TiTi	*	Z	1	1	1	1	1	1	1
	MC_6	$T_1T_2T_3$	IX+d IY+d	DATA	0	1	0	1	1	1	1
BIT b,g	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
BIT b, (HL)	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	HL	DATA	0	1	0	1	1	1	1
BIT b, (IX+d) BIT b, (IY+d)	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	1st operand Address	d	0	1	0	1	1	1	1
	MC_4	$T_1T_2T_3$	3rd op-code Address	3rd op-code	0	1	0	1	0	1	1
	MC_5	$T_1T_2T_3$	IX+d IY+d	DATA	0	1	0	1	1	1	1
CALL mn	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	1st operand Address	n	0	1	0	1	1	1	1
	MC_3	$T_1T_2T_3$	2nd operand Address	m	0	1	0	1	1	1	1
	MC_4	Ti	*	Z	1	1	1	1	1	1	1
	MC_5	$T_1T_2T_3$	SP−1	PCH	1	0	0	1	1	1	1
	MC_6	$T_1T_2T_3$	SP−2	PCL	1	0	0	1	1	1	1
CALL f,mn (If condition is false)	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	1st operand Address	n	0	1	0	1	1	1	1

(Continued)

Instruction	Machine Cycle	States	ADDRESS	DATA	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{LIR}$	$\overline{HALT}$	ST
CALL f,mn (If condition is true)	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	1st operand Address	n	0	1	0	1	1	1	1
	MC_3	$T_1T_2T_3$	2nd operand Address	m	0	1	0	1	1	1	1
	MC_4	Ti	*	Z	1	1	1	1	1	1	1
	MC_5	$T_1T_2T_3$	SP−1	PCH	1	0	0	1	1	1	1
	MC_6	$T_1T_2T_3$	SP−2	PCL	1	0	0	1	1	1	1
CCF	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
CPI CPD	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	HL	DATA	0	1	0	1	1	1	1
	MC_4 ~MC_6	TiTiTi	*	Z	1	1	1	1	1	1	1
CPIR CPDR (If $BC_R \neq 0$ and $Ar \neq (HL)_M$)	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	HL	DATA	0	1	0	1	1	1	1
	MC_4 ~MC_8	TiTiTiTiTi	*	Z	1	1	1	1	1	1	1
CPIR CPDR (If $BC_R = 0$ or $Ar = (HL)_M$)	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	HL	DATA	0	1	0	1	1	1	1
	MC_4 ~MC_6	TiTiTi	*	Z	1	1	1	1	1	1	1
CPL	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
DAA	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	Ti	*	Z	1	1	1	1	1	1	1
DI *1	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0

*1 No interrupts are sampled at the end of a DI instruction.

(Continued)

Instruction	Machine Cycle	States	ADDRESS	DATA	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{LIR}$	$\overline{HALT}$	ST
DJNZ j (If Br≠0)	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	Ti *2	*	Z	1	1	1	1	1	1	1
	MC_3	$T_1T_2T_3$	1st operand Address	j-2	0	1	0	1	1	1	1
	MC_4 $\sim MC_5$	TiTi	*	Z	1	1	1	1	1	1	1
DJNZ j (If Br=0)	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	Ti *2	*	Z	1	1	1	1	1	1	1
	MC_3	$T_1T_2T_3$	1st operand Address	j-2	0	1	0	1	1	1	1
EI *3	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
EX DE, HL EXX	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
EX AF, AF'	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	Ti	*	Z	1	1	1	1	1	1	1
EX (SP), HL	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	SP	DATA	0	1	0	1	1	1	1
	MC_3	$T_1T_2T_3$	SP+1	DATA	0	1	0	1	1	1	1
	MC_4	Ti	*	Z	1	1	1	1	1	1	1
	MC_5	$T_1T_2T_3$	SP+1	H	1	0	0	1	1	1	1
	MC_6	$T_1T_2T_3$	SP	L	1	0	0	1	1	1	1
EX (SP),IX EX (SP),IY	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	SP	DATA	0	1	0	1	1	1	1
	MC_4	$T_1T_2T_3$	SP+1	DATA	0	1	0	1	1	1	1
	MC_5	Ti	*	Z	1	1	1	1	1	1	1

(Continued)

*2 DMA, refresh, and bus release cannot be executed immediately after this state (their requests are ignored).

*3 No interrupts are sampled at the end of an EI instruction.

Instruction	Machine Cycle	States	ADDRESS	DATA	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{LIR}$	$\overline{HALT}$	ST
EX (SP), IX EX (SP), IY	MC$_6$	$T_1T_2T_3$	SP+1	IXH IYH	1	0	0	1	1	1	1
	MC$_7$	$T_1T_2T_3$	SP	IXL IYL	1	0	0	1	1	1	1
HALT	MC$_1$	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	—	———	Next op-code Address	Next op-code	0	1	0	1	0	0	0
IM 0 IM 1 IM 2	MC$_1$	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
INC g DEC g	MC$_1$	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	Ti	*	Z	1	1	1	1	1	1	1
INC (HL) DEC (HL)	MC$_1$	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	$T_1T_2T_3$	HL	DATA	0	1	0	1	1	1	1
	MC$_3$	Ti	*	Z	1	1	1	1	1	1	1
	MC$_4$	$T_1T_2T_3$	HL	DATA	1	0	0	1	1	1	1
INC (IX+d) INC (IY+d) DEC (IX+d) DEC (IY+d)	MC$_1$	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC$_3$	$T_1T_2T_3$	1st operand Address	d	0	1	0	1	1	1	1
	MC$_4$ ~MC$_5$	TiTi	*	Z	1	1	1	1	1	1	1
	MC$_6$	$T_1T_2T_3$	IX+d IY+d	DATA	0	1	0	1	1	1	1
	MC$_7$	Ti	*	Z	1	1	1	1	1	1	1
	MC$_8$	$T_1T_2T_3$	IX+d IY+d	DATA	1	0	0	1	1	1	1
INC ww DEC ww	MC$_1$	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	Ti	*	Z	1	1	1	1	1	1	1
INC IX INC IY DEC IX DEC IY	MC$_1$	$T_1T_2T_3$	1st op-code Address	1st op-dode	0	1	0	1	0	1	0
	MC$_2$	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC$_3$	Ti	*	Z	1	1	1	1	1	1	1

(Continued)

Instruction	Machine Cycle	States	ADDRESS	DATA	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{LIR}$	$\overline{HALT}$	ST
IN A,(m)	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	1st operand Address	m	0	1	0	1	1	1	1
	MC_3	$T_1T_2T_3$	m to A_0~A_7 A to A_8~A_{15}	DATA	0	1	1	0	1	1	1
IN g,(C)	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	BC	DATA	0	1	1	0	1	1	1
INO g,(m)	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	1st operand Address	m	0	1	0	1	1	1	1
	MC_4	$T_1T_2T_3$	m to A_0~A_7 00H to A_8~A_{15}	DATA	0	1	1	0	1	1	1
INI IND	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	BC	DATA	0	1	1	0	1	1	1
	MC_4	$T_1T_2T_3$	HL	DATA	1	0	0	1	1	1	1
INIR INDR (If Br≠0)	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	BC	DATA	0	1	1	0	1	1	1
	MC_4	$T_1T_2T_3$	HL	DATA	1	0	0	1	1	1	1
	MC_5 ~MC_6	TiTi	*	Z	1	1	1	1	1	1	1
INIR INDR (If Br=0)	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	BC	DATA	0	1	1	0	1	1	1
	MC_4	$T_1T_2T_3$	HL	DATA	1	0	0	1	1	1	1

(Continued)

Instruction	Machine Cycle	States	ADDRESS	DATA	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{LIR}$	$\overline{HALT}$	ST
JP mn	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	1st operand Address	n	0	1	0	1	1	1	1
	MC_3	$T_1T_2T_3$	2nd operand Address	m	0	1	0	1	1	1	1
JP f,mn (If f is false)	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	1st operand Address	n	0	1	0	1	1	1	1
JP f,mn (If f is true)	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	1st operand Address	n	0	1	0	1	1	1	1
	MC_3	$T_1T_2T_3$	2nd operand Address	m	0	1	0	1	1	1	1
JP (HL)	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
JP (IX) JP (IY)	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
JR j	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	1st operand Address	j-2	0	1	0	1	1	1	1
	MC_3 ~MC_4	TiTi	*	Z	1	1	1	1	1	1	1
JR C,j JR NC,j JR Z,j JR NZ,j (If condition is false)	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	1st operand Address	j-2	0	1	0	1	1	1	1
JR C,j JR NC,j JR Z,j JR NZ,j (If condition is true)	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	1st operand Address	j-2	0	1	0	1	1	1	1
	MC_3 ~MC_4	TiTi	*	Z	1	1	1	1	1	1	1
LD g,g′	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	Ti	*	Z	1	1	1	1	1	1	1
LD g,m	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	1st operand Address	m	0	1	0	1	1	1	1

(Continued)

Instruction	Machine Cycle	States	ADDRESS	DATA	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{LIR}$	$\overline{HALT}$	ST
LD g, (HL)	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	HL	DATA	0	1	0	1	1	1	1
LD g, (IX+d) LD g, (IY+d)	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	1st operand Address	d	0	1	0	1	1	1	1
	MC_4 ~MC_5	TiTi	*	Z	1	1	1	1	1	1	1
	MC_6	$T_1T_2T_3$	IX+d IY+d	DATA	0	1	0	1	1	1	1
LD (HL),g	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	Ti	*	Z	1	1	1	1	1	1	1
	MC_3	$T_1T_2T_3$	HL	g	1	0	0	1	1	1	1
LD (IX+d),g LD (IY+d),g	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	1st operand Address	d	0	1	0	1	1	1	1
	MC_4 ~MC_6	TiTiTi	*	Z	1	1	1	1	1	1	1
	MC_7	$T_1T_2T_3$	IX+d IY+d	g	1	0	0	1	1	1	1
LD (HL),m	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	1st operand Address	m	0	1	0	1	1	1	1
	MC_3	$T_1T_2T_3$	HL	DATA	1	0	0	1	1	1	1
LD (IX+d),m LD (IY+d),m	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	1st operand Address	d	0	1	0	1	1	1	1
	MC_4	$T_1T_2T_3$	2nd operand Address	m	0	1	0	1	1	1	1
	MC_5	$T_1T_2T_3$	IX+d IY+d	DATA	1	0	0	1	1	1	1
LD A, (BC) LD A, (DE)	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0

(Continued)

Instruction	Machine Cycle	States	ADDRESS	DATA	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{LIR}$	$\overline{HALT}$	ST
LD A, (BC) LD A, (DE)	MC_2	$T_1T_2T_3$	BC DE	DATA	0	1	0	1	1	1	1
LD A,(mn)	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	1st operand Address	n	0	1	0	1	1	1	1
	MC_3	$T_1T_2T_3$	2nd operand Address	m	0	1	0	1	1	1	1
	MC_4	$T_1T_2T_3$	mn	DATA	0	1	0	1	1	1	1
LD (BC),A LD (DE),A	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	Ti	*	Z	1	1	1	1	1	1	1
	MC_3	$T_1T_2T_3$	BC DE	A	1	0	0	1	1	1	1
LD (mn),A	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	1st operand Address	n	0	1	0	1	1	1	1
	MC_3	$T_1T_2T_3$	2nd operand Address	m	0	1	0	1	1	1	1
	MC_4	Ti	*	Z	1	1	1	1	1	1	1
	MC_5	$T_1T_2T_3$	mn	A	1	0	0	1	1	1	1
LD A,I *4 LD A,R *4 LD I,A LD R,A	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
LD ww, mn	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	1st operand Address	n	0	1	0	1	1	1	1
	MC_3	$T_1T_2T_3$	2nd operand Address	m	0	1	0	1	1	1	1
LD IX,mn LD IY,mn	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	1st operand Address	n	0	1	0	1	1	1	1
	MC_4	$T_1T_2T_3$	2nd operand Address	m	0	1	0	1	1	1	1
LD HL, (mn)	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	1st operand Address	n	0	1	0	1	1	1	1

*4 No interrupts are sampled at the end of a LD A, I or LD A, R instruction.

(Continued)

Instruction	Machine Cycle	States	ADDRESS	DATA	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{LIR}$	$\overline{HALT}$	ST
LD HL, (mn)	MC_3	$T_1T_2T_3$	2nd operand Address	m	0	1	0	1	1	1	1
	MC_4	$T_1T_2T_3$	mn	DATA	0	1	0	1	1	1	1
	MC_5	$T_1T_2T_3$	mn+1	DATA	0	1	0	1	1	1	1
LD ww,(mn)	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	1st operand Address	n	0	1	0	1	1	1	1
	MC_4	$T_1T_2T_3$	2nd operand Address	m	0	1	0	1	1	1	1
	MC_5	$T_1T_2T_3$	mn	DATA	0	1	0	1	1	1	1
	MC_6	$T_1T_2T_3$	mn+1	DATA	0	1	0	1	1	1	1
LD IX,(mn) LD IY,(mn)	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	1st operand Address	n	0	1	0	1	1	1	1
	MC_4	$T_1T_2T_3$	2nd operand Address	m	0	1	0	1	1	1	1
	MC_5	$T_1T_2T_3$	mn	DATA	0	1	0	1	1	1	1
	MC_6	$T_1T_2T_3$	mn+1	DATA	0	1	0	1	1	1	1
LD (mn),HL	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	1st operand Address	n	0	1	0	1	1	1	1
	MC_3	$T_1T_2T_3$	2nd operand Address	m	0	1	0	1	1	1	1
	MC_4	T_1	*	Z	1	1	1	1	1	1	1
	MC_5	$T_1T_2T_3$	mn	L	1	0	0	1	1	1	1
	MC_5	$T_1T_2T_3$	mn+1	H	1	0	0	1	1	1	1

(Continued)

Instruction	Machine Cycle	States	ADDRESS	DATA	$\overline{\text{RD}}$	$\overline{\text{WR}}$	$\overline{\text{ME}}$	$\overline{\text{IOE}}$	$\overline{\text{LIR}}$	$\overline{\text{HALT}}$	ST
LD (mn),ww	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	1st operand Address	n	0	1	0	1	1	1	1
	MC_4	$T_1T_2T_3$	2nd operand Address	m	0	1	0	1	1	1	1
	MC_5	Ti	*	Z	1	1	1	1	1	1	1
	MC_6	$T_1T_2T_3$	mn	wwL	1	0	0	1	1	1	1
	MC_7	$T_1T_2T_3$	mn+1	wwH	1	0	0	1	1	1	1
LD (mn),IX LD (mn),IY	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	1st operand Address	n	0	1	0	1	1	1	1
	MC_4	$T_1T_2T_3$	2nd operand Address	m	0	1	0	1	1	1	1
	MC_5	Ti	*	Z	1	1	1	1	1	1	1
	MC_6	$T_1T_2T_3$	mn	IXL IYL	1	0	0	1	1	1	1
	MC_7	$T_1T_2T_3$	mn+1	IXH IYH	1	0	0	1	1	1	1
LD SP, HL	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	Ti	*	Z	1	1	1	1	1	1	1
LD SP,IX LD SP,IY	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC_3	Ti	*	Z	1	1	1	1	1	1	1
LDI LDD	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	HL	DATA	0	1	0	1	1	1	1
	MC_4	$T_1T_2T_3$	DE	DATA	1	0	0	1	1	1	1

(Continued)

Instruction	Machine Cycle	States	ADDRESS	DATA	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{LIR}$	$\overline{HALT}$	ST
LDIR LDDR (If $BC_R \neq 0$)	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	HL	DATA	0	1	0	1	1	1	1
	MC_4	$T_1T_2T_3$	DE	DATA	1	0	0	1	1	1	1
	MC_5 $\sim MC_6$	TiTi	*	Z	1	1	1	1	1	1	1
LDIR LDDR (If $BC_R = 0$)	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	HL	DATA	0	1	0	1	1	1	1
	MC_4	$T_1T_2T_3$	DE	DATA	1	0	0	1	1	1	1
MLT ww	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC_3 $\sim MC_{13}$	TiTiTiTi TiTiTiTi TiTiTi	*	Z	1	1	1	1	1	1	1
NEG	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
NOP	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
OUT (m),A	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	1st operand Address	m	0	1	0	1	1	1	1
	MC_3	Ti	*	Z	1	1	1	1	1	1	1
	MC_4	$T_1T_2T_3$	m to $A_0 \sim A_7$ A to $A_8 \sim A_{15}$	A	1	0	1	0	1	1	1

(Continued)

Instruction	Machine Cycle	States	ADDRESS	DATA	$\overline{\text{RD}}$	$\overline{\text{WR}}$	$\overline{\text{ME}}$	$\overline{\text{IOE}}$	$\overline{\text{LIR}}$	$\overline{\text{HALT}}$	ST
OUT (C),g	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC_3	T_I	*	Z	1	1	1	1	1	1	1
	MC_4	$T_1T_2T_3$	BC	g	1	0	1	0	1	1	1
OUTO (m),g	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	1st operand Address	m	0	1	0	1	1	1	1
	MC_4	Ti	*	Z	1	1	1	1	1	1	1
	MC_5	$T_1T_2T_3$	m to A_0~A_7 00H to A_8~A_{15}	g	1	0	1	0	1	1	1
OTIM OTDM	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC_3	Ti	*	Z	1	1	1	1	1	1	1
	MC_4	$T_1T_2T_3$	HL	DATA	0	1	0	1	1	1	1
	MC_5	$T_1T_2T_3$	C to A_0~A_7 00H to A_8~A_{15}	DATA	1	0	1	0	1	1	1
	MC_6	Ti	*	Z	1	1	1	1	1	1	1
OTIMR OTDMR (If Br≠0)	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC_3	Ti	*	Z	1	1	1	1	1	1	1
	MC_4	$T_1T_2T_3$	HL	DATA	0	1	0	1	1	1	1
	MC_5	$T_1T_2T_3$	C to A_0~A_7 00H to A_8~A_{15}	DATA	1	0	1	0	1	1	1
	MC_6 ~MC_8	TiTiTi	*	Z	1	1	1	1	1	1	1

(Continued)

Instruction	Machine Cycle	States	ADDRESS	DATA	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{LIR}$	$\overline{HALT}$	ST
OTIMR OTDMR (If Br=0)	MC$_1$	T$_1$T$_2$T$_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	T$_1$T$_2$T$_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC$_3$	T$_i$	*	Z	1	1	1	1	1	1	1
	MC$_4$	T$_1$T$_2$T$_3$	HL	DATA	0	1	0	1	1	1	1
	MC$_5$	T$_1$T$_2$T$_3$	C to A$_0$~A$_7$ 00H to A$_8$~A$_{15}$	DATA	1	0	1	0	1	1	1
	MC$_6$	T$_i$	*	Z	1	1	1	1	1	1	1
OUTI OUTD	MC$_1$	T$_1$T$_2$T$_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	T$_1$T$_2$T$_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC$_3$	T$_1$T$_2$T$_3$	HL	DATA	0	1	0	1	1	1	1
	MC$_4$	T$_1$T$_2$T$_3$	BC	DATA	1	0	1	0	1	1	1
OTIR OTDR (If Br≠0)	MC$_1$	T$_1$T$_2$T$_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	T$_1$T$_2$T$_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC$_3$	T$_1$T$_2$T$_3$	HL	DATA	0	1	0	1	1	1	1
	MC$_4$	T$_1$T$_2$T$_3$	BC	DATA	1	0	1	0	1	1	1
	MC$_5$ ~MC$_6$	T$_i$T$_i$	*	Z	1	1	1	1	1	1	1
OTIR OTDR (If Br=0)	MC$_1$	T$_1$T$_2$T$_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	T$_1$T$_2$T$_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC$_3$	T$_1$T$_2$T$_3$	HL	DATA	0	1	0	1	1	1	1
	MC$_4$	T$_1$T$_2$T$_3$	BC	DATA	1	0	1	0	1	1	1
POP zz	MC$_1$	T$_1$T$_2$T$_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	T$_1$T$_2$T$_3$	SP	DATA	0	1	0	1	1	1	1
	MC$_3$	T$_1$T$_2$T$_3$	SP+1	DATA	0	1	0	1	1	1	1
POP IX POP IY	MC$_1$	T$_1$T$_2$T$_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0

(Continued)

Instruction	Machine Cycle	States	ADDRESS	DATA	$\overline{\text{RD}}$	$\overline{\text{WR}}$	$\overline{\text{ME}}$	$\overline{\text{IOE}}$	$\overline{\text{LIR}}$	$\overline{\text{HALT}}$	ST
POP IX POP IY	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	SP	DATA	0	1	0	1	1	1	1
	MC_4	$T_1T_2T_3$	SP+1	DATA	0	1	0	1	1	1	1
PUSH zz	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2 $\sim MC_3$	TiTi	*	Z	1	1	1	1	1	1	1
	MC_4	$T_1T_2T_3$	SP−1	zzH	1	0	0	1	1	1	1
	MC_5	$T_1T_2T_3$	SP−2	zzL	1	0	0	1	1	1	1
PUSH IX PUSH IY	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC_3 $\sim MC_4$	TiTi	*	Z	1	1	1	1	1	1	1
	MC_5	$T_1T_2T_3$	SP−1	IXH IYH	1	0	0	1	1	1	1
	MC_6	$T_1T_2T_3$	SP−2	IXL IYL	1	0	0	1	1	1	1
RET	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	SP	DATA	0	1	0	1	1	1	1
	MC_3	$T_1T_2T_3$	SP+1	DATA	0	1	0	1	1	1	1
RET f (If condition is false)	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2 $\sim MC_3$	TiTi	*	Z	1	1	1	1	1	1	1
RET f (If condition is true)	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	Ti	*	Z	1	1	1	1	1	1	1
	MC_3	$T_1T_2T_3$	SP	DATA	0	1	0	1	1	1	1
	MC_4	$T_1T_2T_3$	SP+1	DATA	0	1	0	1	1	1	1
RETN	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	SP	DATA	0	1	0	1	1	1	1
	MC_4	$T_1T_2T_3$	SP+1	DATA	0	1	0	1	1	1	1

(Continued)

Instruction	Machine Cycle	States	ADDRESS	DATA	$\overline{\text{RD}}$	$\overline{\text{WR}}$	$\overline{\text{ME}}$	$\overline{\text{IOE}}$	$\overline{\text{LIR}}$	$\overline{\text{HALT}}$	ST
RETI	MC$_1$	T$_1$T$_2$T$_3$	1st op-code Address	1st op-code	0	1	0	1	0 *5 1	1	0
	MC$_2$	T$_1$T$_2$T$_3$	2nd op-code Address	2nd op-code	0	1	0	1	0 *5 1	1	1
	MC$_3$ ~MC$_5$	T$_i$T$_i$T$_i$	*	Z	1	1	1	1	1 *5 1	1	1
	MC$_6$	T$_1$T$_2$T$_3$	1st op-code Address	1st op-code	0	1	0	1	0 *5 0	1	1
	MC$_7$	T$_i$	*	Z	1	1	1	1	1 *5 1	1	1
	MC$_8$	T$_1$T$_2$T$_3$	2nd op-code Address	2nd op-code	0	1	0	1	0 *5 0	1	1
	MC$_9$	T$_1$T$_2$T$_3$	SP	data	0	1	0	1	1 *5 1	1	1
	MC$_{10}$	T$_1$T$_2$T$_3$	SP+1	data	0	1	0	1	1 *5 1	1	1
RLCA RLA RRCA RRA	MC$_1$	T$_1$T$_2$T$_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
RLC g RL g RRC g RR g SLA g SRA g SRL g	MC$_1$	T$_1$T$_2$T$_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	T$_1$T$_2$T$_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC$_3$	T$_i$	*	Z	1	1	1	1	1	1	1
RLC (HL) RL (HL) RRC (HL) RR (HL) SLA (HL) SRA (HL) SRL (HL)	MC$_1$	T$_1$T$_2$T$_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	T$_1$T$_2$T$_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC$_3$	T$_1$T$_2$T$_3$	HL	DATA	0	1	0	1	1	1	1
	MC$_4$	T$_i$	*	Z	1	1	1	1	1	1	1
	MC$_5$	T$_1$T$_2$T$_3$	HL	DATA	1	0	0	1	1	1	1

(Continued)

*5 The upper column indicates the $\overline{\text{LIR}}$ pin value when the LIRE bit in the operation mode control register is 1, and the lower column indicates that when the same bit is 0.

Instruction	Machine Cycle	States	ADDRESS	DATA	$\overline{\text{RD}}$	$\overline{\text{WR}}$	$\overline{\text{ME}}$	$\overline{\text{IOE}}$	$\overline{\text{LIR}}$	$\overline{\text{HALT}}$	ST
RLC (IX+d) RLC (IY+d) RL (IX+d) RL (IY+d) RRC (IX+d) RRC (IY+d) RR (IX+d) RR (IY+d) SLA (IX+d) SLA (IY+d) SRA (IX+d) SRA (IY+d) SRL (IX+d) SRL (IY+d)	MC$_1$	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC$_3$	$T_1T_2T_3$	1st operand Address	d	0	1	0	1	1	1	1
	MC$_4$	$T_1T_2T_3$	3rd op-code Address	3rd op-code	0	1	0	1	0	1	1
	MC$_5$	$T_1T_2T_3$	IX+d IY+d	DATA	0	1	0	1	1	1	1
	MC$_6$	Ti	*	Z	1	1	1	1	1	1	1
	MC$_7$	$T_1T_2T_3$	IX+d IY+d	DATA	1	0	0	1	1	1	1
RLD RRD	MC$_1$	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC$_3$	$T_1T_2T_3$	HL	DATA	0	1	0	1	1	1	1
	MC$_4$ ~MC$_7$	TiTiTiTi	*	Z	1	1	1	1	1	1	1
	MC$_8$	$T_1T_2T_3$	HL	DATA	1	0	0	1	1	1	1
RST v	MC$_1$	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC$_2$ ~MC$_3$	TiTi	*	Z	1	1	1	1	1	1	1
	MC$_4$	$T_1T_2T_3$	SP−1	PCH	1	0	0	1	1	1	1
	MC$_5$	$T_1T_2T_3$	SP−2	PCL	1	0	0	1	1	1	1
SCF	MC$_1$	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
SET b,g RES b,g	MC$_1$	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC$_3$	Ti	*	Z	1	1	1	1	1	1	1
SET b, (HL) RES b, (HL)	MC$_1$	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC$_3$	$T_1T_2T_3$	HL	DATA	0	1	0	1	1	1	1
	MC$_4$	Ti	*	Z	1	1	1	1	1	1	1
	MC$_5$	$T_1T_2T_3$	HL	DATA	1	0	0	1	1	1	1

(Continued)

Instruction	Machine Cycle	States	ADDRESS	DATA	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{LIR}$	$\overline{HALT}$	ST

SET b, (IX+d)
SET b, (IY+d)
RES b, (IX+d)
RES b, (IY+d)
	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0

	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	1st operand Address	d	0	1	0	1	1	1	1
	MC_4	$T_1T_2T_3$	3rd op-code Address	3rd op-code	0	1	0	1	0	1	1
	MC_5	$T_1T_2T_3$	IX+d IY+d	DATA	0	1	0	1	1	1	1
	MC_6	T_i	*	Z	1	1	1	1	1	1	1
	MC_7	$T_1T_2T_3$	IX+d IY+d	DATA	1	0	0	1	1	1	1
SLP	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	——	——	FFFFFH	Z	1	1	1	1	1	0	1
TSTIO m	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	1st operand Address	m	0	1	0	1	1	1	1
	MC_4	$T_1T_2T_3$	C to $A_0 \sim A_7$ OOH to $A_8 \sim A_{15}$	DATA	0	1	1	0	1	1	1
TST g	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC_3	T_i	*	Z	1	1	1	1	1	1	1
TST m	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	1st operand Address	m	0	1	0	1	1	1	1
TST (HL)	MC_1	$T_1T_2T_3$	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC_3	T_i	*	Z	1	1	1	1	1	1	1
	MC_4	$T_1T_2T_3$	HL	DATA	0	1	0	1	1	1	1

(Continued)

Interrupts

Instruction	Machine Cycle	States	ADDRESS	DATA	$\overline{\text{RD}}$	$\overline{\text{WR}}$	$\overline{\text{ME}}$	$\overline{\text{IOE}}$	$\overline{\text{LIR}}$	$\overline{\text{HALT}}$	ST
$\overline{\text{NMI}}$	MC_1	$T_1T_2T_3$	Next op-code Address (PC)		0	1	0	1	0	1	0
	MC_2 $\sim MC_3$	TiTi	*	Z	1	1	1	1	1	1	1
	MC_4	$T_1T_2T_3$	SP−1	PCH	1	0	0	1	1	1	1
	MC_5	$T_1T_2T_3$	SP−2	PCL	1	0	0	1	1	1	1
$\overline{\text{INT}}_0$ MODE 0 (RST INSERTED)	MC_1	$T_1T_2T_W$ T_WT_3	Next op-code Address (PC)	1st op-code	1	1	1	0	0	1	0
	MC_2 $\sim MC_3$	TiTi	*	Z	1	1	1	1	1	1	1
	MC_4	$T_1T_2T_3$	SP−1	PCH	1	0	0	1	1	1	1
	MC_5	$T_1T_2T_3$	SP−2	PCL	1	0	0	1	1	1	1
$\overline{\text{INT}}_0$ MODE 0 (CALL INSERTED)	MC_1	$T_1T_2T_W$ T_WT_3	Next op-code Address (PC)	1st op-code	1	1	1	0	0	1	0
	MC_2	$T_1T_2T_3$	PC	n	0	1	0	1	1	1	1
	MC_3	$T_1T_2T_3$	PC+1	m	0	1	0	1	1	1	1
	MC_4	Ti	*	Z	1	1	1	1	1	1	1
	MC_5	$T_1T_2T_3$	SP−1	PC+2(H)	1	0	0	1	1	1	1
	MC_6	$T_1T_2T_3$	SP−2	PC+2(L)	1	0	0	1	1	1	1
$\overline{\text{INT}}_0$ MODE 1	MC_1	$T_1T_2T_W$ T_WT_3	Next op-code Address (PC)		1	1	1	0	0	1	0
	MC_2	$T_1T_2T_3$	SP−1	PCH	1	0	0	1	1	1	1
	MC_3	$T_1T_2T_3$	SP−2	PCL	1	0	0	1	1	1	1
$\overline{\text{INT}}_0$ MODE 2	MC_1	$T_1T_2T_W$ T_WT_3	Next op-code Address (PC)	Vector	1	1	1	0	0	1	0
	MC_2	Ti	*	Z	1	1	1	1	1	1	1
	MC_3	$T_1T_2T_3$	SP−1	PCH	1	0	0	1	1	1	1
	MC_4	$T_1T_2T_3$	SP−2	PCL	1	0	0	1	1	1	1
	MC_5	$T_1T_2T_3$	I, vector	DATA	0	1	0	1	1	1	1
	MC_6	$T_1T_2T_3$	I, vector + 1	DATA	0	1	0	1	1	1	1

(Continued)

Instruction	Machine Cycle	States	ADDRESS	DATA	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{LIR}$	$\overline{HALT}$	ST
$\overline{INT_1}$ $\overline{INT_2}$ Internal interrupts	MC_1	$T_1T_2T_W$ T_WT_3	Next op-code Address (PC)		1	1	1	1	1	1	0
	MC_2	Ti	*	Z	1	1	1	1	1	1	1
	MC_3	$T_1T_2T_3$	SP−1	PCH	1	0	0	1	1	1	1
	MC_4	$T_1T_2T_3$	SP−2	PCL	1	0	0	1	1	1	1
	MC_5	$T_1T_2T_3$	I, vector	DATA	0	1	0	1	1	1	1
	MC_6	$T_1T_2T_3$	I, vector + 1	DATA	0	1	0	1	1	1	1

■ Requests in Each Operating Mode

Current Status		Operation Requests		
			Interrupt Requests	
Chip Operation Mode	Operation Cycle	$\overline{\text{WAIT}}$	$\overline{\text{NMI}}$	$\overline{\text{INT}_0}$–$\overline{\text{INT}_2}$, or Internal Interrupt
Normal operation mode	CPU	Accepted	Accepted at end of instruction	Accepted at end of instruction
	Interrupt acknowledge cycle	Accepted	Not accepted	Not accepted
	DMA	Accepted	Accepted; DMA cycle aborted	Accepted
	Refresh	Accepted *1	Accepted *3	Accepted *3
	Bus release mode	Not accepted	Accepted *2	Accepted *2
Halt mode	DMA	Accepted	Accepted; DMA cycle aborted and halt mode released	Accepted
	Refresh	Accepted *1	Accepted; halt mode released after completion of refresh cycle*3	Accepted; halt mode released after completion of refresh cycle*3
	Bus release mode	Not accepted	Accepted; halt mode released after completion of bus release mode*2	Accepted; halt mode released after completion of bus release mode*2
	Other halt mode	Accepted	Accepted; halt mode released	Accepted; halt mode released
Sleep mode	DMA	Accepted	Accepted; DMA cycle aborted and sleep mode released	Accepted
	Refresh	Accepted *1	Accepted; sleep mode released after completion of refresh cycle*3	Accepted; sleep mode released after completion of refresh cycle*3
	Bus release mode	Not accepted	Accepted; sleep mode released after completion of bus release mode*2	Accepted; sleep mode released after completion of bus release mode*2
	Other sleep mode	Not accepted	Accepted; sleep mode released	Accepted; sleep mode released

■ Requests in Each Operating Mode (cont.)

Current Status		Operation Requests		
			Interrupt Requests	
Chip Operation Mode	Operation Cycle	$\overline{\text{WAIT}}$	$\overline{\text{NMI}}$	$\overline{\text{INT}_0}$–$\overline{\text{INT}_2}$, or Internal Interrupt
System stop mode	Bus release mode	Not accepted	Accepted; system stop mode released after completion of bus release mode*2	Accepted; system stop mode released after completion of bus release mode*2
	Other system stop mode	Not accepted	Accepted; system stop mode released	Accepted; system stop mode released
Reset mode	----------	Not accepted	Not accepted	Not accepted

■ Requests in Each Operating Mode (cont.)

Current Status		Operation Requests		
		Bus Requests		
Chip Operation Mode	Operation Cycle	$\overline{\text{BUSREQ}}$	Refresh Request	DMA Request from $\overline{\text{DREQ}_0}$, $\overline{\text{DREQ}_1}$, or MSCI
Normal operation mode	CPU	Accepted; enters bus release mode at end of machine cycle	Accepted; refresh cycle executed at end of machine cycle	Accepted; DMA cycle executed at end of machine cycle
	Interrupt acknowledge cycle	Accepted; enters bus release mode at end of machine cycle	Accepted; refresh cycle executed at end of machine cycle	Accepted; DMA cycle executed at end of machine cycle
	DMA	Accepted; enters bus release mode at end of machine cycle	Accepted; refresh cycle executed at end of machine cycle	Accepted; DMA cycle executed at end of machine cycle
	Refresh	Accepted; enters bus release mode at end of machine cycle *3	Accepted; refresh cycle executed at end of machine cycle	Accepted; DMA cycle executed at end of machine cycle *3
	Bus release mode	Bus release mode continues	Accepted; refresh cycle executed after completion of bus release mode*2	Accepted; DMA cycle executed after completion of bus release mode *2
Halt mode	DMA	Accepted; bus release mode entered at end of machine cycle	Accepted; refresh cycle executed at end of machine cycle	Accepted; DMA cycle executed at end of machine cycle
	Refresh	Accepted; bus release mode entered at end of machine cycle *3	Accepted; refresh cycle executed at end of machine cycle	Accepted; DMA cycle executed at end of machine cycle *3
	Bus release mode	Bus release mode continues	Accepted; refresh cycle executed after completion of bus release mode*2	Accepted; DMA cycle executed after completion of bus release mode *2
	Other halt mode	Accepted; bus release mode entered at end of machine cycle	Accepted; refresh cycle executed at end of machine cycle	Accepted; DMA cycle executed at end of machine cycle

■ Requests in Each Operating Mode (cont.)

Current Status		Operation Requests		
Chip Operation Mode	Operation Cycle	Bus Requests		
		BUSREQ	Refresh Request	DMA Request from DREQ0, DREQ1, or MSCI
Sleep mode	DMA	Accepted; bus release mode entered at end of machine cycle	Accepted; refresh cycle executed at end of machine cycle	Accepted; DMA cycle executed at end of machine cycle
	Refresh	Accepted; bus release mode entered at end of machine cycle *3	Accepted; refresh cycle executed at end of machine cycle	Accepted; DMA cycle executed at end of machine cycle *3
	Bus release mode	Bus release mode continues	Accepted; refresh cycle executed after completion of bus release mode*2	Accepted; DMA cycle executed after completion of bus release mode *2
	Other sleep mode	Accepted; enters bus release mode	Accepted; refresh cycle executed at end of machine cycle	Accepted; DMA cycle executed at end of machine cycle
System stop mode	Bus release mode	Bus release mode continues	Not accepted	Not accepted
	Other system stop mode	Accepted; enters bus release mode	Not accepted	Not accepted
Reset mode	-------------	Not accepted	Not accepted	Not accepted

*1 Not accepted when the number of programmable wait states is 0.

*2 Requests are held until the bus release mode completes.

*3 Requests are held until the refresh cycle completes.

■ Request Priorities

Requests to the HD64180S are categorized into three types:

① Requests accepted and executed in each state ······ $\overline{\text{WAIT}}$

② Requests accepted and executed in each machine cycle ······ Refresh request
DMA request
$\overline{\text{BUSREQ}}$ request

③ Requests accepted and executed in each instruction ······ Interrupts

In principle, request priorities are as follows:

(High) ① > ② > ③ (Low)

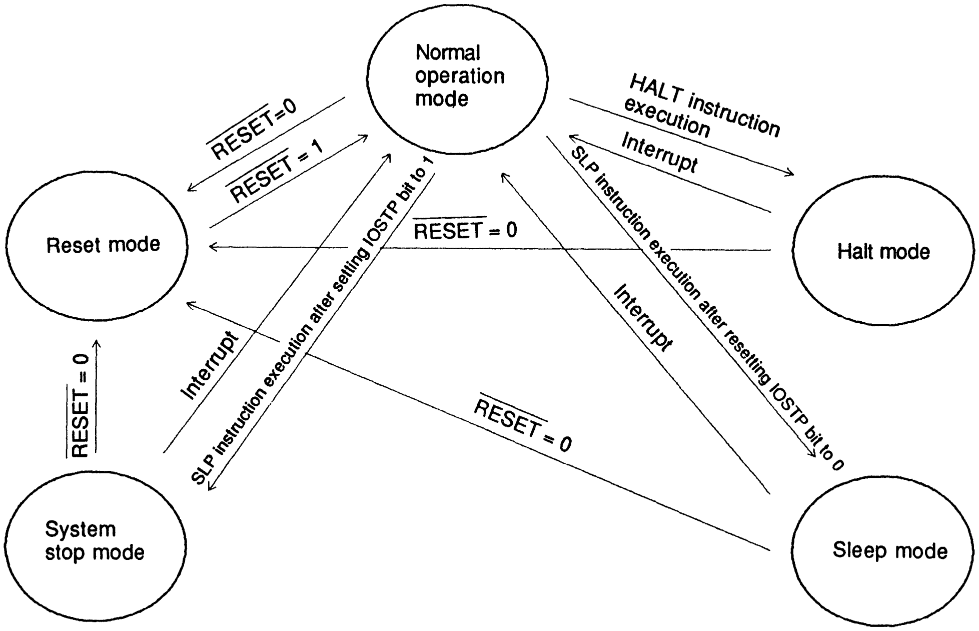
Type ② requests are prioritized as follows:

(High) $\overline{\text{BUSREQ}}$ > Refresh request > DMA request (Low)

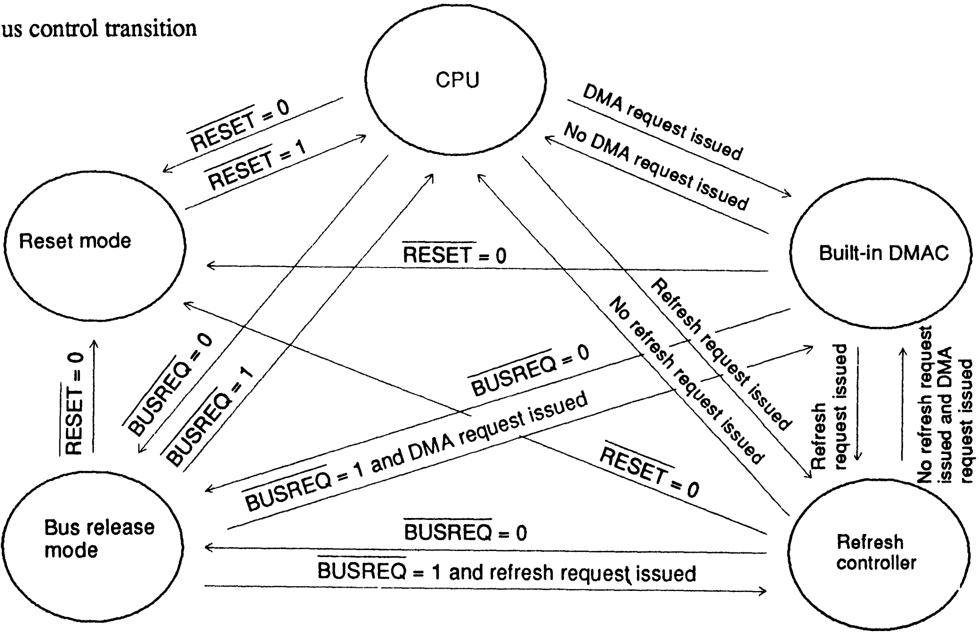
For the priority of type ③ requests, see section 3.6 "Interrupts"

■ State Transition Diagrams

(1) Chip operation mode transition diagram


Normal operation mode
RESET=0
RESET = 1
HALT instruction execution
Interrupt
Reset mode
RESET = 0
Halt mode
SLP instruction execution after setting IOSTP bit to 1
SLP instruction execution after resetting IOSTP bit to 0
Interrupt
Interrupt
RESET = 0
RESET = 0
System stop mode
Sleep mode

(2) Bus control transition



■ Status Signals

Status signals are listed below.

Chip operation mode	Operation cycle		$\overline{LIR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{RD}$	$\overline{WR}$	$\overline{REF}$	$\overline{HALT}$	$\overline{BUSACK}$	ST	$\overline{CS_{0\sim2}}$	$A_0 \sim A_{19}$	$D_0 \sim D_7$
Normal operation mode	CPU	First opcode fetch	0	0	1	0	1	1	1	1	0	OUT (A)	OUT (A)	IN
		Second and third opcode fetch	0	0	1	0	1	1	1	1	1	OUT (A)	OUT (A)	IN
		Memory read	1	0	1	0	1	1	1	1	1	OUT (A)	OUT (A)	IN
		Memory write	1	0	1	1	0	1	1	1	1	OUT (A)	OUT (A)	OUT (A)
		I/O read	1	1	0	0	1	1	1	1	1	1	OUT (A)	IN
		I/O write	1	1	0	1	0	1	1	1	1	1	OUT (A)	OUT (A)
		Internal operation	1	1	1	1	1	1	1	1	1	1	OUT (A)	Z
	Interrupt acknowl-edge(first machine cycle)	$\overline{NMI}$	0	0	1	0	1	1	1	1	0	OUT (A)	OUT (A)	IN
		$\overline{INT_0}$	0	1	0	1	1	1	1	1	0	1	OUT (A)	IN
		$\overline{INT_1}$, $\overline{INT_2}$, and internal interrupts	1	1	1	1	1	1	1	1	0	1	OUT (A)	Z
	Internal DMA	Memory read	1	0	1	0	1	1	1	1	0	OUT (A)	OUT (A)	IN
		Memory write	1	0	1	1	0	1	1	1	0	OUT (A)	OUT (A)	OUT (A)
		I/O read	1	1	0	0	1	1	1	1	0	1	OUT (A)	IN
		I/O write	1	1	0	1	0	1	1	1	0	1	OUT (A)	OUT (A)
		Internal operation	1	1	1	1	1	1	1	1	0	1	OUT (A)	Z
	Refresh		1	0	1	1	1	0	1	1	1	1	OUT (A)	Z
	Bus release mode		1	Z	Z	Z	Z	1	1	0	1	1	Z	Z

1:	High level output
0:	Low level output
OUT (A):	Any output
IN:	Input
Z:	High impedance

■ Status Signals (cont.)

Chip operation mode	Operation cycle		$\overline{LIR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{RD}$	$\overline{WR}$	$\overline{REF}$	$\overline{HALT}$	$\overline{BUSACK}$	ST	$\overline{CS_{0\sim2}}$	$A_0 \sim A_{19}$	$D_0 \sim D_7$
Halt mode	Internal DMA	Memory read	1	0	1	0	1	1	0	1	0	OUT (A)	OUT (A)	IN
		Memory write	1	0	1	1	0	1	0	1	0	OUT (A)	OUT (A)	OUT (A)
		I/O read	1	1	0	0	1	1	0	1	0	1	OUT (A)	IN
		I/O write	1	1	0	1	0	1	0	1	0	1	OUT (A)	OUT (A)
		Internal operation	1	1	1	1	1	1	0	1	0	1	OUT (A)	Z
	Refresh		1	0	1	1	1	0	0	1	1	1	OUT (A)	Z
	Bus release mode		1	Z	Z	Z	Z	1	0	0	1	1	Z	Z
	Halt mode other than above		0	0	1	0	1	1	0	1	0	OUT (A)	OUT (A)	IN
Sleep mode	Internal DMA	Memory read	1	0	1	0	1	1	0	1	0	OUT (A)	OUT (A)	IN
		Memory write	1	0	1	1	0	1	0	1	0	OUT (A)	OUT (A)	OUT (A)
		I/O read	1	1	0	0	1	1	0	1	0	1	OUT (A)	IN
		I/O write	1	1	0	1	0	1	0	1	0	1	OUT (A)	OUT (A)
		Internal operation	1	1	1	1	1	1	0	1	0	1	OUT (A)	Z
	Refresh		1	0	1	1	1	0	0	1	1	1	OUT (A)	Z
	Bus release mode		1	Z	Z	Z	Z	1	0	0	1	1	Z	Z
	Sleep mode other than above		1	1	1	1	1	1	0	1	1	1	1	Z
System stop mode	Bus release mode		1	Z	Z	Z	Z	1	0	0	1	1	Z	Z
	System stop mode other than above		1	1	1	1	1	1	0	1	1	1	1	Z
Reset mode	———		1	1	1	1	1	1	1	1	1	1	Z	Z

1:	High level output
0:	Low level output
OUT (A):	Any output
IN:	Input
Z:	High impedance

■ Pin States in Reset and Low Power Dissipation Modes

Pin name		Pin state		
		Reset mode	Sleep mode	System stop mode
TIN_0, TIN_1		IN (N)	IN (A)	IN (N)
$TOUT_0$, $TOUT_1$		OUT (L)	OUT (A)	HOLD
$\overline{CS_0}$, $\overline{CS_1}$, $\overline{CS_2}$		OUT (H)	OUT (A)	OUT (H)
$\overline{WAIT}$		IN (N)	IN (A)	IN (N)
$\overline{NMI}$		IN (N)	IN (A)	IN (A)
$\overline{INT_0}$, $\overline{INT_1}$, $\overline{INT_2}$		IN (N)	IN (A)	IN (A)
$\overline{RESET}$		IN (A)	IN (A)	IN (A)
$\overline{BUSREQ}$		IN (N)	IN (A)	IN (A)
$\overline{BUSACK}$		OUT (H)	OUT (A)	OUT (A)
ST		OUT (H)	OUT (A)	OUT (H)
$\overline{LIR}$		OUT (H)	OUT (H)	OUT (H)
$\overline{REF}$		OUT (H)	OUT (A)	OUT (H)
$\overline{HALT}$		OUT (H)	OUT (L)	OUT (L)
$\overline{RD}$		OUT (H)	OUT (A)	OUT (H)
$\overline{WR}$		OUT (H)	OUT (A)	OUT (H)
$\overline{ME}$		OUT (H)	OUT (A)	OUT (H)
$\overline{IOE}$		OUT (H)	OUT (A)	OUT (H)
$A_0 \sim A_{19}$		Z	OUT (A)	OUT (H)
$D_0 \sim D_7$		Z	IN (A), OUT (A), Z	Z
$\overline{SYNC}$	Input selected	IN (N)	IN (A)	IN (N)
	Output selected	———	OUT (A)	HOLD
$\overline{RTSM}$		OUT (H)	OUT (A)	HOLD
$\overline{DCDM}$		IN (N)	IN (A)	IN (N)
$\overline{CTSM}$		IN (N)	IN (A)	IN (N)
RXDM		IN (N)	IN (A)	IN (N)

IN (A):	Input (active)
IN (N):	Input (inactive)
OUT (H):	Output (fixed to high level)
OUT (L):	Output (fixed to low level)
OUT (A):	Output (active) - High or low level output
Z:	High impedance
HOLD:	Holding the previous state

■ Pin States in Reset and Low Power Dissipation Modes (cont.)

Pin name		Pin state		
		Reset mode	Sleep mode	System stop mode
RXCM	Input selected	IN (N)	IN (A)	IN (N)
	Output selected	———	OUT (A)	OUT (A)
TXCM	Input selected	IN (N)	IN (A)	IN (N)
	Output selected	———	OUT (A)	OUT (A)
TXDM		OUT (H)	OUT (A)	OUT (H)
$\overline{\text{RTSA}}$		OUT (H)	OUT (A)	HOLD
$\overline{\text{DCDA}}$		IN (N)	IN (A)	IN (N)
$\overline{\text{CTSA}}$		IN (N)	IN (A)	IN (N)
RXDA		IN (N)	IN (A)	IN (N)
RXCA	Input selected	IN (N)	IN (A)	IN (N)
	Output selected	———	OUT (A)	OUT (H)
TXCA	Input selected	IN (N)	IN (A)	IN (N)
	Output selected	———	OUT (A)	OUT (H)
TXDA		OUT (H)	OUT (A)	HOLD
$\overline{\text{DREQ}_0}$, $\overline{\text{DREQ}_1}$		IN (N)	IN (A)	IN (N)
$\overline{\text{TEND}_0}$, $\overline{\text{TEND}_1}$		OUT (H)	OUT (A)	OUT (H)
ϕ		ø clock output	ø clock output	ø clock output

IN (A):	Input (active)
IN (N):	Input (inactive)
OUT (H):	Output (fixed to high level)
OUT (L):	Output (fixed to low level)
OUT (A):	Output (active) - High or low level output
Z:	High impedance
HOLD:	Holding the previous state

■ Built-in Registers

CPU

Register **Address** **Remarks**

Interrupt control register (ICR) 0000H

	7	6	5	4	3	2	1	0
Bit Name	TRAP	UFO	–	–	–	–	–	ITE0
Read/Write	R/W	R	–	–	–	–	–	R/W
Initial Value	0	0	0	0	0	0	0	1

TRAP Status
0: TRAP interrupt not generated
1. TRAP interrupt generated

Undefined Fetch Object Code
0: Second byte of opcode undefined
1: Third byte of opcode undefined

$\overline{\text{INT}}_0$ Enable
0: $\overline{\text{INT}}_0$ disabled
1: $\overline{\text{INT}}_0$ enabled

MMU common base register (CBR) 0001H

	7	6	5	4	3	2	1	0
Bit Name	CB7	CB6	CB5	CB4	CB3	CB2	CB1	CB0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

MMU bank base register (BBR) 0002H

	7	6	5	4	3	2	1	0
Bit Name	BB7	BB6	BB5	BB4	BB3	BB2	BB1	BB0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

MMU common/bank area register (CBAR) 0003H

	7	6	5	4	3	2	1	0
Bit Name	CA3	CA2	CA1	CA0	BA3	BA2	BA1	BA0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	1	1	1	1	0	0	0	0

Four high order bits of the lower address limit for common area 1

Four high order bits of the lower address limit for the bank area

Operation mode control register (OMCR) 0004H

	7	6	5	4	3	2	1	0
Bit Name	LIRE	$\overline{\text{LIRTE}}$	$\overline{\text{IOC}}$	–	–	–	–	–
Read/Write	R/W	W	R/W	–	–	–	–	–
Initial Value	1	1	1	0	0	0	0	0

$\overline{\text{LIR}}$ Enable
0: The $\overline{\text{LIR}}$ output is low only during the opcode fetch cycle 2 of the RETI instruction and the first machine cycle of the $\overline{\text{INT}}_0$ interrupt acknowledge cycle.
1. Normal operation

I/O Compatibility
0: Output of the $\overline{\text{IOE}}$ and $\overline{\text{RD}}$ lines is compatible with that of the Z80-based peripheral LSIs.
1: Normal operation

$\overline{\text{LIR}}$ Temporary Enable
0. When the LIRE bit is 0, the $\overline{\text{LIR}}$ output is low only for the opcode fetch cycle immediately after 0 is written to the $\overline{\text{LIRTE}}$ bit.
1: Normal operation

■ Built-in Registers (cont.)

CPU

Register	Address	Remarks
I/O control register (IOCR)	0005H	

	7	6	5	4	3	2	1	0
Bit Name	IOSTP	–	–	–	–	–	–	–
Read/Write	R/W	–	–	–	–	–	–	–
Initial Value	0	0	0	0	0	0	0	0

I/O Stop
0 Sleep mode (SLP instruction execution)
1 System stop mode (SLP instruction execution)

Register	Address	Remarks
Unused	0006H	
Unused	0007H	

Wait Control

Register	Address	Remarks
Physical address boundary register 0 (PABR0)	0008H	

	7	6	5	4	3	2	1	0
Bit Name	PB07	PB06	PB05	PB04	PB03	PB02	PB01	PB00
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

PAL/PAM Boundary Address (8 high-order bits)

Register	Address	Remarks
Physical address boundary register 1 (PABR1)	0009H	

	7	6	5	4	3	2	1	0
Bit Name	PB17	PB16	PB15	PB14	PB13	PB12	PB11	PB10
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

PAM/PAH Boundary Address (8 high-order bits)

Register	Address	Remarks
Wait control register L (WCRL)	000AH	

	7	6	5	4	3	2	1	0
Bit Name	–	–	–	–	–	PALW2	PALW1	PALW0
Read/Write	–	–	–	–	–	R/W	R/W	R/W
Initial Value	0	0	0	0	0	1	1	1

PAL Area Wait

PALW2	PALW1	PALW0	Number of Wait States
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

■ Built-in Registers (cont.)

Wait Control

Register	Address	Remarks
Wait control register M (WCRM)	000BH	

	7	6	5	4	3	2	1	0
Bit Name	–	–	–	–	–	PAMW2	PAMW1	PAMW0
Read/Write	–	–	–	–	–	R/W	R/W	R/W
Initial Value	0	0	0	0	0	1	1	1
							PAM Area Wait	

PAMW2	PAMW1	PAMW0	Number of Wait States
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

Register	Address	Remarks
Wait control register H (WCRH)	000CH	

	7	6	5	4	3	2	1	0
Bit Name	–	–	–	–	–	PAHW2	PAHW1	PAHW0
Read/Write	–	–	–	–	–	R/W	R/W	R/W
Initial Value	0	0	0	0	0	1	1	1
							PAH Area Wait	

PAHW2	PAHW1	PAHW0	Number of Wait States
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

Register	Address	Remarks
I/O wait control register (IOWCR)	000DH	

	7	6	5	4	3	2	1	0
Bit Name	–	IOH2	IOH1	IOH0	–	IOL2	IOL1	IOL0
Read/Write	–	R/W	R/W	R/W	–	R/W	R/W	R/W
Initial Value	0	1	1	1	0	1	1	1
			I/O High				I/O Low	

IOH2	IOH1	IOH0	Number of Wait States
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

IOL2	IOL1	IOL0	Number of Wait States
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

■ Built-in Registers (cont.)

Wait Control

Register	Address	Remarks
Interrupt wait control register (INTWR)	000EH	

	7	6	5	4	3	2	1	0
Bit Name	–	–	–	–	–	INTW2	INTW1	INTW0
Read/Write	–	–	–	–	–	R/W	R/W	R/W
Initial Value	0	0	0	0	0	1	1	1

Interrupt Wait

INTW2	INTW1	INTW0	Number of Wait States
0	0	0	2
0	0	1	3
0	1	0	4
0	1	1	5
1	0	0	6
1	0	1	7
1	1	0	8
1	1	1	9

Register	Address	Remarks
Refresh wait control register (RWCR)	000FH	

	7	6	5	4	3	2	1	0
Bit Name	–	–	–	–	–	REFW2	REFW1	REFW0
Read/Write	–	–	–	–	–	R/W	R/W	R/W
Initial Value	0	0	0	0	0	1	1	1

Refresh Wait

REFW2	REFW1	REFW0	Number of Wait States
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

Interrupt Control

Register	Address	Remarks
Interrupt status register 0 (ISR0)	0010H	

	7	6	5	4	3	2	1	0
Bit Name	TXRDY1	RXRDY1	TXINT0	RXINT0	TXRDY0	RXRDY0	INT2	INT1
Read/Write	R	R	R	R	R	R	R	R
Initial Value	0	0	0	0	0	0	X	X

ASCI/CSIO TXRDY
0: Request not issued
1. Request issued

ASCI/CSIO RXRDY
0: Request not issued
1: Request issued

MSCI TXINT
0: Request not issued
1: Request issued

MSCI RXINT
0: Request not issued
1: Request issued

MSCI TXRDY
0: Request not issued
1: Request issued

MSCI RXRDY
0: Request not issued
1. Request issued

External Interrupt $\overline{\text{INT}}_2$
0: Request not issued
1: Request issued

External Interrupt $\overline{\text{INT}}_1$
0: Request not issued
1. Request issued

■ Built-in Registers (cont.)

Interrupt Control

Register | **Address** | **Remarks**

Interrupt status register 1 (ISR1) 0011H

	7	6	5	4	3	2	1	0
Bit Name	T1IRQ	T0IRQ	DMIB1	DMIA1	DMIB0	DMIA0	TXINT1	RXINT1
Read/Write	R	R	R	R	R	R	R	R
Initial Value	0	0	0	0	0	0	0	0

Timer Channel 1 Interrupt Request
0 Request not issued
1: Request issued

Timer Channel 0 Interrupt Request
0: Request not issued
1: Request issued

DMA Interrupt B Channel 1
0· Request not issued
1: Request issued

DMA Interrupt A Channel 1
0: Request not issued
1: Request issued

DMA Interrupt B Channel 0
0: Request not issued
1: Request issued

DMA Interrupt A Channel 0
0: Request not issued
1: Request issued

ASCI/CSIO TXINT
0: Request not issued
1: Request issued

ASCI/CSIO RXINT
0: Request not issued
1· Request issued

Interrupt enable register 0 (IER0) 0012H

	7	6	5	4	3	2	1	0
Bit Name	TXRDY1E	RXRDY1E	TXINT0E	RXINT0E	TXRDY0E	RXRDY0E	INT2E	INT1E
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

ASCI/CSIO TXRDY Enable
0: Disabled
1: Enabled

ASCI/CSIO RXRDY Enable
0: Disabled
1. Enabled

MSCI TXINT Enable
0: Disabled
1: Enabled

MSCI RXINT Enable
0: Disabled
1: Enabled

MSCI TXRDY Enable
0: Disabled
1: Enabled

MSCI RXRDY Enable
0: Disabled
1: Enabled

$\overline{\mathrm{INT}_2}$ External Interrupt Enable
0: Disabled
1 Enabled

$\overline{\mathrm{INT}_1}$ External Interrupt Enable
0: Disabled
1: Enabled

Interrupt enable register 1 (IER1) 0013H

	7	6	5	4	3	2	1	0
Bit Name	T1IRQE	T0IRQE	DMIB1E	DMIA1E	DMIB0E	DMIA0E	TXINT1E	RXINT1E
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Timer Channel 1 Interrupt Request Enable
0: Disabled
1: Enabled

Timer Channel 0 Interrupt Request Enable
0: Disabled
1 Enabled

DMA Interrupt B Channel 1 Enable
0· Disabled
1· Enabled

DMA Interrupt A Channel 1 Enable
0: Disabled
1: Enabled

DMA Interrupt B Channel 0 Enable
0: Disabled
1: Enabled

DMA Interrupt A Channel 0 Enable
0: Disabled
1. Enabled

ASCI/CSIO TXINT Enable
0 Disabled
1: Enabled

ASCI/CSIO RXINT Enable
0: Disabled
1: Enabled

Interrupt vector low register (IL) 0014H

	7	6	5	4	3	2	1	0
Bit Name	IL7	IL6	–	–	–	–	–	–
Read/Write	R/W	R/W	–	–	–	–	–	–
Initial Value	0	0	0	0	0	0	0	0

Fixed code

Low order byte of vector address

Built-in Registers (cont.)

Interrupt Control

Register	Address	Remarks
Unused	0015H	
Unused	0016H	
Unused	0017H	

Refresh Control

Register	Address	Remarks
Refresh control register (RCR)	0018H	

	7	6	5	4	3	2	1	0
Bit Name	REFE	–	–	–	–	CYC2	CYC1	CYC0
Read/Write	R/W	–	–	–	–	R/W	R/W	R/W
Initial Value	1	0	0	0	0	0	0	0

Refresh Enable

0· Refresh cycles not inserted
1· Refresh cycles inserted

Cycle Select

- Insertion interval

000: 32 states
001: 64 states
010: 96 states
011: 128 states
100· 160 states
101. 192 states
110: 224 states
111: 256 states

Unused	0019H	
Unused	001AH	
Unused	001BH	

Bus Control

Register	Address	Remarks
DMA priority control register (PCR)	001CH	

	7	6	5	4	3	2	1	0
Single-block Transfer Mode (dual address) Single-block Transfer Mode (single address) Chained-block Transfer Mode	–	–	–	–	–	–	–	PR0
Read/Write	–	–	–	–	–	–	–	R/W
Initial Value	0	0	0	0	0	0	0	0

Channel Priority

0: Channel 0 has priority over channel 1
1: Channel 1 has priority over channel 0

Register	Address	Remarks
DMA master enable register (DMER)	001DH	

	7	6	5	4	3	2	1	0
Single-block Transfer Mode (dual address) Single-block Transfer Mode (single address) Chained-block Transfer Mode	DME	–	–	–	–	–	–	–
Read/Write	R/W	–	–	–	–	–	–	–
Initial Value	1	0	0	0	0	0	0	0

DMA Master-Enable

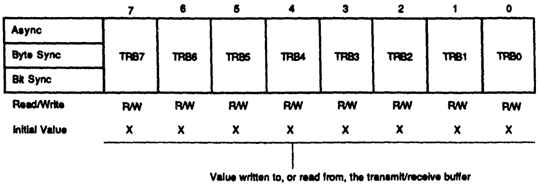
0 Disable
1 Enable

■ Built-in Registers (cont.)

Bus Control

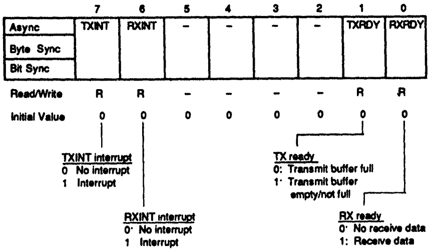
Register	Address	Remarks
Unused	001EH	
Unused	001FH	

MSCI

Register	Address	Remarks
MSCI TX/RX buffer register (MTRB)	0020H	

	7	6	5	4	3	2	1	0
Async Byte Sync Bit Sync	TRB7	TRB6	TRB5	TRB4	TRB3	TRB2	TRB1	TRB0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	X	X	X	X	X	X	X	X

Value written to, or read from, the transmit/receive buffer

MSCI status register 0 (MST0)	0021H	

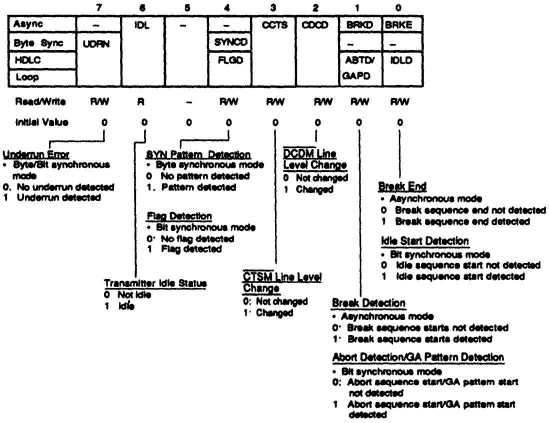
	7	6	5	4	3	2	1	0
Async Byte Sync Bit Sync	TXINT	RXINT	–	–	–	–	TXRDY	RXRDY
Read/Write	R	R	–	–	–	–	R	R
Initial Value	0	0	0	0	0	0	0	0

TXINT interrupt
0 No interrupt
1 Interrupt

RXINT interrupt
0· No interrupt
1 Interrupt

TX ready
0: Transmit buffer full
1· Transmit buffer empty/not full

RX ready
0· No receive data
1: Receive data

MSCI status register 1 (MST1)	0022H	

	7	6	5	4	3	2	1	0
Async	–	IDL	–	–	CCTS	CDCD	BRKD	BRKE
Byte Sync	UDRN			SYNCD			–	–
HDLC				FLGD			ABTD/ GAPD	IDLD
Loop								
Read/Write	R/W	R	–	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Underrun Error
- Byte/Bit synchronous mode
0. No underrun detected
1 Underrun detected

Transmitter Idle Status
0 Not idle
1 Idle

SYN Pattern Detection
- Byte synchronous mode
0 No pattern detected
1. Pattern detected

Flag Detection
- Bit synchronous mode
0· No flag detected
1 Flag detected

CTSM Line Level Change
0: Not changed
1· Changed

DCDM Line Level Change
0 Not changed
1 Changed

Break Detection
- Asynchronous mode
0· Break sequence starts not detected
1· Break sequence starts detected

Abort Detection/GA Pattern Detection
- Bit synchronous mode
0: Abort sequence start/GA pattern start not detected
1 Abort sequence start/GA pattern start detected

Break End
- Asynchronous mode
0 Break sequence end not detected
1 Break sequence end detected

Idle Start Detection
- Bit synchronous mode
0 Idle sequence start not detected
1 Idle sequence start detected

HITACHI

■ Built-in Registers (cont.)

MSCI

Register | **Address** | **Remarks**

MSCI status register 2 (MST2) 0023H

	7	6	5	4	3	2	1	0
Async	–	PMP	PE	FRME	OVRN	–	–	–
Byte Sync		–	–	–		CRCE		
Bit Sync	EOM	SHRT	ABT	RBIT				
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	–	–
Initial Value	0	0	0	0	0	0	0	0

End of Receive Frame
- Bit synchronous mode
0: Receive frame end not detected
1 End of receive frame detected

Parity/MP Bit
- Asynchronous mode
0· Parity or MP bit "0"
1 Paritv or MP bit "1"

Short Frame
- Bit synchronous mode
0: Normal end of frame
1: Short frame detected

Framing Error
- Asynchronous mode
0: No framing error detected
1: Framing error detected

Residue Bit Frame
- Bit synchronous mode
0· Normal end of frame
1: Frame with residue bits detected

Parity Error
- Asynchronous mode
0: No parity error detected
1 Parity error detected

Abort End Frame
- Bit synchronous mode
0 Normal end of frame
1 Frame with abort end detected

CRC Error
- Byte/Bit synchronous mode
0 No CRC error detected
1 CRC error detected

Overrun Error
0: No overrun error detected
1· Overrun error detected

MSCI status register 3 (MST3) 0024H

	7	6	5	4	3	2	1	0
Async	–	–	–	–	$\overline{CTS}$	$\overline{DCD}$	TXENBL	RXENBL
Byte Sync				SRCH				
Bit Sync			SLOOP					
Read/Write	–	–	R	R	R	R	R	R
Initial Value	0	0	0	0	X	X	0	0

Sending on Loop
- Bit synchronous mode
0· Transmits no MSCI data
1: Transmits MSCI data

$\overline{CTSM}$ Input Line Status
0. $\overline{CTSM}$ low level
1: $\overline{CTSM}$ high level

TX Enable
0: Disable
1. Enable

Search Mode
- Byte/Bits synchronous mode
0. ADPLL normal mode
1 ADPLL search mode

RX Enable
0: Disable
1 Enable

$\overline{DCDM}$ Input Line Status
0: $\overline{DCDM}$ low level
1 $\overline{DCDM}$ high level

MSCI frame status register (MFST) 0025H

	7	6	5	4	3	2	1	0
Async	–	–	–	–	–	–	–	–
Byte Sync								
Bit Sync	EOMF	SHRTF	ABTF	RBITF	OVRNF	CRCEF		
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	–	–
Initial Value	0	0	0	0	0	0	0	0

Frame status at receive completion

3

■ Built-in Registers (cont.)

MSCI

Register	Address	Remarks

MSCI interrupt enable register 0 (MIE0) 0026H

	7	6	5	4	3	2	1	0
Async	TXINTE	RXINTE	–	–	–	–	TXRDYE	RXRDYE
Byte Sync								
Bit Sync								
Read/Write	R/W	R/W	–	–	–	–	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

TXINT Interrupt Enable
0 Disable
1 Enable

RXINT Interrupt Enable
0. Disable
1 Enable

TXRDY Interrupt Enable
0. Disable
1 Enable

RXRDY Interrupt Enable
0 Disable
1 Enable

MSCI interrupt enable register 1 (MIE1) 0027H

	7	6	5	4	3	2	1	0
Async	–	IDLE	–	–	CCTSE	CDCDE	BRKDE	BRKEE
Byte Sync	UDRNE			SYNCDE			–	–
HDLC				FLGDE			ABTDE/ GAPDE	IDLDE
Loop								
Read/Write	R/W	R/W	–	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

IDL Interrupt Enable
0: Disable
1· Enable

CCTS Interrupt Enable
0. Disable
1 Enable

BRKD Interrupt Enable
- Asynchronous mode
0 Disable
1 Enable

UDRN Interrupt Enable
- Byte/Bit synchronous mode
0: Disable
1 Enable

ABTD/GAPD Interrupt Enable
- Bit synchronous mode
0. Disable
1· Enable

SYNCD Interrupt Enable
- Byte synchronous mode
0: Disable
1: Enable

FLGD Interrupt Enable
- Bit synchronous mode
0: Disable
1 Enable

CDCD Interrupt Enable
0: Disable
1. Enable

BRKE Interrupt Enable
- Asynchronous mode
0 Disable
1 Enable

IDLD Interrupt Enable
- Bit synchronous mode
0. Disable
1· Enable

MSCI interrupt enable register 2 (MIE2) 0028H

	7	6	5	4	3	2	1	0
Async	–	PMPE	PEE	FRMEE	OVRNE	–	–	–
Byte Sync		–	–	–		CRCEE		
Bit Sync	EOME	SHRTE	ABTE	RBITE				
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	–	–
Initial Value	0	0	0	0	0	0	0	0

EOM Interrupt Enable
- Bit synchronous mode
0: Disable
1: Enable

CRCE Interrupt Enable
- Byte/bit synchronous mode
0. Disable
1: Enable

OVRN Interrupt Enable
0: Disable
1. Enable

PMP Interrupt Enable
- Asynchronous mode
0· Disable
1. Enable

FRME Interrupt Enable
- Asynchronous mode
0. Disable
1· Enable

SHRT Interrupt Enable
- Bit synchronous mode
0: Disable
1· Enable

PE Interrupt Enable
- Asynchronous mode
0· Disable
1. Enable

RBIT Interrupt Enable
- Bit synchronous mode
0 Disable
1· Enable

ABT Interrupt Enable
- Bit synchronous mode
0 Disable
1· Enable

■ Built-in Registers (cont.)

MSCI

Register	Address	Remarks
MSCI frame interrupt enable register (MFIE)	0029H	

	7	6	5	4	3	2	1	0
Async	–	–	–	–	–	–	–	–
Byte Sync								
Bit Sync	EOMFE							
Read/Write	R/W	–	–	–	–	–	–	–
Initial Value	0	0	0	0	0	0	0	0

EOMF Interrupt Enable
- Bit synchronous mode

0: Disable
1: Enable

Register	Address	Remarks
MSCI command register (MCMD)	002AH	

	7	6	5	4	3	2	1	0
Async	–	–	CMD5	CMD4	CMD3	CMD2	CMD1	CMD0
Byte Sync								
Bit Sync								
Read/Write	–	–	W	W	W	W	W	W
Initial Value	–	–	–	–	–	–	–	–

Command

- Transmit Commands

000001: TX reset
000010: TX enable
000011: TX disable
000100: TX CRC initialization
000101: Exclusion from TX CRC calculation
000110: End of message
000111: Abort transmission
001000· MP bit on
001001: TX buffer clear
Others: Reserved

- Receive Commands

010001: RX reset
010010. RX enable
010011: RX disable
010100: RX CRC initialization
010101· Message reject
010110: Search MP bit
010111 Exclusion from RX CRC calculation
011000: Forcing RX CRC calculation

- Other Commands

100001· Channel reset
110001. Enter search mode
000000 No operation

Register	Address	Remarks
MSCI ıııode register 0 (MMD0)	002BH	

	7	6	5	4	3	2	1	0
Async	PRTCL2	PRTCL1	PRTCL0	AUTO	–	–	STOP1	STOP0
Byte Sync						CRCCC	CRC1	CRC0
Bit Sync								
Read/Write	R/W	R/W	R/W	R/W	–	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Protocol Mode

000	Asynchronous mode
001·	Byte-sync, Mono-sync mode
010	Byte-sync, Bi-sync mode
011.	Byte-sync, External synchronous mode
100	Bit-sync, HDLC mode
101	Bit-sync, Loop mode
110	Reserved
111	Reserved

Auto-Enable

0 Auto-enable reset
1 Auto-enable set

CRC Calculation Code
- Byte/Bit synchronous mode

0 Disable
1. Enable

Stop Bit Length
- Asynchronous mode

00	1 bit
01	1 5 bits
10	2 bits
11	Reserved

CRC Calculation Expression and Initial Value
- Byte/Bit synchronous mode

0X	CRC-16
1X:	CRC-CCITT
X0	Initial value = all 0s
X1	Initial value = all 1s

■ Built-in Registers (cont.)

MSCI

Register | **Address** | **Remarks**

MSCI mode register 1 (MMD1) 002CH

	7	6	5	4	3	2	1	0
Async	BRATE1	BRATE0	TXCHR1	TXCHR0	RXCHR1	RXCHR0	PMPM1	PMPM0
Byte Sync	–	–	–	–	–	–	–	–
Bit Sync	ADDRS1	ADDRS0						
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Bit Rate
- Asynchronous mode

00 1/1 clock rate
01 1/16 clock rate
10 1/32 clock rate
11 1/64 clock rate

Address Field Check
- Bit synchronous mode

00 Address field no-check
01 Single address 1
10: Single address 2'
11 Dual address

Transmit Character Length
- Asynchronous mode

00 8 bits/character
01 7 bits/character
10 6 bits/character
11 5 bits/character

Receive Character Length
- Asynchronous mode

00 8 bits/character
01 7 bits/character
10 6 bits/character
11 5 bits/character

Parity/Multiprocessor Mode
- Asynchronous mode

00 No parity/MP bit
01 MP bit appended (by command)
10 Even parity appended and checked
11 Odd parity appended and checked

MSCI mode register. 2 (MMD2) 002DH

	7	6	5	4	3	2	1	0
Async	–	–	–	–	–	–		
Byte Sync	NRZFM	CODE1	CODE0	DRATE1	DRATE0		CNCT1	CNCT0
Bit Sync								
Read/Write	R/W	R/W	R/W	R/W	R/W	–	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

NRZ or FM Select
- Byte/Bit synchronous mode

0 NRZ
1 FM

Transmission Code Type
- Byte/Bit synchronous mode
- NRZ

00. NRZ
01 NRZI
10 Reserved
11 Reserved
- FM

00 Manchester
01 FM1
10 FM0
11 Reserved

ADPLL Operating Clock/Bit Rate
- Byte/Bit synchronous mode

00 x8
01 x16
10 x32
11 Reserved

Channel Connection
00 Full duplex communications
01 Auto echo
10 Reserved
11 Local loop back

MSCI control register (MCTL) 002EH

	7	6	5	4	3	2	1	0
Async	TXRDYC	–	–	–	BRK	–	–	$\overline{\text{RTS}}$
Byte Sync			UDRNC	IDLC	–	SYNCLD		
Bit Sync						–	GOP	
Read/Write	R/W	–	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	1

TX Ready State Control
0 TXRDY bit goes to 1 when the transmit buffer is empty
1. TXRDY bit goes to 1 when the transmit buffer is not full

Underrun State Control
- Byte synchronous mode

0 Enters idle state immediately
1 Enters idle state after CRC transmission
- Bit synchronous mode

0· Enters idle state after aborting transmission
1 Enters idle state after FCS and flag transmission

Idle State Control
- Byte/Bit synchronous mode

0. Transmits a mark
1 Transmits an idle pattern

Send Break
- Asynchronous mode

0· Off
1· On (break send)

SYN Character Load Enable
- Byte sync mode

0· Disable
1· Enable

Go Active on Poll
- Bit synchronous loop mode

0· Disable
1 Enable

Request to Send
0 $\overline{\text{RTSM}}$ line at low level
1 RTSM line at high level

■ Built-in Registers (cont.)

MSCI

Register	Address	Remarks
MSCI synchronous/address register 0 (MSA0)	002FH	

	7	6	5	4	3	2	1	0
Async	–	–	–	–	–	–	–	–
Byte Sync	SA07	SA06	SA05	SA04	SA03	SA02	SA01	SA00
Bit Sync								
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	1	1	1	1	1	1	1	1

SYN Pattern for Reception/Address Field Check

- Byte synchronous mode

Mono-sync	SYN pattern for reception
Bi-sync	SYN pattern for transmission and reception (bits 7–0)
External-sync	Unused

- Bit synchronous mode

HDLC mode	Address field not checked	Unused
	Single address 1	Bits 7–0 of the secondary station address
	Single address 2	Unused
	Dual address	Bits 7–0 of the secondary station address
Loop mode	Address field not checked	Unused
	Single address 1	Bits 7–0 of the secondary station address
	4-bit address	Bits 7–4 of the secondary station address
	Dual address	Bits 7–0 of the secondary station address

Register	Address	Remarks
MSCI synchronous/address register 1 (MSA1)	0030H	

	7	6	5	4	3	2	1	0
Async	–	–	–	–	–	–	–	–
Byte Sync	SA17	SA16	SA15	SA14	SA13	SA12	SA11	SA10
Bit Sync								
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	1	1	1	1	1	1	1	1

SYN Pattern for Transmission/Address Field Check

- Byte synchronous mode

Mono-sync	SYN pattern for transmission
Bi-sync	SYN pattern for transmission and reception (bits 15–8)
External-sync	SYN pattern for transmission

- Bit synchronous mode

HDLC mode	Address field not checked	Unused
	Single address 1	Unused
	Single address 2	Bits 15–8 of the secondary station address
	Dual address	Bits 15–8 of the secondary station address
Loop mode	Address field not checked	Unused
	Single address 1	Unused
	4-bit address	Unused
	Dual address	Bits 15–8 of the secondary station address

Register	Address	Remarks
MSCI idle pattern register (MIDL)	0031H	

	7	6	5	4	3	2	1	0
Async	–	–	–	–	–	–	–	–
Byte Sync	IDL7	IDL6	IDL5	IDL4	IDL3	IDL2	IDL1	IDL0
Bit Sync								
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	1	1	1	1	1	1	1	1

Idle Pattern

■ Built-in Registers (cont.)

MSCI

Register **Address** **Remarks**

MSCI time constant register (MTMC) 0032H

	7	6	5	4	3	2	1	0
Async Byte Sync Bit Sync	TMC7	TMC6	TMC5	TMC4	TMC3	TMC2	TMC1	TMC0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	1

Value loaded to the reload timer (1 – 256)

MSCI RX clock source register (MRXS) 0033H

	7	6	5	4	3	2	1	0
Async Byte Sync Bit Sync	–	RXCS2	RXCS1	RXCS0	RXBR3	RXBR2	RXBR1	RXBR0
Read/Write	–	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Receive Clock Source
000· RXCM line inpu
010: RXCM line input (noise suppression)
100 Internal baud rate generator (BRG) output
110: ADPLL output
(BRG output for ADPLL operating clock)
111· ADPLL output
(RXCM line input for ADPLL operating clock)
Others: Reserved

Receive Baud Rate
• Clock division ratio
0000. 1/1
0001 1/2
0010. 1/4
0011· 1/8
0100. 1/16
0101 1/32
0110· 1/64
0111 1/128
1000: 1/256
1001 1/512
Others: Reserved

MSCI TX clock source register (MTXS) 0034H

	7	6	5	4	3	2	1	0
Async Byte Sync Bit Sync	–	TXCS2	TXCS1	TXCS0	TXBR3	TXBR2	TXBR1	TXBR0
Read/Write	–	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Transmit Clock Source
000: TXCM line input
100. Internal baud rate generator (BRG) output
110. Receiver clock
Others: Reserved

Transmit Baud Rate
• Clock division ratio
0000: 1/1
0001 1/2
0010 1/4
0011: 1/8
0100. 1/16
0101: 1/32
0110 1/64
0111: 1/128
1000: 1/256
1001· 1/512
Others: Reserved

Register	Address
Unused	0035H
Unused	0036H
Unused	0037H

■ Built-in Registers (cont.)

ASCI/CSIO

Register	Address	Remarks
ASCI TX/RX buffer register (TRB)	0038H	

	7	6	5	4	3	2	1	0
Async Clocked Serial	TRB7	TRB6	TRB5	TRB4	TRB3	TRB2	TRB1	TRB0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	X	X	X	X	X	X	X	X

Transmit/Receive buffer value

Register	Address	Remarks
ASCI status register 0 (ST0)	0039H	

	7	6	5	4	3	2	1	0
Async Clocked Serial	TXINT	RXINT	–	–	–	–	TXRDY	RXRDY
Read/Write	R	R	–	–	–	–	R	R
Initial Value	0	0	0	0	0	0	0	0

TXINT Interrupt
0· TX interrupt not requested
1: TX interrupt requested

RXINT Interrupt
0: RX interrupt not requested
1 RX interrupt requested

RX Ready
0· Receive data does not exist
1· Receive data exists

TX Ready
0: Transmit buffer not empty
1. Transmit buffer empty

Register	Address	Remarks
ASCI status register 1 (ST1)	003AH	

	7	6	5	4	3	2	1	0
Async Clocked Serial	–	IDL	–	–	CCTS	CDCD	BRKD	BRKE
Read/Write	–	R	–	–	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Transmitter Idle State
0 TX not idle
1: TX idle

$\overline{\text{CTSA}}$ Line Level Change
0. Line level not changed
1· Line level changed

Break End Detection
• Asynchronous mode
0: Break end not detected
1. Break end detected

$\overline{\text{DCDA}}$ Line Level Change
0: Line level not changed
1. Line level changed

Break Start Detection
• Asynchronous mode
0. Break start not detected
1 Break start detected

Register	Address	Remarks
ASCI status register 2 (ST2)	003BH	

	7	6	5	4	3	2	1	0
Async Clocked Serial	–	PMP	PE	FRME	OVRN	–	–	–
Read/Write	–	R/W	R/W	R/W	R/W	–	–	–
Initial Value	0	0	0	0	0	0	0	0

Parity/MP Bit
Parity/MP bit value
0· Parity/MP bit value 0
1. Parity/MP bit value 1

Overrun Error
0 No overrun error detected
1 Overrun error detected

Framing Error
0. No framing error detected
1 Framing error detected

Parity Error
0 No parity error detected
1· Parity error detected

Built-in Registers (cont.)

ASCI/CSIO

Register	Address	Remarks
ASCI status register 3 (ST3)	003CH	

	7	6	5	4	3	2	1	0
Async Clocked Serial	–	–	–	–	$\overline{\text{CTS}}$	$\overline{\text{DCD}}$	TXENBL	RXENBL
Read/Write	–	–	–	–	R	R	R	R
Initial Value	0	0	0	0	X	X	0	0

$\overline{\text{CTSA}}$ Input Line Level
0. $\overline{\text{CTSA}}$ line low level
1 $\overline{\text{CTSA}}$ line high level

TX Enable
0 Disable
1 Enable

$\overline{\text{DCDA}}$ Input Line Level
0: $\overline{\text{DCDA}}$ line low level
1 DCDA line high level

RX Enable
0 Disable
1 Enable

Register	Address
Unused	003DH
ASCI interrupt enable register 0 (IE0)	003EH

	7	6	5	4	3	2	1	0
Async Clocked Serial	TXINTE	RXINTE	–	–	–	–	TXRDYE	RXRDYE
Read/Write	R/W	R/W	–	–	–	–	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

TXINT Interrupt Enable
0 Disable
1 Enable

RXINT Interrupt Enable
0 Disable
1 Enable

TXRDY Interrupt Enable
0 Disable
1 Enable

RXRDY Interrupt Enable
0 Disable
1 Enable

Register	Address
ASCI interrupt enable register 1 (IE1)	003FH

	7	6	5	4	3	2	1	0
Async Clocked Serial	–	IDLE	–	–	CCTSE	CDCDE	BRKDE	BRKEE
Read/Write	–	R/W	–	–	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

IDL Interrupt Enable
0. Disable
1 Enable

CCTS Interrupt Enable
0 Disable
1. Enable

BRKD Interrupt Enable
- Asynchronous mode
0 Disable
1 Enable

CDCD Interrupt Enable
0 Disable
1: Enable

BRKE Interrupt Enable
- Asynchronous mode
0 Disable
1. Enable

■ Built-in Registers (cont.)

ASCI/CSIO

Register **Address** **Remarks**

ASCI interrupt enable register 2 (IE2) 0040H

	7	6	5	4	3	2	1	0
Async	–	PMPE	PEE	FRMEE	OVRNE	–	–	–
Clocked Serial								
Read/Write	–	R/W	R/W	R/W	R/W	–	–	–
Initial Value	0	0	0	0	0	0	0	0

PMP Interrupt Enable
0: Disable
1· Enable

PE Interrupt Enable
0· Disable
1: Enable

FRME Interrupt Enable
- Asynchronous mode
0. Disable
1: Enable

OVRN Interrupt Enable
0. Disable
1: Enable

Unused 0041H

ASCI command register (CMD) 0042H

	7	6	5	4	3	2	1	0
Async	-	-	CMD5	CMD4	CMD3	CMD2	CMD1	CMD0
Clocked Serial								
Read/Write	–	–	W	W	W	W	W	W
Initial Value	–	–	–	–	–	–	–	–

Command

- Transmit commands
000001. TX reset
000010· TX enable
000011. TX disable
001000· MP bit on
001001: TX buffer clear

- Receive commands
010001: RX reset
010010. RX enable
010011· RX disable
010110: Search MP bit

- Other commands
100001: Channel reset
000000· No operation
Others: Reserved

ASCI mode register 0 (MD0) 0043H

	7	6	5	4	3	2	1	0
Async	PRTCL2	PRTCL1	PRTCL0	AUTO	–	–	STOP1	STOP2
Clocked Serial							–	–
Read/Write	R/W	R/W	R/W	R/W	–	–	R/W	R/W
	0	0	0	0	0	0	0	0

Protocol Mode
000: Asynchronous mode
110: Clocked serial mode
Other values are reserved

Auto-Enable
0: Auto-enable function not used
1 Auto-enable function used

Stop Bit Length
- Asynchronous mode
00: 1 bit
10: 2 bits
- Clocked serial
Reserved

ASCI mode register 1 (MD1) 0044H

	7	6	5	4	3	2	1	0
Async	BRATE1	BRATE0	TXCHR1	TXCHR0	RXCHR1	RXCHR0	PMPM1	PMPM0
Clocked Serial								
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Bit Rate
- Asynchronous mode
00: 1/1 of clock rate
01: 1/16 of clock rate
10: 1/32 of clock rate
11: 1/64 of clock rate
- Clocked serial mode
These bits should be set to 00.

Transmit Character Length
00: 8 bits/character
01: 7 bits/character

Receive Character Length
00: 8 bits/character
01: 7 bits/character

Parity/Multiprocessor Mode
00: No parity/MP bit
01: MP bit appended (value specified by command)
10: Even parity appended and checked
11: Odd parity appended and checked

■ Built-in Registers (cont.)

ASCI/CSIO

Register	Address	Remarks
ASCI mode register 2 (MD2)	0045H	

	7	6	5	4	3	2	1	0
Async	–	–	–	–	–	–	CNCT1	CNCT0
Clocked Serial								
Read/Write	–	–	–	–	–	–	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Channel Connection
00· Full duplex
01: Auto-echo
10: Reserved
11: Local loop-back

Register	Address	Remarks
ASCI control register (CTL)	0046H	

	7	6	5	4	3	2	1	0
Async	–	–	–	–	BRK	–	–	$\overline{\text{RTS}}$
Clocked Serial								
Read/Write	–	–	–	–	R/W	–	–	R/W
Initial Value	0	0	0	0	0	0	0	1

Break Send
- Asynchronous mode
0· Off (Normal operation)
1· On (Break send)
- Clocked serial mode
Set this bit to 0

Request to Send
0: $\overline{\text{RTSA}}$ low level output
1. $\overline{\text{RTSA}}$ high level output

Register	Address	Remarks
Unused	0047H	
Unused	0048H	
Unused	0049H	

■ Built-in Registers (cont.)

ASCI/CSIO

Register **Address** **Remarks**

ASCI time constant register (TMC) 004AH

	7	6	5	4	3	2	1	0
Async Clocked Serial	TMC7	TMC6	TMC5	TMC4	TMC3	TMC2	TMC1	TMC0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	1

Reload Timer Value (1 – 256)

ASCI RX clock source register (RXS) 004BH

	7	6	5	4	3	2	1	0
Async Clocked Serial	–	RXCS2	RXCS1	RXCS0	–	–	–	–
Read/Write	–	R/W	R/W	R/W	–	–	–	–
Initial Value	0	0	0	0	0	0	0	0

Receive Clock Source
- Asynchronous mode

000 RXCA line input
100 Internal baud rate generator (BRG) output
Others Reserved

RX Master/Slave Mode Select
- Clocked serial mode

000 Slave mode
100 Master mode
Others· Reserved

ASCI TX clock source register (TXS) 004CH

	7	6	5	4	3	2	1	0
Async Clocked Serial	–	TXCS2	TXCS1	TXCS0	TXBR3	TXBR2	TXBR1	TXBR0
Read/Write	–	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Transmit Clock Source
- Asynchronous mode

000 TXCA line input
100: Internal baud rate generator (BRG) output
Others Reserved

TX Master/Slave Mode Select
- Clocked serial mode

000 Slave mode
100 Master mode
Others. Reserved

Baud Rate
- Clock division ratio

0000	1/1	0101	1/32
0001	1/2	0110	1/64
0010	1/4	0111	1/128
0011	1/8	1000	1/256
0100	1/16	1001	1/512

Others Reserved

Unused	004DH
Unused	004EH
Unused	004FH

■ Built-in Registers (cont.)

Timer (channel 0)

Register	Address	Remarks
Timer up-counter channel 0 (TCNT channel 0)	0050H	

	7	6	5	4	3	2	1	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Timer constant register channel 0 (TCONR channel 0)	0051H	

	7	6	5	4	3	2	1	0
Read/Write	W	W	W	W	W	W	W	W
Initial Value	1	1	1	1	1	1	1	1

Timer control/status register channel 0 (TCSR channel 0)	0052H	

	7	6	5	4	3	2	1	0
Bit Name	CMF	ECMI	–	TME	TOS1	TOS0	CKS1	CKS0
Read/Write	R	R/W	–	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Compare Match Flag
0. TCNT and TCONR are not equal
1 TCNT and TCONR are equal

CMF Interrupt Enable
0· Disable
1: Enable

Timer Enable
0: Count stop
1: Count start

Timer Output Select
00· Output fixed to 0
01. Toggled output
10· Output 0
11· Output 1

Input Clock Select
00 BC
01: BC/8
10: BC/128
11: External event count signal

Timer expand prescale register channel 0 (TEPR channel 0)	0053H	

	7	6	5	4	3	2	1	0
Bit Name	EEP	–	–	–	–	ECKS2	ECKS1	ECKS0
Read/Write	R/W	–	–	–	–	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Enable Expand Prescaler
0. Clock is selected by the CKS1-0 bits in TCSR
1: Clock is selected by the ECKS2-0 bits in TEPR

Expand Clock Input Select
000. BC
001· BC/2
010· BC/4
011. BC/8
100: BC/16
101 BC/32
110: BC/64
111· BC/128

■ Built-in Registers (cont.)

Timer (channel 1)

Register	Address	Remarks
Timer up-counter channel 1 (TCNT channel 1)	0054H	

	7	6	5	4	3	2	1	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Register	Address	Remarks
Timer constant register channel 1 (TCONR channel 1)	0055H	

	7	6	5	4	3	2	1	0
Read/Write	W	W	W	W	W	W	W	W
Initial Value	1	1	1	1	1	1	1	1

Register	Address	Remarks
Timer control/status register channel 1 (TCSR channel 1)	0056H	

	7	6	5	4	3	2	1	0
Bit Name	CMF	ECMI	–	TME	TOS1	TOS0	CKS1	CKS0
Read/Write	R	R/W	–	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Compare Match Flag
0 TCNT and TCONR are not equal
1 TCNT and TCONR are equal

CMF Interrupt Enable
0 Disable
1 Enable

Timer Enable
0 Count stop
1 Count start

Timer Output Select
00 Output fixed to 0
01 Toggled output
10 Output 0
11 Output 1

Input Clock Select
00 BC
01 BC/8
10. BC/128
11· External event count signal

Register	Address	Remarks
Timer expand prescale register channel 1 (TEPR channel 1)	0057H	

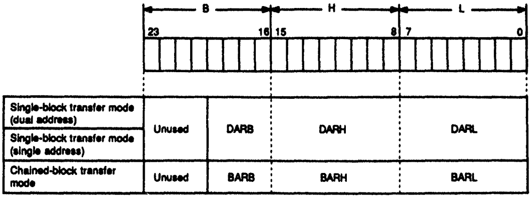
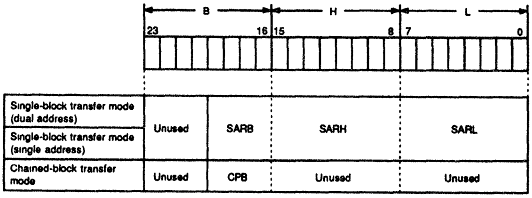
	7	6	5	4	3	2	1	0
Bit Name	EEP	–	–	–	–	ECKS2	ECKS1	ECKS0
Read/Write	R/W	–	–	–	–	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Enable Expand Prescaler
0 Clock is selected by the CKS1-0 bits in TCSR
1 Clock is selected by the ECKS2-0 bits in TEPR

Expand Clock Input Select
000 BC
001 BC/2
010 BC/4
011 BC/8
100 BC/16
101 BC/32
110 BC/64
111 BC/128

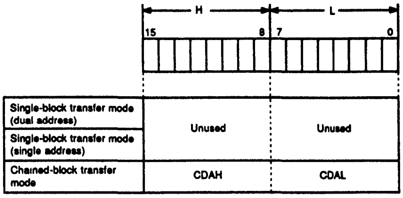
■ Built-in Registers (cont.)

DMAC (channel 0)

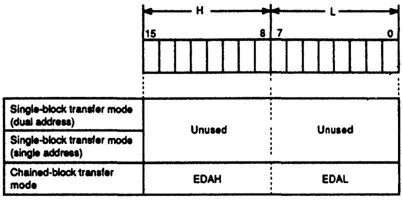
Register	Address	Remarks
Destination address register L channel 0/buffer address register L channel 0 (DARL channel 0/BARL channel 0)	0058H	 B H L 23 16 15 8 7 0 Single-block transfer mode (dual address) Single-block transfer mode (single address) Chained-block transfer mode Unused DARB DARH DARL Unused BARB BARH BARL
Destination address register H channel 0/buffer address register H channel 0 (DARH channel 0/BARH channel 0)	0059H	
Destination address register B channel 0/buffer address register B channel 0 (DARB channel 0/BARB channel 0)	005AH	
Source address register L channel 0 (SARL channel 0)	005BH	 B H L 23 16 15 8 7 0 Single-block transfer mode (dual address) Single-block transfer mode (single address) Chained-block transfer mode Unused SARB SARH SARL Unused CPB Unused Unused
Source address register H channel 0 (SARH channel 0)	005CH	
Source address register B channel 0/chain pointer base channel 0 (SARB channel 0/CPB channel 0)	005DH	

■ Built-in Registers (cont.)

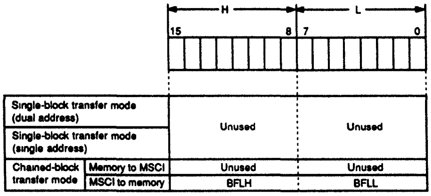
DMAC (channel 0)

Register	Address	Remarks
Current descriptor address register L channel 0 (CDAL channel 0)	005EH	
Current descriptor address register H channel 0 (CDAH channel 0)	005FH	

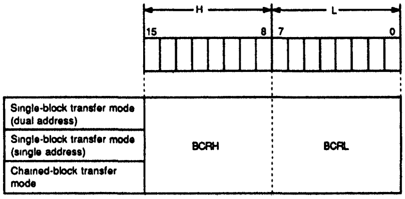
Single-block transfer mode (dual address)	Unused	Unused
Single-block transfer mode (single address)		
Chained-block transfer mode	CDAH	CDAL

Register	Address	Remarks
Error descriptor address register L channel 0 (EDAL channel 0)	0060H	
Error descriptor address register H channel 0 (EDAH channel 0)	0061H	

Single-block transfer mode (dual address)	Unused	Unused
Single-block transfer mode (single address)		
Chained-block transfer mode	EDAH	EDAL

Register	Address	Remarks
Receive buffer length L channel 0 (BFLL channel 0)	0062H	
Receive buffer length H channel 0 (BFLH channel 0)	0063H	

Single-block transfer mode (dual address)		Unused	Unused
Single-block transfer mode (single address)			
Chained-block transfer mode	Memory to MSCI	Unused	Unused
	MSCI to memory	BFLH	BFLL

Register	Address	Remarks
Byte count register L channel 0 (BCRL channel 0)	0064H	
Byte count register H channel 0 (BCRH channel 0)	0065H	
Unused	0066H	
Unused	0067H	

Single-block transfer mode (dual address)	BCRH	BCRL
Single-block transfer mode (single address)		
Chained-block transfer mode		

■ Built-in Registers (cont.)

DMAC (channel 0)

Register **Address** **Remarks**

DMA status register channel 0 (DSR channel 0) 0068H

	7	6	5	4	3	2	1	0
Single-block transfer mode (dual address)	EOT	–	–	–	–	–	DE	$\overline{\text{DWE}}$
Single-block transfer mode (single address)								
Chained-block transfer mode		EOM	BOF	COF				
Read/Write	R/W	R/W	R/W	R/W	–	–	R/W	W
Initial Value	0	0	0	0	0	0	0	1

End of Transfer
0 Transfer not completed
1 Transfer completed

Counter Overflow
- Chained-block transfer
0: Error not detected
1. Error detected

DMA Enable
0· Disable
1: Enable

Buffer Overflow/Underflow
- Chained block transfer
0: Error not detected
1 Error detected

DE Bit Write Enable
0 Enable
1. Disable

End of Frame Transfer
- Chained-block transfer
0· Frame transfer not completed
1. Frame transfer completed

DMA mode register A channel 0 (DMRA channel 0) 0069H

	7	6	5	4	3	2	1	0
Single-block transfer mode (dual address)	RSEL1	RSEL0	AMOD	TMOD	–	–	CNTE	DMS
Single-block transfer mode (single address)					RT			
Chained-block transfer mode						NF		
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	1	0	0	0	0	0

$\overline{\text{DREQ}}$ Input Mode
- Single-block
(dual address)
0· $\overline{\text{DREQ}}$ level sensitive
1: $\overline{\text{DREQ}}$ edge sensitive
- Single-block
(single address)/
Chained-block modes
Set this bit to 0

DMA Transfer Request Source
00 External line
01: Reserved
10: MSCI
11· Reserved

DMA Transfer Direction
- Single-block
(single address)/
Chained-block modes
0: MSCI to memory
1 Memory to MSCI

DMA Transfer Mode
00· Single-block transfer
(single address)
01. Chained-block transfer
10: Single-block transfer
(dual address)
11: Reserved

Frame-End Interrupt-Counter Enable/ Disable
- Single-block transfer
(single/dual address)
Set this bit to 0.
- Chained-block transffer
0. Frame-end interrupt-counter disabled
1 Frame-end interrupt-counter enabled

Number of DMA Frames
- Chained-block transfers
0 Single frame
1 Multi-frame

DMA mode register B channel 0 (DMRB channel 0) 006AH

	7	6	5	4	3	2	1	0
Single-block transfer mode (dual address)	–	–	–	DM1	DM0	SM1	SM0	MMOD
Single-block transfer mode (single address)				–	–	–	–	–
Chained-block transfer mode								
Read/Write	–	–	–	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	1	1	0	0	0

Destination (address increment or decrement)
- Single-block (dual address)
00: Memory (+1)
01: Memory (–1)
10: Memory (fixed)
11: I/O (fixed)

Source (address increment or decrement)
- Single-block (dual address)
00: Memory (+1)
01: Memory (–1)
10: Memory (fixed)
11. I/O (fixed)

Mode for Memory-to-Memory Transfers
- Single-block (dual address)
0: Cycle steal mode
1: Burst mode

■ Built-in Registers (cont.)

DMAC (channel 0)

Register | **Address** | **Remarks**

Frame-end interrupt-counter channel 0 (FCT channel 0) — 006BH

	7	6	5	4	3	2	1	0
Single-block transfer mode (dual address)	–	–	–	–	–	–	–	–
Single-block transfer mode (single address)								
Chained-block transfer mode					FCT3	FCT2	FCT1	FCT0
Read/Write	–	–	–	–	R	R	R	R
Initial Value	0	0	0	0	0	0	0	0

Frame-End Interrupt-Counter Value

DMA interrupt enable register channel 0 (DIR channel 0) — 006CH

	7	6	5	4	3	2	1	0
Single-block transfer mode (dual address)	EOTE	–	–	–	–	–	–	–
Single-block transfer mode (single address)								
Chained-block transfer mode		EOME	BOFE	COFE				
Read/Write	R/W	R/W	R/W	R/W	–	–	–	–
Initial Value	0	0	0	0	0	0	0	0

Transfer End Interrupt Enable
0 Disable
1 Enable

Frame Transfer End Interrupt Enable
- Chained-block transfer mode
0 Disable
1 Enable

Buffer Overflow/Underflow Interrupt Enable
- Chained-block transfer mode
0 Disable
1 Enable

Counter Overflow Interrupt Enable
- Chained-block transfer mode
0· Disable
1 Enable

DMA command register channel 0 (DCR channel 0) — 006DH

	7	6	5	4	3	2	1	0
Single-block transfer mode(dual address)	–	–	–	–	–	–	CMD1	CMD0
Single-block transfer mode(single address)								
Chained-block transfer mode								
Read/Write	–	–	–	–	–	–	W	W
Initial Value	–	–	–	–	–	–	–	–

Command Specification
01 Software abort
10: Frame-end interrupt-counter-clear
Others: Reserved

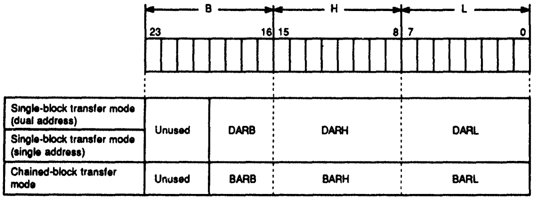
Command Name	Function
Software abort (01H)	Initializes the corresponding DMAC channel (see figure 6-2). All DMAC registers maintain their previous value.
Frame-end interrupt - counter-clear (02H)	Clears the frame-end interrupt-counter (FCT) of the corresponding DMAC channel to 0H and the EOM bit in the DSR to 0.

Unused — 006EH

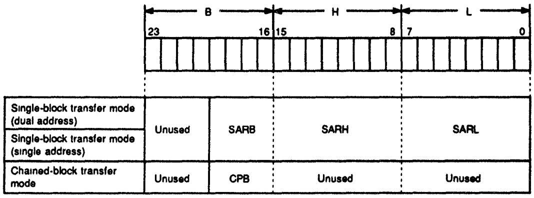
Unused — 006FH

■ Built-in Registers (cont.)

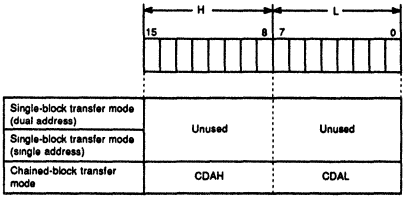
DMAC (channel 1)

Register	Address	Remarks
Destination address register L channel 1/buffer address register L channel 1 (DARL channel 1/BARL channel 1)	0070H	
Destination address register H channel 1/buffer address register H channel 1 (DARH channel 1/BARH channel 1)	0071H	
Destination address register B channel 1/buffer address register B channel 1 (DARB channel 1/BARB channel 1)	0072H	

Single-block transfer mode (dual address) Single-block transfer mode (single address)	Unused	DARB	DARH	DARL
Chained-block transfer mode	Unused	BARB	BARH	BARL

Source address register L channel 1 (SARL channel 1)	0073H	
Source address register H channel 1 (SARH channel 1)	0074H	
Source address register B channel 1/chain pointer base channel 1 (SARB channel 1/ CPB channel 1)	0075H	

Single-block transfer mode (dual address) Single-block transfer mode (single address)	Unused	SARB	SARH	SARL
Chained-block transfer mode	Unused	CPB	Unused	Unused

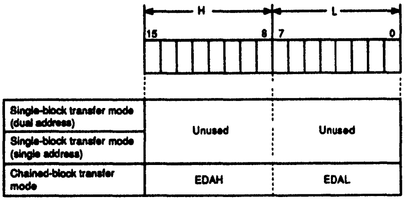
Current descriptor addr^ss register L channe! 1 (CDAL channel 1)	0076H	
Current descriptor address register H channel 1 (CDAH channel 1)	0077H	

Single-block transfer mode (dual address) Single-block transfer mode (single address)	Unused	Unused
Chained-block transfer mode	CDAH	CDAL

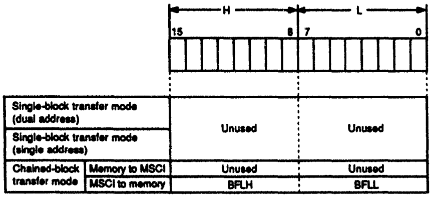
■ Built-in Registers (cont.)

DMAC (channel 1)

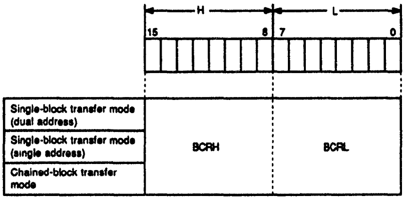
Register	Address	Remarks
Error descriptor address register L channel 1 (EDAL channel 1)	0078H	
Error descriptor address register H channel 1 (EDAH channel 1)	0079H	


H
L
15
8
7
0
Single-block transfer mode (dual address)
Single-block transfer mode (single address)
Chained-block transfer mode
Unused
Unused
EDAH
EDAL

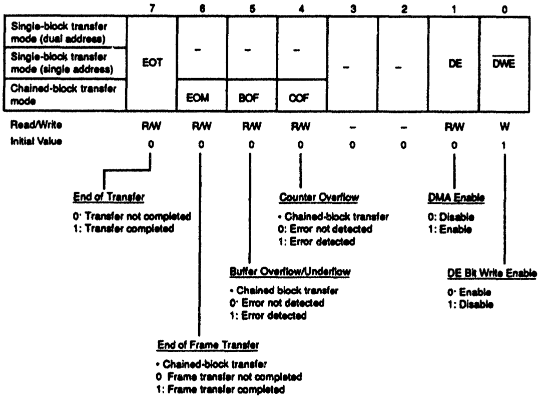
Register	Address	Remarks
Receive buffer length L channel 1 (BFLL channel 1)	007AH	
Receive buffer length H channel 1 (BFLH channel 1)	007BH	


H
L
15
8
7
0
Single-block transfer mode (dual address)
Single-block transfer mode (single address)
Chained-block transfer mode
Memory to MSCI
MSCI to memory
Unused
Unused
Unused
Unused
BFLH
BFLL

Register	Address	Remarks
Byte count register L channel 1 (BCRL channel 1)	007CH	
Byte count register H channel 1 (BCRH channel 1)	007DH	
Unused	007EH	
Unused	007FH	


H
L
15
8
7
0
Single-block transfer mode (dual address)
Single-block transfer mode (single address)
Chained-block transfer mode
BCRH
BCRL

Register	Address	Remarks
DMA status register channel 1 (DSR channel 1)	0080H	



■ Built-in Registers (cont.)

DMAC (channel 1)

Register | **Address** | **Remarks**

DMA mode register A channel 1 (DMRA channel 1) 0081H

	7	6	5	4	3	2	1	0
Single-block transfer mode (dual address)	RSEL1	RSEL0	AMOD	TMOD	–	–	CNTE	DMS
Single-block transfer mode (single address)					RT			
Chained-block transfer mode						NF		
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	1	0	0	0	0	0

DMA Transfer Request Source

00: External line
01 Reserved
10 MSCI
11. Reserved

DMA Transfer Direction

- Single-block (single address)/ Chained-block modes
0 MSCI to memory
1: Memory to MSCI

DREQ Input Mode

- Single-block (dual address)
0. DREQ level sensitive
1· DREQ edge sensitive
- Single-block (single address)/ Chained-block modes
Set this bit to 0

DMA Transfer Mode

00· Single-block transfer (single address)
01 Chained-block transfer
10. Single-block transfer (dual address)
11 Reserved

Frame-End Interrupt-Counter Enable/ Disable

- Single-block transfer (single/dual address)
Set this bit to 0
- Chained-block transffer
0: Frame-end interrupt-counter disabled
1 Frame-end interrupt-counter enabled

Number of DMA Frames

- Chained-block transfers
0 Single frame
1 Multi-frame

DMA mode register B channel 1 (DMRB channel 1) 0082H

	7	6	5	4	3	2	1	0
Single-block transfer mode (dual address)	–	–	–	DM1	DM0	SM1	SM0	MMOD
Single-block transfer mode (single address)				–	–	–	–	–
Chained-block transfer mode								
Read/Write	–	–	–	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	1	1	0	0	0

Destination (address increment or decrement)

- Single-block (dual address)
00 Memory (+1)
01. Memory (–1)
10. Memory (fixed)
11 I/O (fixed)

Source (address increment or decrement)

- Single-block (dual address)
00 Memory (+1)
01: Memory (–1)
10: Memory (fixed)
11 I/O (fixed)

Mode for Memory-to-Memory Transfers

- Single-block (dual address)
0 Cycle steal mode
1: Burst mode

Frame-end interrupt-counter channel 1 (FCT channel 1) 0083H

	7	6	5	4	3	2	1	0
Single-block transfer mode (dual address)	–	–	–	–	–	–	–	–
Single-block transfer mode (single address)								
Chained-block transfer mode					FCT3	FCT2	FCT1	FCT0
Read/Write	–	–	–	–	R	R	R	R
Initial Value	0	0	0	0	0	0	0	0

Frame-End Interrupt-Counter Value

■ Built-in Registers (cont.)

DMAC (channel 1)

Register	Address	Remarks
DMA interrupt enable register channel 1 (DIR channel 1)	0084H	

	7	6	5	4	3	2	1	0
Single-block transfer mode (dual address)	EOTE	–	–	–	–	–	–	–
Single-block transfer mode (single address)								
Chained-block transfer mode		EOME	BOFE	COFE				
Read/Write	R/W	R/W	R/W	R/W	–	–	–	–
Initial Value	0	0	0	0	0	0	0	0

Transfer End Interrupt Enable
0 Disable
1 Enable

Frame Transfer End Interrupt Enable
• Chained-block transfer mode
0 Disable
1 Enable

Buffer Overflow/Underflow Interrupt Enable
• Chained-block transfer mode
0 Disable
1 Enable

Counter Overflow Interrupt Enable
• Chained-block transfer mode
0 Disable
1 Enable

Register	Address	Remarks
DMA command register channel 1 (DCR channel 1)	0085H	

	7	6	5	4	3	2	1	0
Single-block transfer mode(dual address)	–	–	–	–	–	–	CMD1	CMD0
Single-block transfer mode(single address)								
Chained-block transfer mode								
Read/Write	–	–	–	–	–	–	W	W
Initial Value	–	–	–	–	–	–	–	–

Command Specification
01: Software abort
10: Frame-end interrupt-counter-clear
Others: Reserved

Command Name	Function
Software abort (01H)	Initializes the corresponding DMAC channel (see figure 6-2). All DMAC registers maintain their previous value.
Frame-end interrupt - counter-clear (02H)	Clears the frame-end interrupt-counter (FCT) of the corresponding DMAC channel to 0H and the EOM bit in the DSR to 0.

Register	Address	Remarks
Unused	0086H	
Unused	0087H	
Reserved	0088H | 00DFH	

Built-in Registers (cont.)

External I/O (LOW)

Register	Address	Remarks
IOL	00E0H | 00EFH	

External I/O (HIGH)

Register	Address	Remarks
IOH	00F0H | FFFFH	

HD641180X, HD643180X, HD647180X— MCU (Micro Controller Unit)

■ DESCRIPTION

The HD643180X provides instruction compatibility with the HD64180 and incorporates a 16-kbyte Mask ROM, 512-byte RAM, memory management unit (MMU), DMA controller, timer, asynchronous serial communications interface (ASCI), clocked serial I/O ports (CSI/O), analog comparator and parallel I/O pins on a single chip.

The HD647180X incorporates a 16-kbyte PROM instead of mask ROM.

The internal PROM can be programmed and verified under the same specifications as the 27256 type EPROM (V_{PP}12.5V) using a general-purpose PROM writer.

In addition, the HD643180X and HD647180X are functionally identical except for their internal ROMs.

The HD641180X functions in the same way as the HD643180X or HD647180X, except that the HD641180X has no internal ROM.

■ FEATURES

Software

- Instruction set compatible with the HD64180

Hardware

- 16-kbyte ROM (HD643180X and HD647180X) and 512-byte RAM
- Timer
  - —One-channel 16-bit timer with input capture, output compare, and timer overflow functions
  - —Two-channel 16-bit reload timer
- Six-channel analoag comparator
- 54 parallel I/O pins
  - —Includes eight high current pins (I_{OL} = 10mA)
- MMU with 1-Mbyte memory physical address space
- Two-channel DMA controller
- Two-channel ASCI
- One-channel CSI/O
- Four external and eleven internal interrupts
- DRAM refresh controller and low speed memory, I/O interface
- Operating frequency up to 8 MHz (ϕ clock)
- Low power operation
- Four operation modes (HD643180X and HD647180X)
  - —Mode 0: single-chip mode
  - —Mode 1: expanded mode (internal ROM disabled)
  - —Mode 2: expanded mode (internal ROM enabled)
  - —Mode 3: PROM programming mode (HD647180X only)
- Internal ROM data protect function (HD647180X only)
- Packages
  - —80-pin quad flat package
  - —84-pin plastic leaded chip carrier
  - —90-pin dual inline package

■ BLOCK DIAGRAM

The HD647180X combines a high-performance CPU core with many of the systems and I/O resources required by a broad range of applications (figure 2).

The CPU core consists of five functional blocks

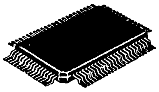
- Clock generator
- Bus state controller
- Interrupt controller
- Memory management unit (MMU)
- Central processing unit (CPU)

The Integrated I/O resources comprise the remaining four functional blocks.

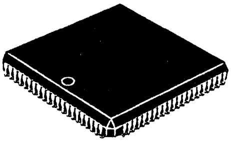
- DMA controller (DMAC· two channels)
- Asynchronous serial communication interface (ASCI: two channels)
- Clocked serial I/O port (CSI/O. one channel)
- Programmable reload timer (PRT. two channels)
- Programmable timer 2 (PT2· one channel)
- Analog comparator (six channels)
- I/O ports

The memory consists of:

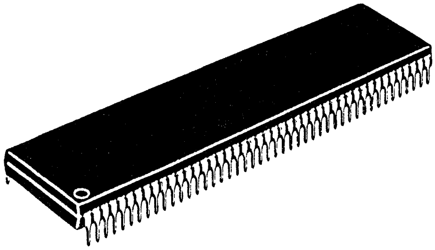
- RAM (512 bytes)
- PROM (16 kbyte). HD647180X
- Mask ROM (16 kbyte): HD643180X



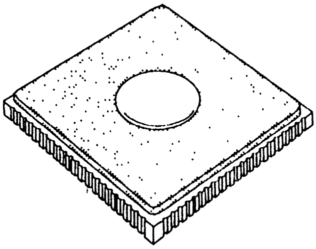
(FP-80B)



(CP-84)



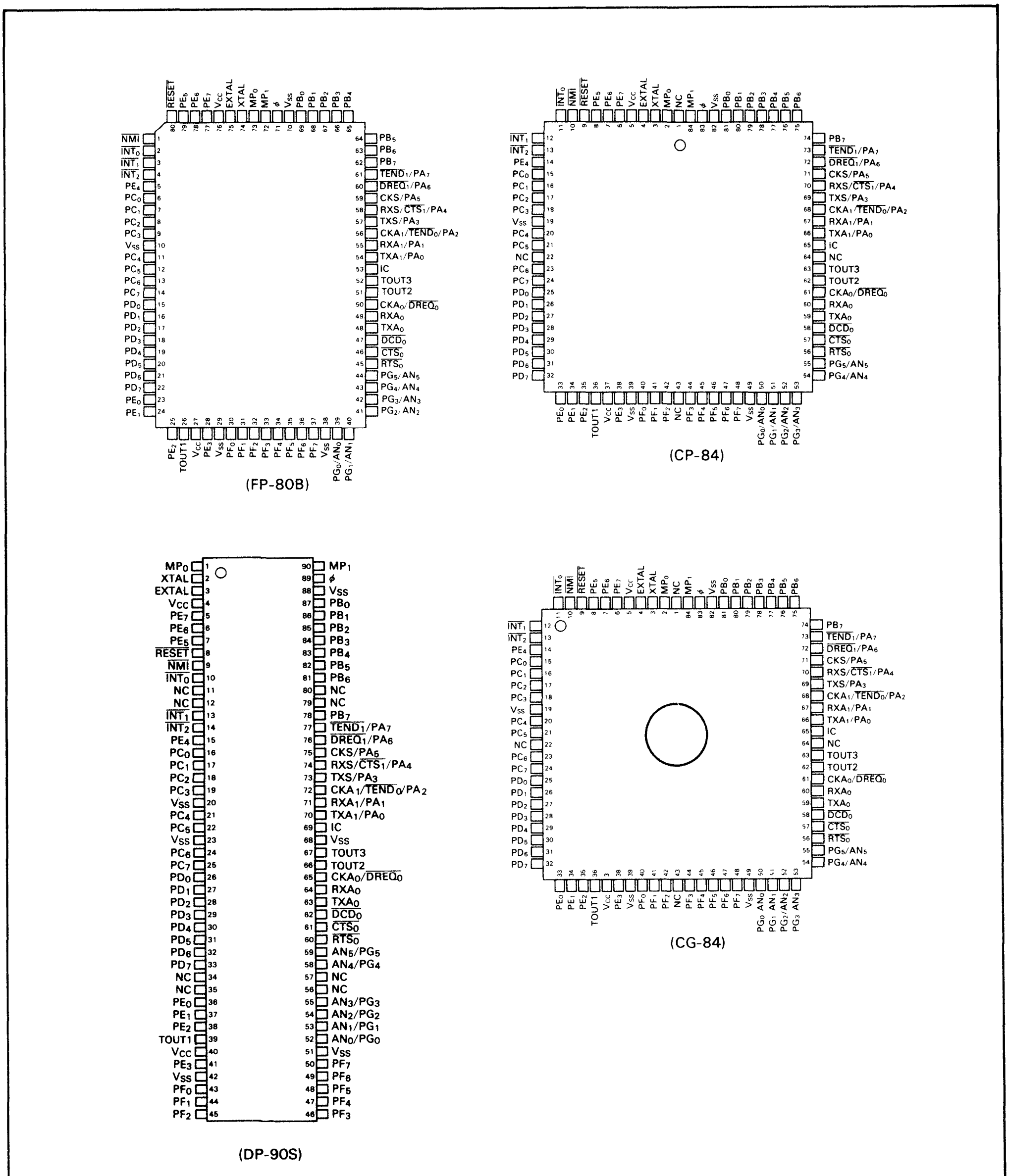
(DP-90S)



(CG-84)

■ Pin Assignment

Figure 1 shows a top view of the HD641180X, HD643180X and HD647180X packages. Table 1 shows the pin functions in the four modes.



Note) NC: Not connected. Please leave the NC pins open.
Note) CG-84: HD647180X only

Figure 1. Pin Assignment

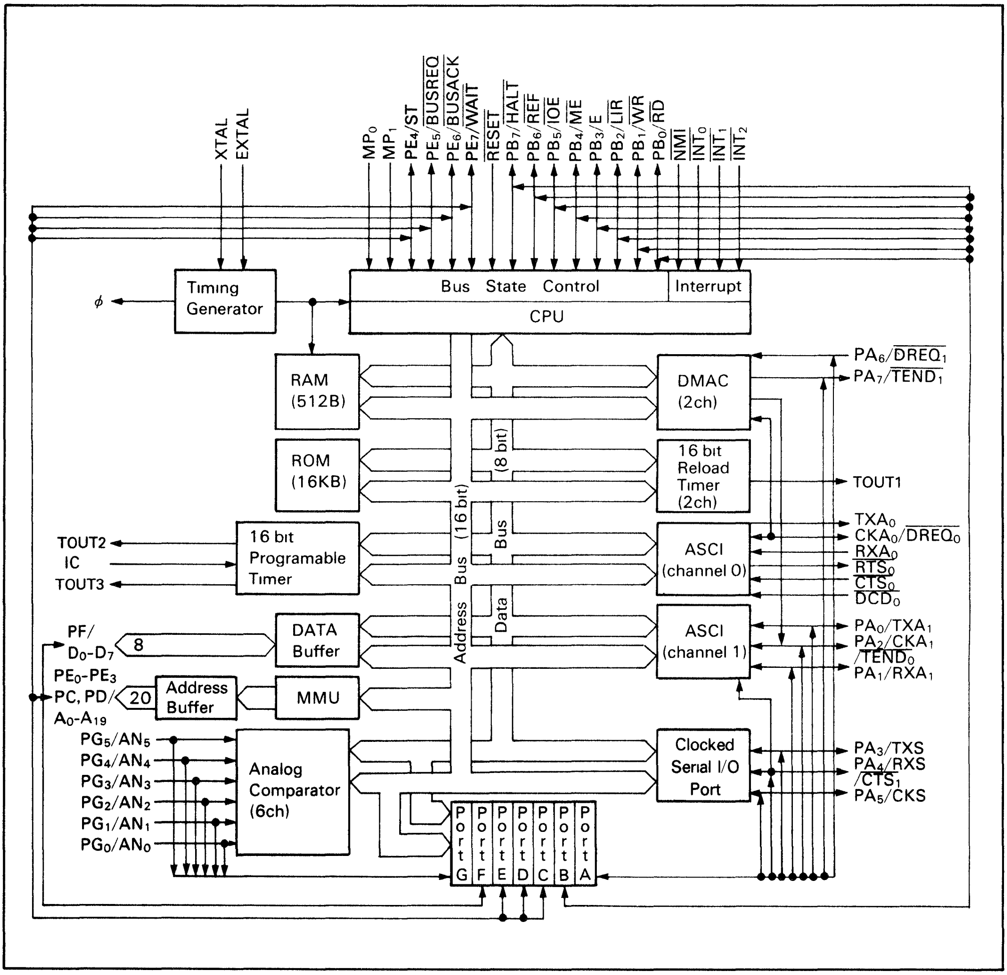


Figure 2. Block Diagram (HD643180X, HD647180X)

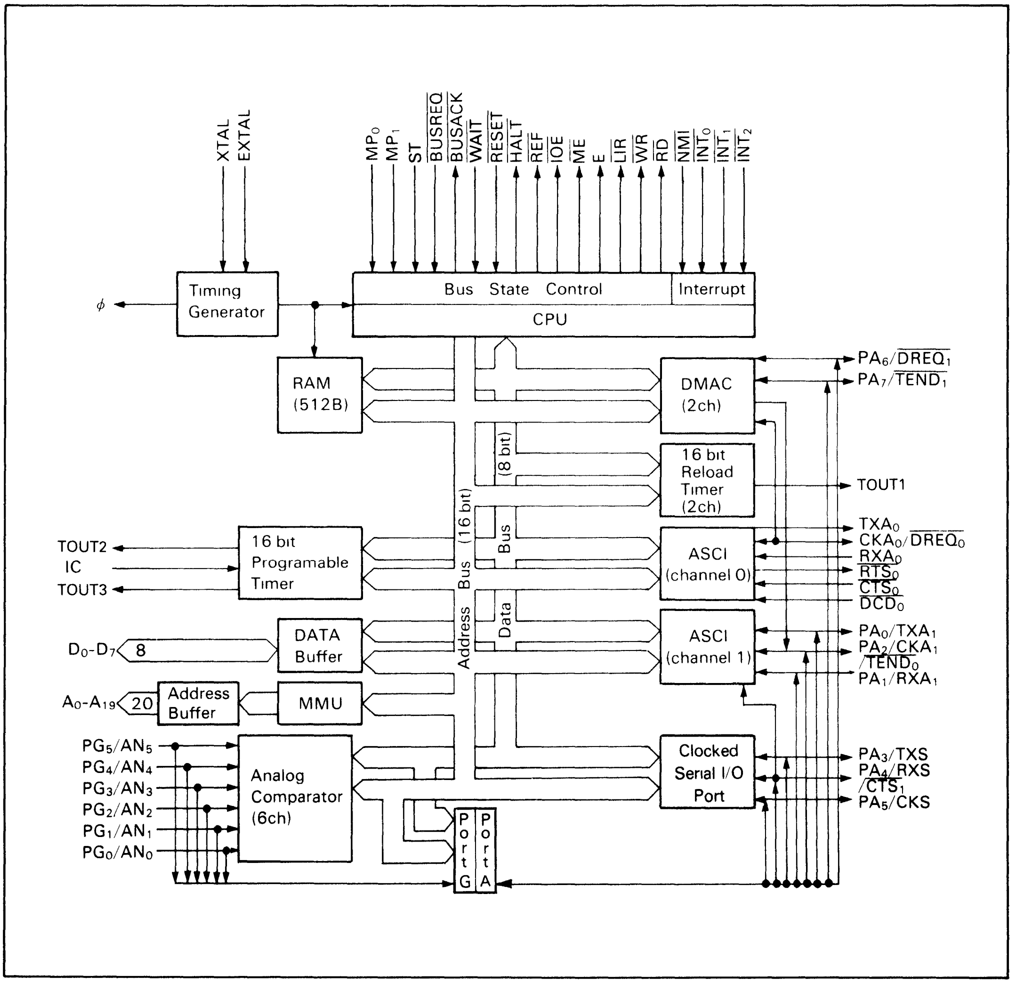


Figure 3. Block Diagram (HD641180X)

Table 1 Pin Function (HD643180X, HD647180X)

Pin No.			Operating Mode 0	Operating Mode 1	Operating Mode 2	Operating Mode 3 (HD647180X only)
FP-80B	CG-84 CP-84	DP-90S				
1	10	9	$\overline{NMI}$	←	←	A_9
2	11	10	$\overline{INT_0}$	←	←	–
3	12	13	$\overline{INT_1}$	←	←	–
4	13	14	$\overline{INT_2}$	←	←	–
5	14	15	PE_4	ST	←	–
6	15	16	PC_0	A_0	←	←
7	16	17	PC_1	A_1	←	←
8	17	18	PC_2	A_2	←	←
9	18	19	PC_3	A_3	←	←
10	19	20	Vss	←	←	←
11	20	21	PC_4	A_4	←	←
12	21	22	PC_5	A_5	←	←
13	23	24	PC_6	A_6	←	←
14	24	25	PC_7	A_7	←	←
15	25	26	PD_0	A_8	A_8/PD_0	A_8
16	26	27	PD_1	A_9	A_9/PD_1	–
17	27	28	PD_2	A_{10}	A_{10}/PD_2	A_{10}
18	28	29	PD_3	A_{11}	A_{11}/PD_3	A_{11}
19	29	30	PD_4	A_{12}	A_{12}/PD_4	A_{12}
20	30	31	PD_5	A_{13}	A_{13}/PD_5	A_{13}
21	31	32	PD_6	A_{14}	A_{14}/PD_6	A_{14}
22	32	33	PD_7	A_{15}	A_{15}/PD_7	$\overline{OE}$
23	33	36	PE_0	A_{16}	A_{16}/PE_0	$\overline{CE}$
24	34	37	PE_1	A_{17}	A_{17}/PE_1	–
25	35	38	PE_2	A_{18}	A_{18}/PE_2	–
26	36	39	TOUT1	←	←	–
27	37	40	Vcc	←	←	←
28	38	41	PE_3	A_{19}	A_{19}/PE_3	–
29	39	42	Vss	←	←	←
30	40	43	PF_0	D_0	←	O_0
31	41	44	PF_1	D_1	←	O_1
32	42	45	PF_2	D_2	←	O_2
33	44	46	PF_3	D_3	←	O_3
34	45	47	PF_4	D_4	←	O_4
35	46	48	PF_5	D_5	←	O_5
36	47	49	PF_6	D_6	←	O_6
37	48	50	PF_7	D_7	←	O_7
38	49	51	Vss	←	←	←
39	50	52	PG_0/AN_0	←	←	–
40	51	53	PG_1/AN_1	←	←	–
41	52	54	PG_2/AN_2	←	←	–

Notes: ← Same as previous column
– No function

For the HD641180X pin function, please refer to table heading Operation Mode 1.

Table 1 Pin Function (HD643180X, HD647180X) (cont.)

Pin No.			Operating Mode 0	Operating Mode 1	Operating Mode 2	Operating Mode 3 (HD647180X only)
FP-80B	CG-84 CP-84	DP-90S				
42	53	55	PG_3/AN_3	←	←	–
43	54	58	PG_4/AN_4	←	←	–
44	55	59	PG_5/AN_5	←	←	–
45	56	60	$\overline{RTS_0}$	←	←	–
46	57	61	$\overline{CTS_0}$	←	←	–
47	58	62	$\overline{DCD_0}$	←	←	–
48	59	63	TXA_0	←	←	–
49	60	64	RXA_0	←	←	–
50	61	65	$CKA_0/\overline{DREQ_0}$	←	←	–
51	62	66	TOUT2	←	←	–
52	63	67	TOUT3	←	←	–
53	65	69	IC	←	←	–
54	66	70	TXA_1/PA_0	←	←	–
55	67	71	RXA_1/PA_1	←	←	–
56	68	72	$CKA_1/\overline{TEND_0}/PA_2$	←	←	–
57	69	73	TXS/PA_3	←	←	–
58	70	74	$RXS/\overline{CTS_1}/PA_4$	←	←	–
59	71	75	CKS/PA_5	←	←	–
60	72	76	$\overline{DREQ_1}/PA_6$	←	←	–
61	73	77	$\overline{TEND_1}/PA_7$	←	←	–
62	74	78	PB_7	$\overline{HALT}$	←	–
63	75	81	PB_6	$\overline{REF}$	←	–
64	76	82	PB_5	$\overline{IOE}$	←	–
65	77	83	PB_4	$\overline{ME}$	←	–
66	78	84	PB_3	E	←	–
67	79	85	PB_2	$\overline{LIR}$	←	–
68	80	86	PB_1	$\overline{WR}$	←	–
69	81	87	PB_0	$\overline{RD}$	←	–
70	82	88	V_{SS}	←	←	←
71	83	89	ϕ	←	←	–
72	84	90	MP_1	←	←	←
73	2	1	MP_0	←	←	←
74	3	2	XTAL	←	←	←
75	4	3	EXTAL	←	←	←
76	5	4	V_{CC}	←	←	←
77	6	5	PE_7	$\overline{WAIT}$	←	–
78	7	6	PE_6	$\overline{BUSACK}$	←	–
79	8	7	PE_5	$\overline{BUSREQ}$	←	–
80	9	8	$\overline{RESET}$	←	←	V_{PP}
–	–	23	V_{SS}	←	←	←
–	–	68	V_{SS}	←	←	←

■ CPU Architecture

The five CPU core functional blocks are described in this section.

Clock Generator

The clock generator generates the system clock (ϕ) from an external crystal or external clock input. Also, the system clock is programmably prescaled to generate timing for the on-chip I/O and system support devices.

Bus State Controller

The bus state controller performs all status/control bus activity. This includes external bus cycle wait state timing, $\overline{\text{RESET}}$, DRAM refresh, and master DMA bus exchange. Generates 'dual-bus' control signals for compatibility with peripheral devices.

Interrupt Controller

The interrupts controller monitors and prioritizes the four external and eight internal interrupt sources. A variety of interrupt response modes are programmable.

Memory Management Unit (MMU)

Maps the CPU 64-kbyte logical memory address space into a 1-Mbyte physical memory address space. The MMU organization preserves software object code compatibility while providing extended memory access and uses an efficient 'common area—bank area' scheme. I/O accesses (64-kbyte I/O address space) bypass the MMU.

Central Processing Unit (CPU)

The CPU is microcoded to implement an upward-compatible superset of the 8-bit standard software instruction set. Many instructions require fewer clock cycles for execution and seven new instructions are added.

Mode Selection

Mode program pins, MP_0 and MP_1 determine the operation mode of the LSI (table 4).

■ I/O Resources

DMA Controller (DMAC)

The two channel DMAC provides high speed memory to/from memory, memory to/from I/O, and memory to/from memory-mapped I/O transfers. The DMAC features edge or level sense request input, address increment/decrement/no-change and (for memory to/from memory transfers) programmable burst or cycle steal transfer. In addition, the DMAC can directly access the full 1-Mbyte of physical memory address space (the MMU is bypassed during DMA) and transfers (up to 64-kbyte in length) can cross 64-kbyte boundaries.

Asynchronous Serial Communication Interface (ASCI)

The ASCI provides two separate full-duplex UARTs and includes a programmable baud rate generator, modem control signals, and a multiprocessor communication format. The ASCI can use the DMAC for high-speed serial data transfer, reducing CPU overhead.

Clocked Serial I/O Port (CSI/O)

The CSI/O half-duplex clocked serial transmitter and receiver can be used for simple, high-speed connection to another microprocessor or microcomputer.

Programmable Reload Timer (PRT)

The PRT contains two separate channels, each consisting of 16-bit timer data and 16-bit timer reload registers. The time base is the system clock divided by 20 (fixed) and PRT channel 1 has an optional output allowing waveform generation.

Programmable Timer 2 (PT2)

The PT2 16-bit programmable timer can measure an input waveform and generate two independent output waveforms. The pulse widths of both input/output waveforms vary from microseconds to seconds.

Analog Comparator

The HD641180X/HD643180X/HD647180X provides an analog comparator with 6 channels. Each channel can be programmed as a reference voltage (V_{ref}) input pin or a compared voltage (V_{in}) input pin.

Input Output Port (I/O Port)

The HD643180X/HD647180X provides seven I/O ports. (port A—G). Each port consists of a data direction register (DDR) to determine the directions of the individual pins, an output data register (ODR) to hold output data and an input data register (IDR) to latch input date. However, Port G does not have a DDR or ODR since it is an input-only port.

■ Pins Signal Description

XTAL, EXTAL: Crystal (Input)

XTAL and EXTAL are the crystal oscillator connections. An external TTL clock can be input on EXTAL. XTAL should be left open if an external TTL clock is used. Note that XTAL. XTAL is schmitt triggered. See DC characteristics.

ϕ (OUT)

ϕ is the system clock output. Its frequency is equal to one-half of the crystal oscillator's.

$\overline{\text{RESET}}$: CPU Reset (Input)

When $\overline{\text{RESET}}$ is low, it initializes the HD641180X/HD643180X/HD647180X CPU. All output signals are held inactive during reset.

A_0-A_{19}: Address Bus (Output, Three-State)

The address bus enters the high-impedance state during reset and when another device acquires the bus as indicated by $\overline{\text{BUSREQ}}$ and $\overline{\text{BUSACK}}$ low. During reset, the address function is selected.

D_0-D_7: Data Bus (Input/Output, Three-State)

The bidirectional 8-bit data bus enters the high-impedance state during reset and when another device acquires the bus as indicated by $\overline{\text{BUSREQ}}$ and $\overline{\text{BUSACK}}$ low.

$\overline{\text{RD}}$: Read (Output, Three-State)

During a CPU read cycle, $\overline{\text{RD}}$ enables transfer from the external memory or I/O device to the CPU data bus.

$\overline{\text{WR}}$: Write (Output, Three-State)

During a CPU write cycle, $\overline{\text{WR}}$ enables transfer from the CPU data bus to the external memory or I/O device.

$\overline{\text{ME}}$: Memory Enable (Output, Three-State)

$\overline{\text{ME}}$ indicates memory read or write operations. The HD641180X/HD643180X/HD647180X asserts $\overline{\text{ME}}$ low in the following cases.

- When fetching instructions and operands
- When reading or writing memory data
- During DMA memory access cycles
- During dynamic RAM refresh cycles

$\overline{\text{IOE}}$: I/O Enable (Output, Three-State)

$\overline{\text{IOE}}$ indicates I/O read or write operations. The HD641180X/HD643180X/HD647180X asserts $\overline{\text{IOE}}$ low in the following cases:

- When reading or writing I/O data
- During DMA I/O access cycles
- During $\overline{\text{INT}_0}$ acknowledge cycle

$\overline{\text{WAIT}}$: Bus Cycle Wait (Input)

$\overline{\text{WAIT}}$ introduces wait states to extend memory and I/O cycles. If low at the falling edge of T_2, a wait state (Tw) is inserted. Wait states will continue to be inserted until the $\overline{\text{WAIT}}$ input is sampled high at the falling edge of Tw, at which time the bus cycle will proceed to completion.

E: Enable (Output)

E is a synchronous clock for connection to HD63× × series and other 6800/6500 series compatible peripheral LSIs.

$\overline{\text{BUSREQ}}$: Bus Request (Input)

Another device may request use of the bus by asserting $\overline{\text{BUSREQ}}$ low. The CPU will stop executing instructions and place the address bus, data bus, $\overline{\text{RD}}$, $\overline{\text{WR}}$, $\overline{\text{ME}}$, and $\overline{\text{IOE}}$ in the high-impedance state.

$\overline{\text{BUSACK}}$: Bus Acknowledge (Output)

When the CPU completes bus release (in response to $\overline{\text{BUSREQ}}$ low), it will assert $\overline{\text{BUSACK}}$ low. This acknowledges that the bus is free for use by the requesting device.

$\overline{\text{HALT}}$: Halt/Sleep Status (Output)

$\overline{\text{HALT}}$ is asserted low after execution of the HALT or SLP instructions. Used with $\overline{\text{LIR}}$ and ST output pins to encode CPU status (table 2).

$\overline{\text{LIR}}$: Load Instruction Register (Output)

$\overline{\text{LIR}}$ is asserted low when the current cycle is an opcode fetch cycle. Used with $\overline{\text{HALT}}$ and ST output pins to encode CPU status (table 2).

ST: Status (Output)

ST is used with the $\overline{\mathrm{HALT}}$ and $\overline{\mathrm{LIR}}$ output pins to encode CPU status (table 2).

Table 2 Status Summary

ST	$\overline{\mathrm{HALT}}$	$\overline{\mathrm{LIR}}$	Operation
0	1	0	CPU operation (1st opcode fetch)
1	1	0	CPU operation (2nd opcode and 3rd opcode fetch)
1	1	1	CPU operation (MC except for opcode fetch)
0	X	1	DMA operation
0	0	0	Halt mode
1	0	1	Sleep mode (including System stop mode)

Note X: Don't care
MC: Machine cycle

$\overline{\mathrm{REF}}$: Refresh (Output)

When low, $\overline{\mathrm{REF}}$ indicates that the CPU is in a dynamic RAM refresh cycle and the low-order 8 bits (A_0-A_7) of the address bus contain the refresh address.

$\overline{\mathrm{NMI}}$: Non-Maskable Interrupt (Input)

When high to low is detected, it forces the CPU to save certain state information and vector to an interrupt service routine at address 0066H. The saved state information is restored by executing the RETN (return from non-maskable interrupt) instruction.

$\overline{\mathrm{INT}}_0$: Maskable Interrupt Level 0 (Input)

When low, $\overline{\mathrm{INT}}_0$ requests a CPU interrupt (unless masked) and saves certain state information unless masked by software. $\overline{\mathrm{INT}}_0$ requests service using one of three software programmable interrupt modes (table 3).

Table 3 Interrupt Modes

Mode	Operation
0	Instruction fetched and executed from data bus
1	Instruction fetched and executed from address 0038H
2	Vector system: Low-order 8 bits of vector table address fetched from data bus

In all modes, the saved state information is restored by executing the RETI (return from interrupt) instruction.

$\overline{\text{INT}_1}$, $\overline{\text{INT}_2}$: Maskable Interrupt Levels 1, 2 (Input)

When low, $\overline{\text{INT}_1}$ and $\overline{\text{INT}_2}$ request a CPU interrupt (unless masked) and save certain state information unless masked by software. $\overline{\text{INT}_1}$ and $\overline{\text{INT}_2}$ (and internally generated interrupts) request interrupt service using a vector system similar to mode 2 of $\overline{\text{INT}_0}$.

$\overline{\text{DREQ}_0}$ DMA Request—Channel 0 (Input)

$\overline{\text{DREQ}_0}$ low (programmable edge or level sense) requests DMA transfer service from channel 0 of the HD641180X/HD643180X/HD647180X DMAC. $\overline{\text{DREQ}_0}$ is used for channel 0 memory to/from I/O and memory to/from memory-mapped I/O transfers. $\overline{\text{DREQ}_0}$ is not used for memory to/from memory transfers. This pin is multiplexed with CKA_0.

$\overline{\text{TEND}_0}$: Transfer End—Channel 0 (Output)

$\overline{\text{TEND}_0}$ is asserted low synchronous with the last write cycle of channel 0 DMA transfer to indicate DMA completion to an external device. This pin is multiplexed with CKA_1.

$\overline{\text{DREQ}_1}$: DMA Request—Channel 1 (Input)

$\overline{\text{DREQ}_1}$ low (programmable edge or level sense) requests DMA transfer service from channel 1 of the HD641180X/HD643180X/HD647180X DMAC. Channel 1 supports memory to/from I/O transfers.

$\overline{\text{TEND}_1}$: Transfer End— Channel 1 (Output)

$\overline{\text{TEND}_1}$ is asserted low synchronous with the last write cycle of channel 1 DMA transfer to indicate DMA completion to an external device.

TXA_0: Asynchronous Transmit Data—Channel 0 (Output)

TXA_0 is the asynchronous transmit data from channel 0 of the asynchronous serial communication interface (ASCI).

RXA_0: Asynchronous Receive Data—Channel 0 (Input)

RXA_0 is the asynchronous receive data to channel 0 of the ASCI.

CKA_0: Asynchronous Clock—Channel 0 (Input/Output)

CKA_0 is the clock input/output for channel 0 of the ASCI. This pin is multiplexed (software selectable) with $\overline{\text{DREQ}_0}$.

$\overline{\text{RTS}_0}$: Request to Send—Channel 0 (Output)

$\overline{\text{RTS}_0}$ is the programmable modem control output signal for channel 0 of the ASCI.

$\overline{\text{CTS}_0}$: Clear to Send—Channel 0 (Output)

$\overline{\text{CTS}_0}$ is the modem control input signal for channel 0 of the ASCI.

$\overline{\text{DCD}_0}$: Data Carrier Detect—Channel 0 (Output)

$\overline{\text{DCD}_0}$ is the modem control input signal for channel 0 of the ASCI.

TXA_1: Asynchronous Transmit Data—Channel 1 (Output)

TXA_1 is the asynchronous transmit data from channel 1 of the ASCI.

RXA_1: Asynchronous Receive Data—Channel 1 (Input)

RXA_1 is the asynchronous receive data to channel 1 of the ASCI.

CKA_1: Asynchronous Clock—Channel 1 (Input/Output)

CKA_1 is the clock input/output for channel 1 of the ASCI. This pin is multiplexed (software selectable) with $\overline{\text{TEND}_0}$.

$\overline{\text{CTS}_1}$: Clear to Send—Channel 1 (Input)

$\overline{\text{CTS}_1}$ is the modem control input signal for channel 1 of the ASCI. This pin is multiplexed (software selectable) with RXS.

TXS: Clocked Serial Transmit Data (Output)

Clocked serial transmit data from the Clocked Serial I/O Port (CSI/O).

RXS: Clocked Serial Receive Data (Input)

Clocked serial receive data to the CSI/O. This pin is multiplexed (software selectable) with ASCI channel 1 $\overline{CTS_1}$ modem control input.

CKS: Serial Clock (Input/Output)

Input or output clock for the CSI/O.

TOUT1: Timer Output (Output)

Pulse output from Programmable Reload Timer channel 1.

AN_0-AN_5: Comparator (Input)

AN_0-AN_5 input data to the analog comparator. Select two of these pins and apply the reference voltage (V_{ref}) and the voltage to be compared (V_{in}) to them.

PA_0-PA_7, PB_0-PB_7, PC_0-PC_7, PD_0-PD_7, PE_0, PE_7, PF_0-PF_7: Parallel Ports A-F (Input/Output)

Ports A-F are 8-bit I/O ports. Each pin of each port can be individually configured as an input or output depending on the port data direction register. At reset, each port is initialized as an input port.

PG_0-PG_5: Parallel Port G (Input)

Port G is a 6-bit input port.

IC: Input Capture (Input)

IC inputs the input capture signal for timer 2.

TOUT2, TOUT3: Timer Output 2, 3 (Output)

TOUT2 and TOUT3 are timer 2's outputs.

MP_0, MP_1: Mode Program 0, 1 (Input)

The mode program pins, MP_0 and MP_1, determine the operation mode of the MPU as shown in table 4.

Table 4. Operating Mode Selection ın

MP_1	MP_0	ROM	RAM	Operating Mode	Applicable Wide-Range
0	0	I	I	0; Single chip mode	HD643180X HD647180X
0	1	E	I	1; Expanded mode 1	HD643180X HD647180X HD641180X
1	0	I	I	2; Expanded mode 2	HD643180X HD647180X
1	1	I	—	3; PROM programming mode (HD647180X only)	HD647180X

I: Internal E: External
Select mode 1 (MP_1 = 0, MP_2 = 1) for the HD641180X.

Vcc, Vss: Power

VCC is power supply. VSS is the ground.

■ Multiplexed Pins

PA_0/TXA_1, PA_1/RXA_1, PA_3/TXS, PA_5/CKS, $\text{PA}_6/\overline{\text{DREQ}_1}$, $\text{PA}_7/\overline{\text{TEND}_1}$

At reset, PA_0/TXA_1, PA_1/RXA_1, PA_3/TXS, PA_5/CKS, $\text{PA}_6/\overline{\text{DREQ}_1}$, and $\text{PA}_7/\overline{\text{TEND}_1}$ are configured as port A input. They can be used as TXA_1, RXA_1, TXS, CKS, $\overline{\text{DREQ}_1}$, and $\overline{\text{TEND}_1}$ by setting the corresponding bit in the port A disable register to 1.

$\text{PA}_2/\text{CKA}_1/\overline{\text{TEND}_0}$

At reset, $\text{PA}_2/\text{CKA}_1/\overline{\text{TEND}_0}$ is configured as a port A input. The function of this pin depends on the combination of bit 2 in the port A disable register (DERA2) and the CKA1D bit in the ASCI control register channel 1 (table 5).

Table 5. $\text{PA}_2/\text{CKA}_1/\overline{\text{TEND}_0}$ State

DERA2	CKA1D	Pin Function
0	0, 1	PA_2
1	0	CKA_1
	1	$\overline{\text{TEND}_0}$

PA$_4$/RXS/$\overline{\text{CTS}}_1$

At reset, PA$_4$/RXS/$\overline{\text{CTS}}_1$ is configured as a port A input. The function of this pin depends on the combination of bit 4 in the port A disable register (DERA4) and the CTS1E bit in the ASCI status register channel 1 (table 6).

Table 6. PA$_4$/RXS/$\overline{\text{CTS}}_1$ State

DERA4	CTS1E	Pin Function
0	0, 1	PA$_4$
1	0	RXS
	1	$\overline{\text{CTS}}_1$

CKA$_0$/$\overline{\text{DREQ}}_0$

CKA$_0$/$\overline{\text{DREQ}}_0$ is configured as the CKA$_0$ at reset. When either the DM1 or SM1 bit of the DMA mode registers 1, this bit is forcibly configured as the $\overline{\text{DREQ}}_0$ input, even if it has been configured as an output pin.

PG$_0$/AN$_0$, PG$_1$/AN$_1$, PG$_2$/AN$_2$, PG$_3$/AN$_3$, PG$_4$/AN$_4$, PG$_5$/AN$_5$

These pins cannot be configured as parallel port input pins (TTL-level input pins) alternate with analog comparator input pins. When using these pins as a TTL input port, read the port G input data register (IDRG).

When using these pins as an analog comparator's channel input, read the comparator control/status register (CCSR).

■ Absolute Maximum Ratings

Item	Symbol	Value	Unit
Supply Voltage	V_{CC}	−0.3 to +7.0	V
Input Voltage	V_{in}	−0.3 to V_{CC}+0.3	V
Operating Temperature	T_{opr}	−20 to +75	°C
Storage Temperature	T_{stg}	−55 to +150	°C

Note: Permanent LSI damage may occur if maximum ratings are exceeded. Normal operation should be under recommended operating conditions. If these conditions are exceeded, it could effect reliability of LSI.
Storage Temperature of the HD647180X is T_{stg} = −55 ~ +125°C.

■ ELECTRICAL CHARACTERISTICS

• DC Characteristics (V_{CC} = 5V ± 10%, V_{SS} = 0V, Ta = -20 to +75°C, unless otherwise noted)

Symbol	Item		Min	Typ	Max	Unit	Condition
V_{IH1}	Input High Voltage $\overline{\text{RESET}}$, EXTAL, $\overline{\text{NMI}}$		V_{CC}−0.6	−	V_{CC}+0.3	V	
V_{IH2}	Input High Voltage Except $\overline{\text{RESET}}$, EXTAL, $\overline{\text{NMI}}$		2.0	−	V_{CC}+0 3	V	
V_{IL1}	Input Low Voltage $\overline{\text{RESET}}$, EXTAL, $\overline{\text{NMI}}$		−0.3	−	0.6	V	
V_{IL2}	Input Low Voltage Except $\overline{\text{RESET}}$, EXTAL, $\overline{\text{NMI}}$		−0.3	−	0.8	V	
V_{OH}	Output High Voltage All outputs		2.4	−	−	V	I_{OH} = −200 μA
			V_{CC}− 1 2	−	−		I_{OH} = −20 μA
V_{OL}	Output Low Voltage All Outputs		−	−	0.45	V	I_{OL} = 2.2 mA
I_{IL}	Input Leakage Current All Inputs Except XTAL, EXTAL, $\overline{\text{RESET}}$		−	−	1 0	μA	Vin=0.5 to V_{CC} − 0.5 V
I_{TL}	Three State Leakage Current		−	−	1.0	μA	Vin=0.5 to V_{CC} − 0.5 V
I_{CC} (Note)	Power Dissipation (Normal Operation)		−	20	40	mA	f= 4 MHz
			−	25	50		f= 6 MHz
			−	30	60		f= 8 MHz
	Power Dissipation (System Stop Mode)		−	5	10	mA	f= 4 MHz
			−	6.3	12.5		f= 6 MHz
			−	7.5	15		f= 8 MHz
Cp	Pin Capacitance	$\overline{\text{RESET}}$	−	−	120	pF	Vin=0V, f=1 MHz Ta=25°C
		Except $\overline{\text{RESET}}$			20		

Note. V_{IHmin} = V_{CC} − 1.0 V, V_{ILmax} = 0.8 V (All input pins except $\overline{\text{RESET}}$, EXTAL $\overline{\text{NMI}}$)
V_{IHmin} = V_{CC} − 0.6 V, V_{ILmax} = 0.6 V ($\overline{\text{RESET}}$, EXTAL, $\overline{\text{NMI}}$)
(all output terminals are at no load.)

Symbol	Item		Min	Typ	Max	Unit	Condition
V_{IHP}	Input High-Level Voltage		2 0	–	V_{CC} + 0.3	V	
V_{ILP}	Input Low-Level Voltage		−0.3	–	0 8	V	
V_{OHP}	Output High-Level Voltage		2.4	–	–	V	I_{OH}=−200 μA
			V_{CC}−1 2	–	–		I_{OH}=−20 μA
V_{OLP}	Output Low-Level Voltage		–	–	0.45	V	* I_{OL}=2.2 mA
			–	–	1.0		** I_{OL}=10 mA
V_{in}	Analog Comparator Input Level Voltage	High level	V_{ref}+0 1	–	–	V	
		Low level	–	–	V_{ref}−0 1		
V_{ref}		V_{TH}	0	–	V_{CC}×0.8	V	
I_{ILP}	Input Leak Current		–	–	1.0	μA	V_{in}=0.5 to V_{CC} − 0.5

Note. *: Port A-F
** Port F only

• AC Characteristics (V_{SS} = 0V, Ta = –20 to +75°C, unless otherwise noted)

			HD641180X-4 HD643180X-4 HD647180X-4		HD641180X-6 HD643180X-6 HD647180X-6		HD641180X-8L HD643180X-8L HD647180X-8L		
Symbol	Item		min	max	min	max	min	max	unit
V_{CC}	Power Supply		4 5	5 5	4 5	5 5	4 75	5 25	V
t_{cyc}	Clock Cycle Time		250	2000	162	250	125	250	ns
t_{CHW}	Clock High Pulse Width		110	–	65	–	50	–	ns
t_{CLW}	Clock Low Pulse Width		110	–	65	–	50	–	ns
t_{cf}	Clock Fall Time		–	15	–	15	–	15	ns
t_{cr}	Clock Rise Time		–	15	–	15	–	15	ns
t_{ECYC}	External Clock Cycle Time		125	1000	81	125	62 5	125	ns
t_{EXHW}	External Clock High Pulse Width		50	–	30	–	25	–	ns
t_{EXLW}	External Clock Low Pulse Width		50	–	30	–	25	–	ns
t_{EXr} (Note 1)	External Clock Rise Time		–	25	–	25	–	25	ns
t_{EXf} (Note 1)	External Clock Fall time		–	25	–	25	–	25	ns
t_{AD}	Address Delay Time		–	100	–	75	–	65	ns
t_{AS}	Address Set-up Time ($\overline{ME}$ or $\overline{IOE}$ ↓)		50	–	30	–	20	–	ns
t_{MED1}	$\overline{ME}$ Delay Time 1		–	75	–	45	–	45	ns
t_{RDD1}	$\overline{RD}$ Delay Time 1	$\overline{IOC}$=1	–	75	–	45	–	45	ns
		$\overline{IOC}$=0	–	80	–	50	–	45	
t_{LD1}	$\overline{LIR}$ Delay Time 1		–	100	–	80	–	70 (Note 2)	ns
t_{AH}	Address Hold Time 1 ($\overline{ME}$, $\overline{IOE}$, $\overline{RD}$ or $\overline{WR}$ ↑)		80	–	35	–	20	–	ns
t_{MED2}	$\overline{ME}$ Delay Time 2		–	75	–	45	–	45	ns
t_{RDD2}	$\overline{RD}$ Delay Time 2		–	75	–	45	–	45	ns
t_{LD2}	$\overline{LIR}$ Delay Time 2		–	100	–	80	–	70 (Note 2)	ns
t_{DRS}	Data Read Set-up Time		60	–	55	–	45	–	ns
t_{DRH}	Data Read Hold Time		0	–	0	–	0	–	ns
t_{STD1}	ST Delay Time 1		–	110	–	90	–	70	ns
t_{STD2}	ST Delay Time 2		–	110	–	90	–	70	ns
t_{WS}	$\overline{WAIT}$ Set-up Time		80	–	40	–	40	–	ns
t_{WH}	$\overline{WAIT}$ Hold Time		70	–	40	–	40	–	ns
t_{WDZ}	Write Data Floating Delay Time		–	100	–	95	–	70	ns
t_{WRD1}	$\overline{WR}$ Delay Time 1		–	80	–	50	–	45	ns
t_{WDD}	Write Data Delay Time		–	110	–	90	–	80	ns
t_{WDS}	Write Data Set-up Time ($\overline{WR}$ ↓)		60	–	40	–	20	–	ns

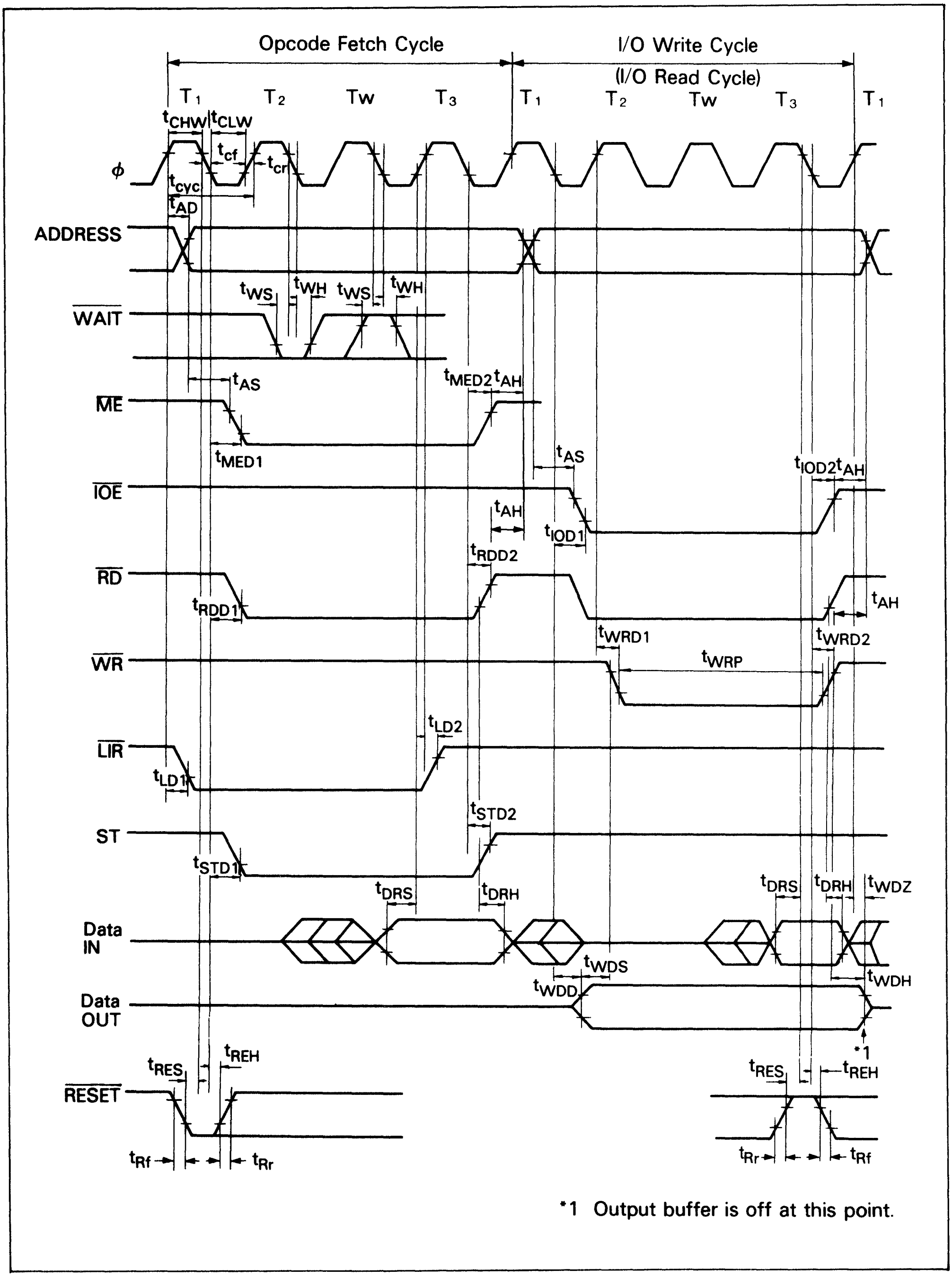
Note 1 External clock rise/fall time (t_{EXr}, t_{EXf}) may be shortened for satisfying external clock pulse width (t_{EXHW}, t_{EXLW})

Note 2 For a loading capacitance of less than or equal to 40 picofarads and operating temperature from 0 to 50 degrees, substract 10 nanoseconds from the value given in the maximum columns

			HD641180X-4 HD643180X-4 HD647180X-4		HD641180X-6 HD643180X-6 HD647180X-6		HD641180X-8L HD643180X-8L HD647180X-8L		
Symbol	Item		min	max	min	max	min	max	unit
t_{WRD2}	$\overline{\text{WR}}$ Delay Time 2		–	80	–	50	–	45	ns
t_{WRP}	$\overline{\text{WR}}$ Pulse Width		280	–	170	–	130	–	ns
t_{WDH}	Write Data Hold Time ($\overline{\text{WR}}$ ↑)		60	–	40	–	15	–	ns
t_{IOD1}	$\overline{\text{IOE}}$ Delay Time 1	$\overline{\text{IOC}}$= 1	–	75	–	45	–	45	ns
		$\overline{\text{IOC}}$=0	–	80	–	50	–	45	
t_{IOD2}	$\overline{\text{IOE}}$ Delay Time 2		–	75	–	45	–	45	ns
t_{IOD3}	$\overline{\text{IOE}}$ Delay Time 3 ($\overline{\text{LIR}}$ ↓)		540	–	340	–	250	–	ns
t_{INTS}	$\overline{\text{INT}}$ Set-up Time (ϕ ↓)		80	–	50	–	40	–	ns
t_{INTH}	$\overline{\text{INT}}$ Hold Time (ϕ ↓)		70	–	40	–	40	–	ns
t_{NMIW}	$\overline{\text{NMI}}$ Pulse Width		120	–	120	–	100	–	ns
t_{BRS}	$\overline{\text{BUSREQ}}$ Set-up Time (ϕ ↓)		80	–	50	–	40	–	ns
t_{BRH}	$\overline{\text{BUSREQ}}$ Hold Time (ϕ ↓)		70	–	40	–	40	–	ns
t_{BAD1}	$\overline{\text{BUSACK}}$ Delay Time 1		–	100	–	95	–	70	ns
t_{BAD2}	$\overline{\text{BUSACK}}$ Delay Time 2		–	100	–	95	–	70	ns
t_{BZD}	Bus Floating Delay Time		–	130	–	125	–	90	ns
t_{MEWH}	$\overline{\text{ME}}$ Pulse Width (HIGH)		200	–	110	–	90	–	ns
t_{MEWL}	$\overline{\text{ME}}$ Pulse Width (LOW)		210	–	125	–	100	–	ns
t_{RFD1}	$\overline{\text{REF}}$ Delay Time 1		–	110	–	90	–	80	ns
t_{RFD2}	$\overline{\text{REF}}$ Delay Time 2		–	110	–	90	–	80	ns
t_{HAD1}	$\overline{\text{HALT}}$ Delay Time 1		–	110	–	90	–	80	ns
t_{HAD2}	$\overline{\text{HALT}}$ Delay Time 2		–	110	–	90	–	80	ns
t_{DRQS}	$\overline{\text{DREQi}}$ Set-up Time		80	–	50	–	40	–	ns
t_{DRQH}	$\overline{\text{DREQi}}$ Hold Time		70	–	40	–	40	–	ns
t_{TED1}	$\overline{\text{TENDi}}$ Delay Time 1		–	85	–	70	–	60	ns
t_{TED2}	$\overline{\text{TENDi}}$ Delay Time 2		–	85	–	70	–	60	ns
t_{ED1}	Enable Delay Time 1		–	100	–	95	–	70	ns
t_{ED2}	Enable Delay Time 2		–	100	–	95	–	70	ns
P_{WEH}	E Pulse Width (HIGH)		150	–	75	–	65	–	ns
P_{WEL}	E Pulse Width (LOW)		300	–	180	–	130	–	ns

Symbol	Item	HD641180X-4 HD643180X-4 HD647180X-4		HD641180X-6 HD643180X-6 HD647180X-6		HD641180X-8L HD643180X-8L HD647180X-8L		unit
		min	max	min	max	min	max	
t_{Er}	Enable Rise Time	—	25	—	20	—	20	ns
t_{Ef}	Enable Fall Time	—	25	—	20	—	20	ns
t_{TOD}	Timer Output Delay Time	—	300	—	300	—	200	ns
t_{STDI}	CSI/O Transmit Data Delay Time (Internal Clock Operation)	—	200	—	200	—	200	ns
t_{STDE}	CSI/O Transmit Data Delay Time (External Clock Operation)	—	7.5tcyc +300	—	7.5tcyc +300	—	7.5tcyc +200	ns
t_{SRSI}	CSI/O Receive Data Set-up Time (Internal Clock Operation)	1	—	1	—	1	—	tcyc
t_{SRHI}	CSI/O Receive Data Hold Time (Internal Clock Operation)	1	—	1	—	1	—	tcyc
t_{SRSE}	CSI/O Receive Data Set-up Time (External Clock Operation)	1	—	1	—	1	—	tcyc
t_{SRHE}	CSI/O Receive Data Hold Time (External Clock Operation)	1	—	1	—	1	—	tcyc
t_{RES}	$\overline{\text{RESET}}$ Set-up Time	120	—	120	—	100	—	ns
t_{REH}	$\overline{\text{RESET}}$ Hold Time	80	—	80	—	70	—	ns
t_{OSC}	Oscillator Stabilization Time	—	20	—	20	—	20	ms
t_{EXr}	External Clock Rise Time (EXTAL)	—	25	—	25	—	25	ns
t_{EXf}	External Clock Fall Time (EXTAL)	—	25	—	25	—	25	ns
t_{Rr}	$\overline{\text{RESET}}$ Rise Time	—	50	—	50	—	50	ms
t_{Rf}	$\overline{\text{RESET}}$ Fall Time	—	50	—	50	—	50	ms
t_{Ir}	Input Rise Time (except EXTAL, $\overline{\text{RESET}}$)	—	100	—	100	—	100	ns
t_{If}	Input Fall Time (except EXTAL, $\overline{\text{RESET}}$)	—	100	—	100	—	100	ns
t_{PWD}	Port Data Output Delay Time	—	110	—	90	—	80	ns
t_{PDSU}	Port Data Input Setup Time	80	—	50	—	50	—	ns
t_{PDH}	Port Data Input Hold Time	60	—	40	—	40	—	ns

The HD643180X differs from HD647180X in chip design and manufacturing process Be careful when using the HD647180X system for the HD643180X since characteristics values are not exactly the same though guaranteed values are identical



*1 Output buffer is off at this point.

Figure 4. CPU Timing (Opcode Fetch Cycle) I/O Write Cycle (I/O Read Cycle) When IOC = 1

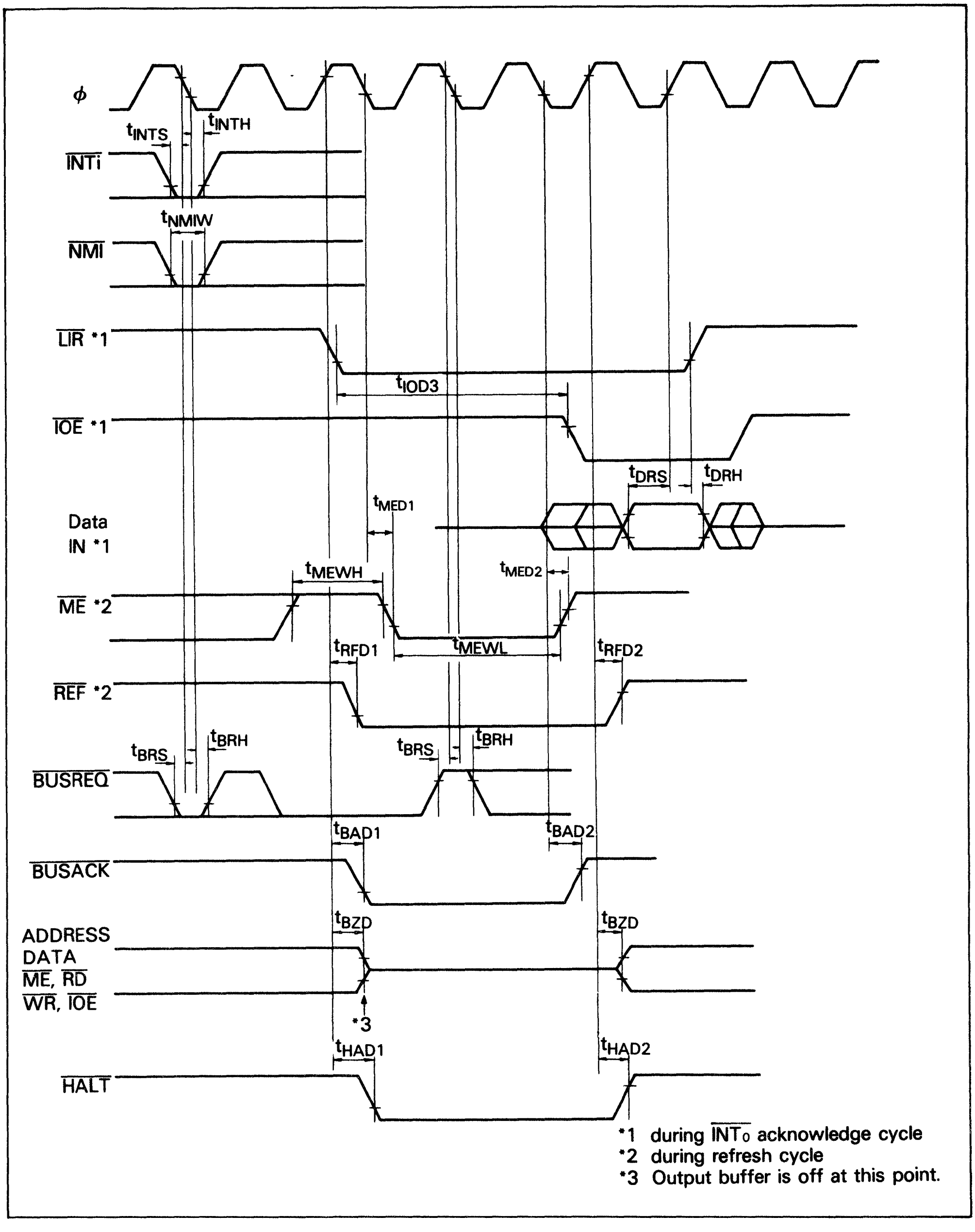


**Figure 5. CPU Timing ($\overline{INT_0}$ Acknowledge Cycle,
Refresh Cycle,
Bus Release Mode,
Halt Mode,
Sleep Mode,
System Stop Mode When $\overline{IOC}$ = 1)**

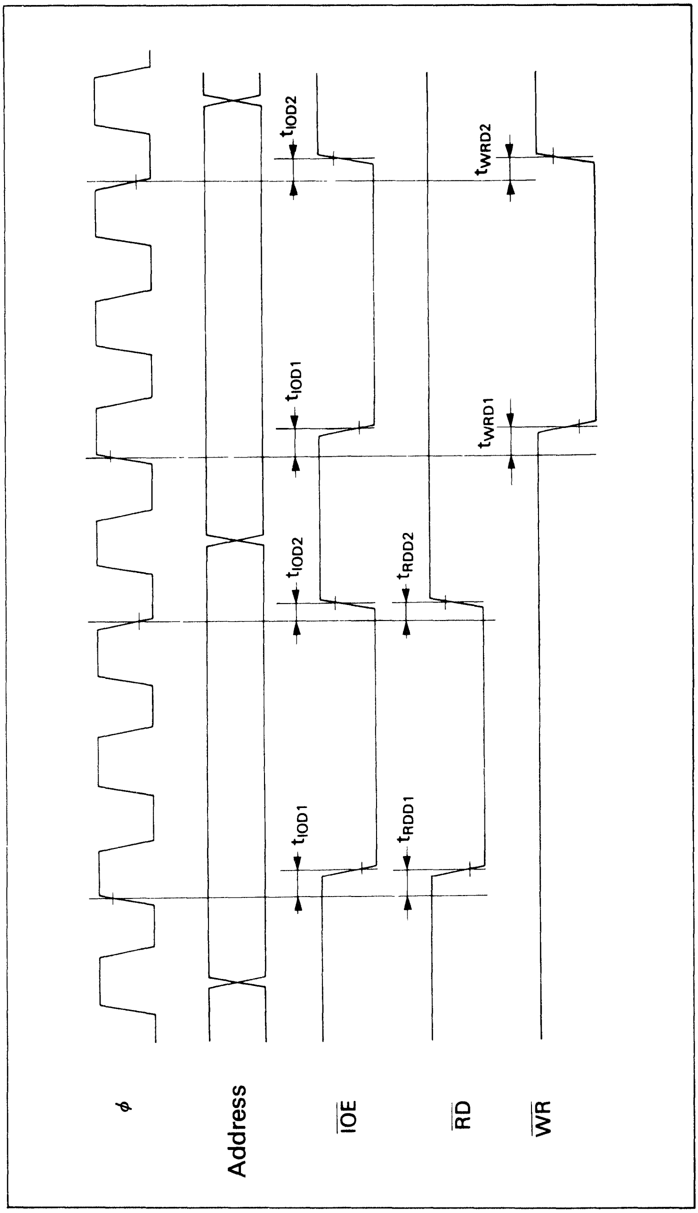


Figure 6. CPU Timing ($\overline{\text{IOC}}$ = 0)

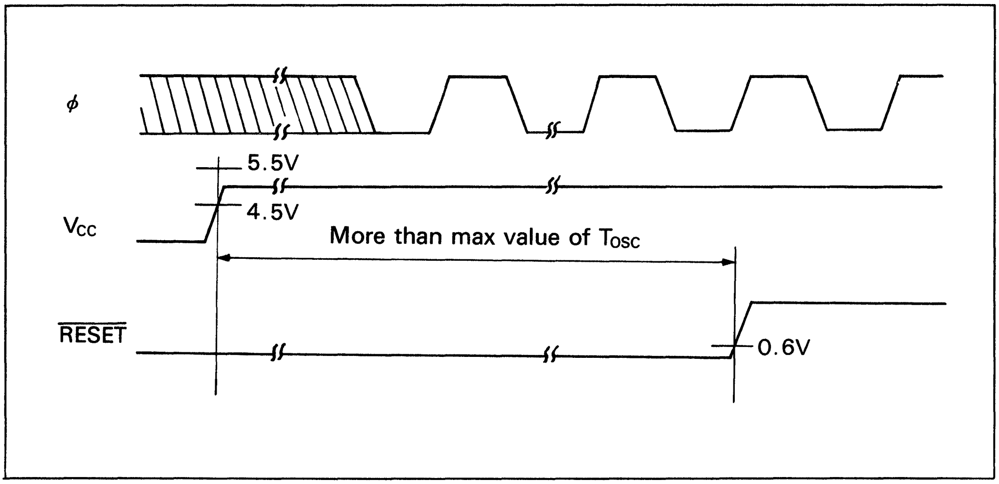
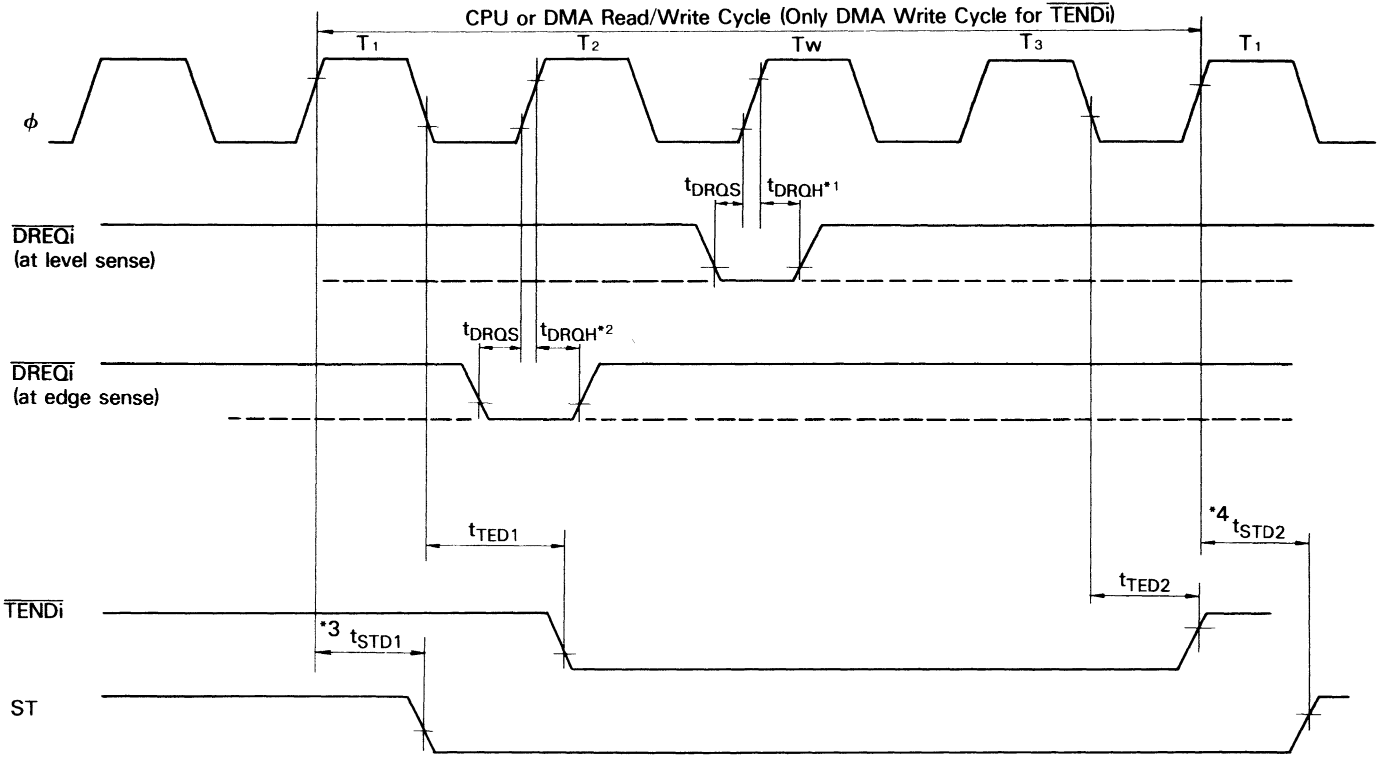


Figure 7. CPU Timing



*1 t_{DRQS} and t_{DRQH} are specified for the rising edge of clock followed by T_3.
*2 t_{DRQS} and t_{DRQH} are specified for the rising edge of clock.
*3 DMA cycle starts.
*4 CPU cycle starts.

Figure 8. DMA Control Signals

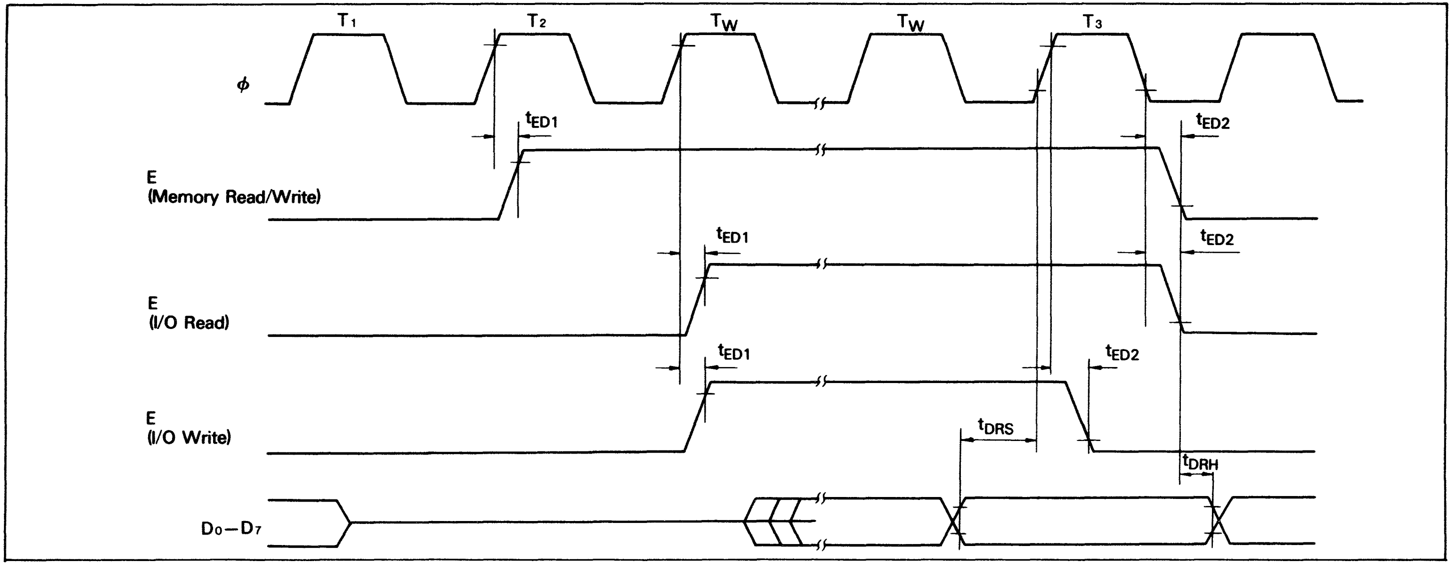


Figure 8A. E Clock Timing (Memory Read/Write Cycle, I/O Read/Write Cycle)

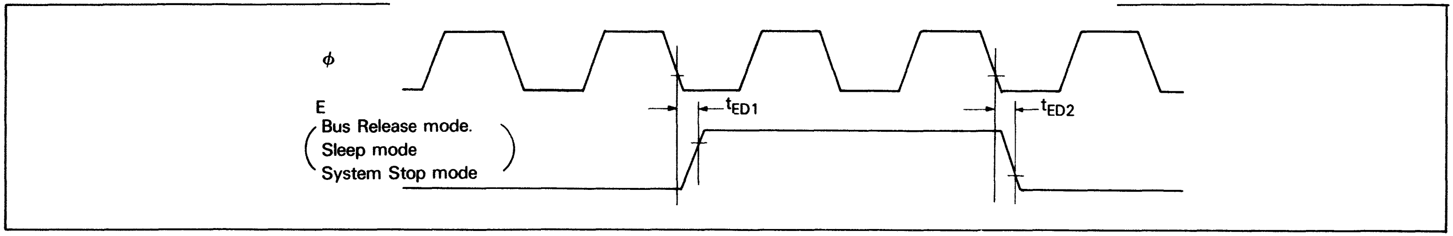


Figure 9. E Clock Timing (Bus Release Mode, Sleep Mode, System Stop Mode)

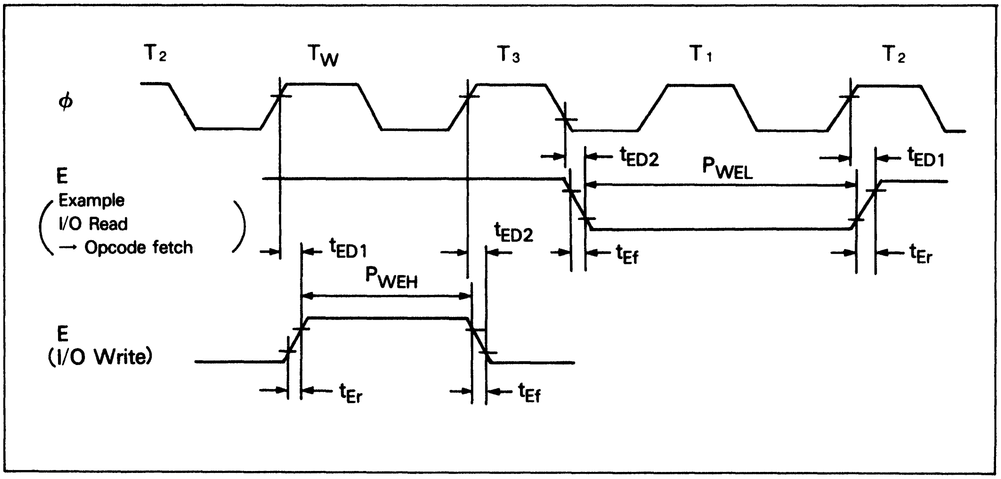


Figure 9A. E Clock Timing (Minimum Timing Example of P_{WEL} and P_{WEH})

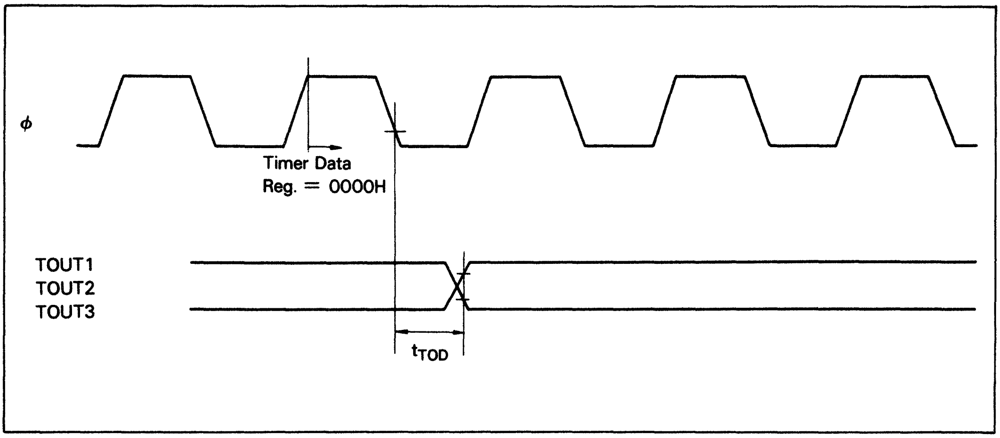


Figure 10. Timer Output Timing

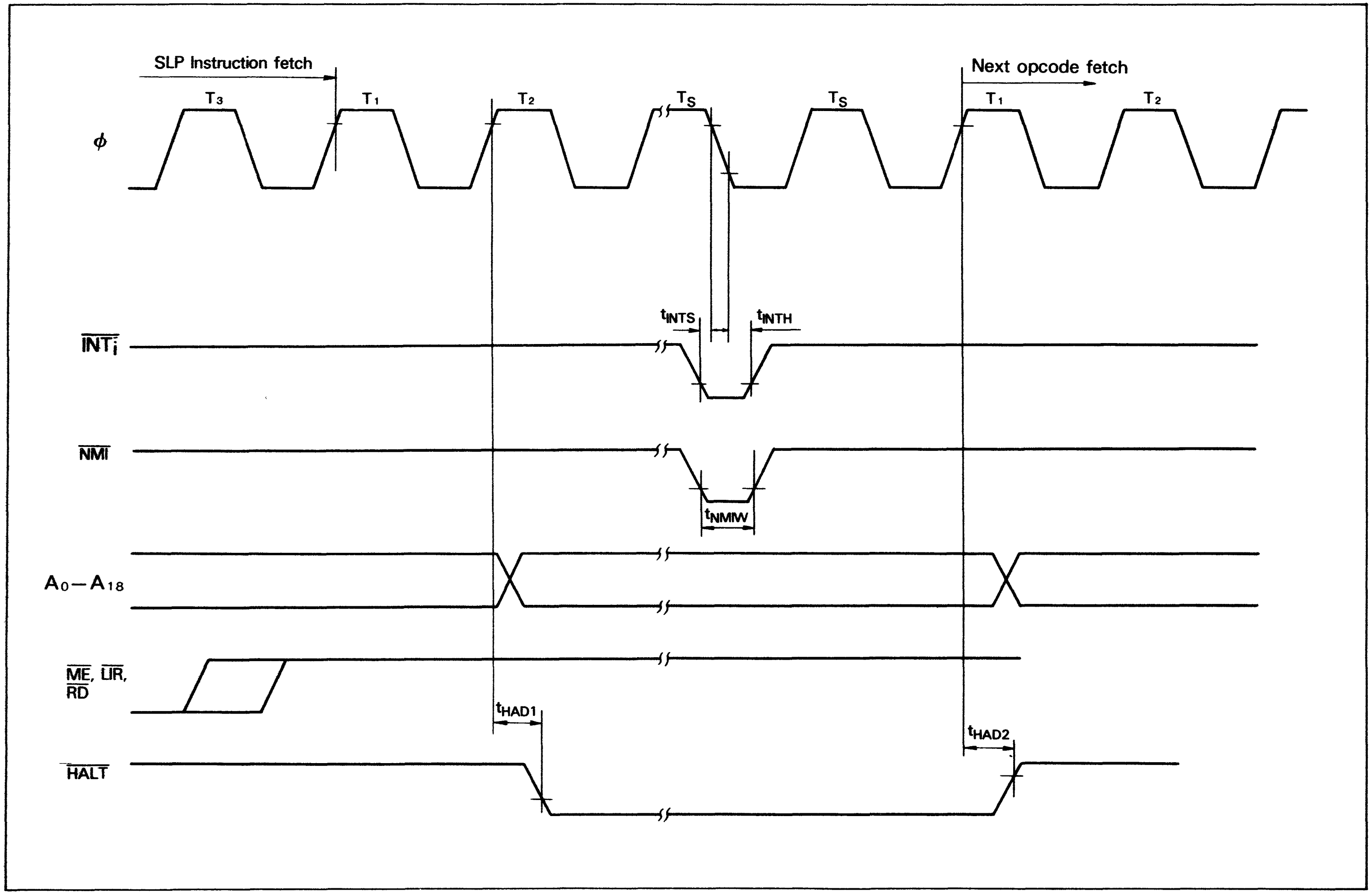


Figure 11. SLP Execution Cycle

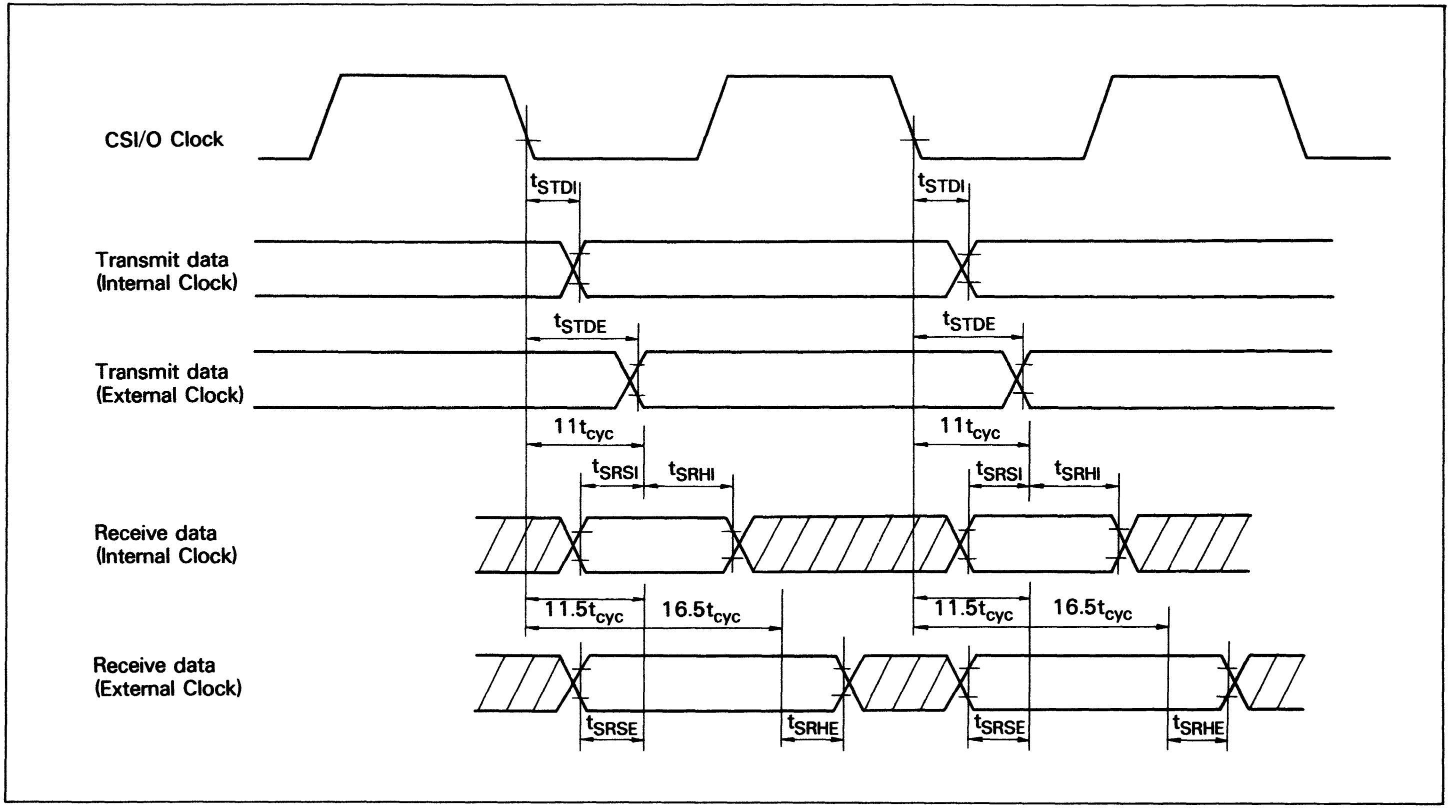

CSI/O Clock
Transmit data (Internal Clock)
Transmit data (External Clock)
Receive data (Internal Clock)
Receive data (External Clock)
t_{STDI}
t_{STDE}
$11t_{cyc}$
t_{SRSI}
t_{SRHI}
$11.5t_{cyc}$
$16.5t_{cyc}$
t_{SRSE}
t_{SRHE}

Figure 12. CSI/O Receive/Transmit Timing

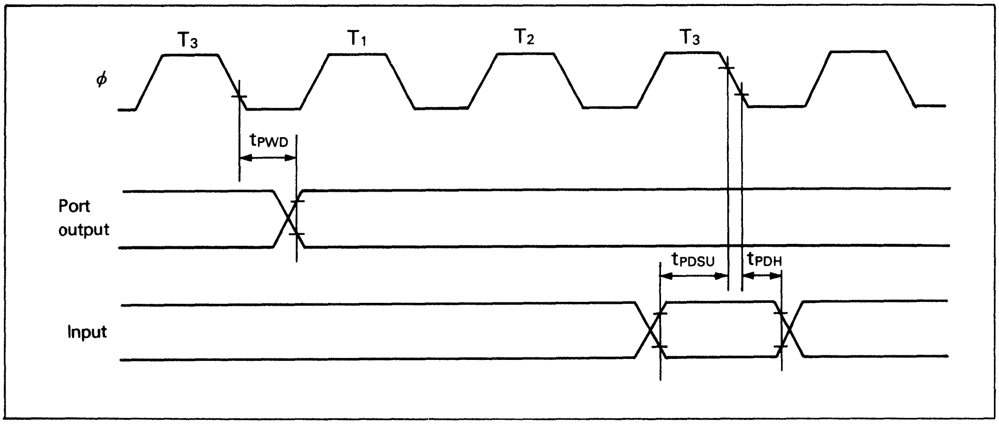


Figure 13. Port Input and Output Timing

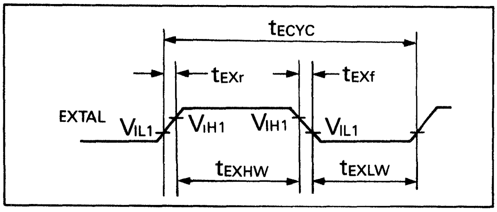


Figure 14. External Clock Rise Time and Fall Time

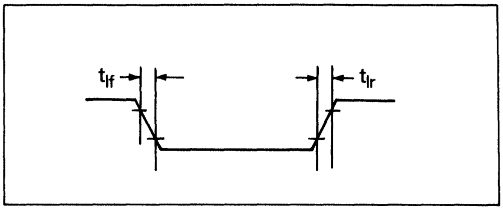


Figure 15. Input Rise Time and Fall Time (Except EXTAL, $\overline{\text{RESET}}$

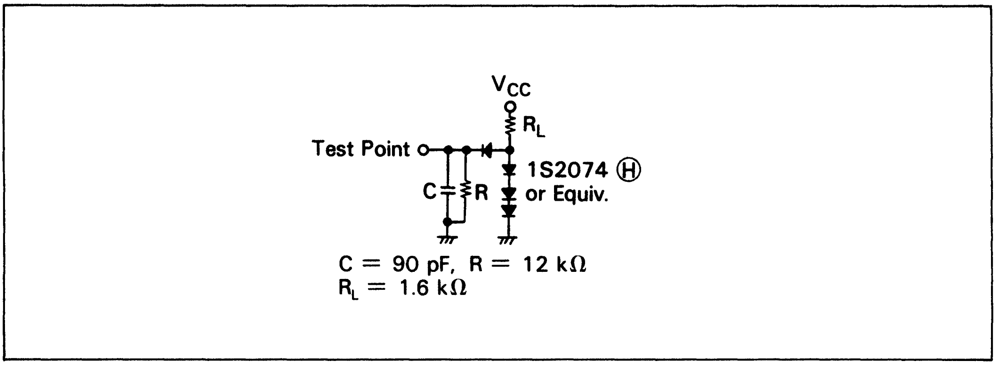


Figure 16. Bus Timing Test Load (TTL Load)

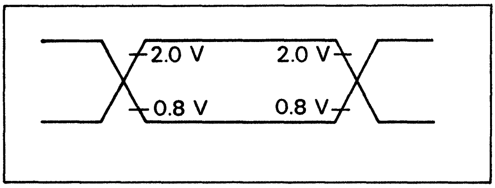


Figure 17. Reference Level (Input)

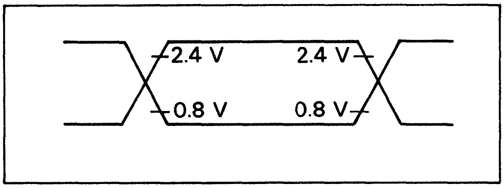


Figure 18. Reference Level (Output)

■ INSTRUCTION SET

Register

g, g', ww, xx, yy, and zz specify a register to be used. g and g' specify an 8-bit register. ww, xx, yy, and zz specify a 16-bit pair of 8-bit registers. Table 7 shows the correspondence between symbols and registers.

Table 7 Register Specification

g,g'	Reg.	ww	Reg.	xx	Reg.	yy	Reg.	zz	Reg.
000	B	00	BC	00	BC	00	BC	00	BC
001	C	01	DE	01	DE	01	DE	01	DE
010	D	10	HL	10	IX	10	IY	10	HL
011	E	11	SP	11	SP	11	SP	11	AF
100	H								
101	L								
111	A								

Note: H and L suffixed to ww,xx,yy,zz (ex. wwH, IXL) indicate upper and lower 8 bits of the 16-bit register, respectively.

Bit

b specifies a bit to be manipulated in the bit manipulation instruction. Table 8 shows the correspondence between b and bits.

Table 8 Bit Specification

b	Bit
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

Condition

f specifies the condition in program control instructions. Table 9 shows the correspondence between f and conditions.

Table 9 Condition Specification

f	Condition	
000	NZ	non zero
001	Z	zero
010	NC	non carry
011	C	carry
100	PO	parity odd
101	PE	parity even
110	P	sign plus
111	M	sign minus

Restart Address

v specifies a restart address. Table A-4 shows the correspondence between v and restart addresses.

Table 10 Restart Address Specification

v	Address
000	00H
001	08H
010	10H
011	18H
100	20H
101	28H
110	30H
111	38H

Flag

The following symbols show the flag conditions:

· : not affected
↕ : affected
× : undefined
S : set to 1
R : reset to 0
P : parity
V : overflow

Miscellaneous

$(\quad)_M$: Data in the memory address
$(\quad)_I$: Data in the I/O address
m or n : 8-bit data
mn : 16-bit data
r : 8-bit register
R : 16-bit register
$b \cdot (\quad)_M$: Contents of bit b in the memory address
b·gr : Contents of bit b in the register gr
d or j : 8-bit signed displacement
S : Source addressing mode
D : Destination addressing mode
· : AND operation
+ : OR operation
+ : EXCLUSIVE OR operation
** : Added new instructions to Z80

Instruction Summary

Data Manipulation Instructions

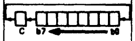
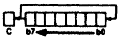
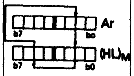
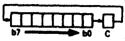
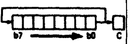
Table 11 Arithmetic and Logical Instructions (8 Bit)

Operation Name	Mnemonics	Opcode	Addressing							Bytes	States	Operation	Flag					
													7	6	4	2	1	0
			IMMED	EXT	IND	REG	REGI	IMP	REL				S	Z	H	P/V	N	C
ADD	ADD A,g	10 000 g				S		D		1	4	Ar+gr→Ar	↕	↕	↕	V	R	↕
	ADD A,(HL)	10 000 110					S	D		1	6	Ar+(HL)$_M$→Ar	↕	↕	↕	V	R	↕
	ADD A,m	11 000 110 〈 m 〉	S					D		2	6	Ar+m→Ar	↕	↕	↕	V	R	↕
	ADD A,(IX+d)	11 011 101 10 000 110 〈 d 〉			S			D		3	14	Ar+(IX+d)$_M$→Ar	↕	↕	↕	V	R	↕
	ADD A,(IY+d)	11 111 101 10 000 110 〈 d 〉			S			D		3	14	Ar+(IY+d)$_M$→Ar	↕	↕	↕	V	R	↕
ADC	ADC A,g	10 001 g				S		D		1	4	Ar+gr+c→Ar	↕	↕	↕	V	R	↕
	ADC A,(HL)	10 001 110					S	D		1	6	Ar+(HL)$_M$+c→Ar	↕	↕	↕	V	R	↕
	ADC A,m	11 001 110 〈 m 〉	S					D		2	6	Ar+m+c→Ar	↕	↕	↕	V	R	↕
	ADC A,(IX+d)	11 011 101 10 001 110 〈 d 〉			S			D		3	14	Ar+(IX+d)$_M$+c→Ar	↕	↕	↕	V	R	↕
	ADC A,(IY+d)	11 111 101 10 001 110 〈 d 〉			S			D		3	14	Ar+(IY+d)$_M$+c→Ar	↕	↕	↕	V	R	↕
AND	AND g	10 100 g				S		D		1	4	Ar·gr→Ar	↕	↕	S	P	R	R
	AND HL)	10 100 110					S	D		1	6	Ar·(HL)$_M$→Ar	↕	↕	S	P	R	R
	AND m	11 100 110 〈 m 〉	S					D		2	6	Ar·m→Ar	↕	↕	S	P	R	R
	AND (IX+d)	11 011 101 10 100 110 〈 d 〉			S			D		3	14	Ar·(IX+d)$_M$→Ar	↕	↕	S	P	R	R
	AND (IY+d)	11 111 101 10 100 110 〈 d 〉			S			D		3	14	Ar·(IY+d)$_M$→Ar	↕	↕	S	P	R	R
Compare	CP g	10 111 g				S		D		1	4	Ar−gr	↕	↕	↕	V	S	↕
	CP (HL)	10 111 110					S	D		1	6	Ar− HL)$_M$	↕	↕	↕	V	S	↕
	CP m	11 111 110 〈 m 〉	S					D		2	6	Ar−m	↕	↕	↕	V	S	↕
	CP (IX+d)	11 011 101 10 111 110 〈 d 〉			S			D		3	14	Ar−(IX+d)$_M$	↕	↕	↕	V	S	↕
	CP (IY+d)	11 111 101 10 111 110 〈 d 〉			S			D		3	14	Ar−(IY+d)$_M$	↕	↕	↕	V	S	↕
Complement	CPL	00 101 111						S/D		1	3	$\overline{Ar}$→Ar	·		S	·	S	·
DEC	DEC g	00 g 101				S/D				1	4	gr−1→gr	↕	↕	↕	V	S	·
	DEC (HL)	00 110 101					S/D			1	10	(HL)$_M$−1→(HL)$_M$	↕	↕	↕	V	S	·
	DEC (IX+d)	11 011 101 00 110 101 〈 d 〉			S/D					3	18	(IX+d)$_M$−1→(IX+d)$_M$	↕	↕	↕	V	S	·
	DEC (IY+d)	11 111 101 00 110 101 〈 d 〉			S/D					3	18	(IY+d)$_M$−1→(IY+d)$_M$	↕	↕	↕	V	S	·
INC	INC g	00 g 100				S/D				1	4	gr+1→gr	↕	↕	↕	V	R	·
	INC (HL)	00 110 100					S/D			1	10	(HL)$_M$+1→(HL)$_M$	↕	↕	↕	V	R	·
	INC (IX+d)	11 011 101 00 110 100 〈 d 〉			S/D					3	18	(IX+d)$_M$+1→(IX+d)$_M$	↕	↕	↕	V	R	·

Table 11 Arithmetic and Logical Instructions (8 Bit) (cont)

Operation Name	Mnemonics	Opcode	Addressing							Bytes	States	Operation	Flag					
													7	6	4	2	1	0
			IMMED	EXT	IND	REG	REGI	IMP	REL				S	Z	H	P/V	N	C
MULT	MLT ww **	11 101 101				S/D				2	17	wwHr×wwLr→ww$_R$	·	·	·	·	·	·
		01 ww1 100																
Negate	NEG	11 101 101						S/D		2	6	0−Ar→Ar	↕	↕	↕	V	S	↕
		01 000 100																
OR	OR g	10 110 g				S		D		1	4	Ar+gr→Ar	↕	↕	R	P	R	R
	OR (HL)	10 110 110					S	D		1	6	Ar+(HL)$_M$→Ar	↕	↕	R	P	R	R
	OR m	11 110 110	S					D		2	6	Ar+m→Ar	↕	↕	R	P	R	R
		〈 m 〉																
	OR (IX+d)	11 011 101			S			D		3	14	Ar+(IX+d)$_M$→Ar	↕	↕	R	P	R	R
		10 110 110																
		〈 d 〉																
	OR (IY+d)	11 111 101			S			D		3	14	Ar+(IY+d)$_M$→Ar	↕	↕	R	P	R	R
		10 110 110																
		〈 d 〉																
SUB	SUB g	10 010 g				S		D		1	4	Ar−gr→Ar	↕	↕	↕	V	S	↕
	SUB (HL)	10 010 110					S	D		1	6	Ar−(HL)$_M$→Ar	↕	↕	↕	V	S	↕
	SUB m	11 010 110	S					D		2	6	Ar−m→Ar	↕	↕	↕	V	S	↕
		〈 m 〉																
	SUB (IX+d)	11 011 101			S			D		3	14	Ar−(IX+d)$_M$→Ar	↕	↕	↕	V	S	↕
		10 010 110																
		〈 d 〉																
	SUB (IY+d)	11 111 101			S			D		3	14	Ar−(IY+d)$_M$→Ar	↕	↕	↕	V	S	↕
		10 010 110																
		〈 d 〉																
SUBC	SBC A,g	10 011 g				S		D		1	4	Ar−gr−c→Ar	↕	↕	↕	V	S	↕
	SBC A,(HL)	10 011 110					S	D		1	6	Ar−(HL)$_M$−c→Ar	↕	↕	↕	V	S	↕
	SBC A,m	11 011 110	S					D		2	6	Ar−m−c→Ar	↕	↕	↕	V	S	↕
		〈 m 〉																
	SBC A,(IX+d)	11 011 101			S			D		3	14	Ar−(IX+d)$_M$−c→Ar	↕	↕	↕	V	S	↕
		10 011 110																
		〈 d 〉																
	SBC A,(IY+d)	11 111 101			S			D		3	14	Ar−(IY+d)$_M$−c→Ar	↕	↕	↕	V	S	↕
		10 011 110																
		〈 d 〉																
Test	TST g **	11 101 101				S				2	7	Ar·gr	↕	↕	S	P	R	R
		00 g 100																
	TST (HL) **	11 101 101					S			2	10	Ar·(HL)$_M$	↕	↕	S	P	R	R
		00 110 100																
	TST m **	11 101 101	S							3	9	Ar·m	↕	↕	S	P	R	R
		01 100 100																
		〈 m 〉																
XOR	XOR g	10 101 g				S		D		1	4	Ar⊕gr→Ar	↕	↕	R	P	R	R
	XOR (HL)	10 101 110					S	D		1	6	Ar⊕(HL)$_M$→Ar	↕	↕	R	P	R	R
	XOR m	11 101 110	S					D		2	6	Ar⊕m→Ar	↕	↕	R	P	R	R
		〈 m 〉																
	XOR (IX+d)	11 011 101			S			D		3	14	Ar⊕(IX+d)$_M$→Ar	↕	↕	R	P	R	R
		10 101 110																
		〈 d 〉																
	XOR (IY+d)	11 111 101			S			D		3	14	Ar⊕(IY+d)$_M$→Ar	↕	↕	R	P	R	R
		10 101 110																
		〈 d 〉																

Table 12 Rotate and Shift Instructions

Operation Name	Mnemonics	Opcode	Addressing							Bytes	States	Operation	Flag					
													7	6	4	2	1	0
			IMMED	EXT	IND	REG	REGI	IMP	REL				S	Z	H	P/V	N	C
Rotate and Shift Data	RLA	00 010 111						S/D		1	3		·	·	R	·	R	↕
	RL g	11 001 011 00 010 g				S/D				2	7		↕	↕	R	P	R	↕
	RL (HL)	11 001 011 00 010 110					S/D			2	13		↕	↕	R	P	R	↕
	RL (IX+d)	11 011 101 11 001 011 〈 d 〉 00 010 110			S/D					4	19		↕	↕	R	P	R	↕
	RL (IY+d)	11 111 101 11 001 011 〈 d 〉 00 010 110			S/D					4	19		↕	↕	R	P	R	↕
	RLCA	00 000 111						S/D		1	3		·	·	R	·	R	↕
	RLC g	11 001 011 00 000 g				S/D				2	7		↕	↕	R	P	R	↕
	RLC (HL)	11 001 011 00 000 110					S/D			2	13		↕	↕	R	P	R	↕
	RLC (IX+d)	11 011 101 11 001 011 〈 d 〉 00 000 110			S/D					4	19		↕	↕	R	P	R	↕
	RLC (IY+d)	11 111 101 11 001 011 〈 d 〉 00 000 110			S/D					4	19		↕	↕	R	P	R	↕
	RLD	11 101 101 01 101 111						S/D		2	16		↕	↕	R	P	R	·
	RRA	00 011 111						S/D		1	3		·	·	R	·	R	↕
	RR g	11 001 011 00 011 g				S/D				2	7		↕	↕	R	P	R	↕
	RR (HL)	11 001 011 00 011 110					S/D			2	13		↕	↕	R	P	R	↕
	RR (IX+d)	11 011 101 11 001 011 〈 d 〉 00 011 110			S/D					4	19		↕	↕	R	P	R	↕
	RR (IY+d)	11 111 101 11 001 011 〈 d 〉 00 011 110			S/D					4	19		↕	↕	R	P	R	↕
	RRCA	00 001 111						S/D		1	3		·	·	R	·	R	↕
	RRC g	11 001 011 00 001 g				S/D				2	7		↕	↕	R	P	R	↕
	RRC (HL)	11 001 011 00 001 110					S/D			2	13		↕	↕	R	P	R	↕
	RRC (IX+d)	11 011 101 11 001 011 〈 d 〉 00 001 110			S/D					4	19		↕	↕	R	P	R	↕
	RRC (IY+d)	11 111 101 11 001 011 〈 d 〉 00 001 110			S/D					4	19		↕	↕	R	P	R	↕

(continued)

Table 12 Rotate and Shift Instructions (cont)

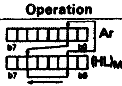
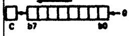
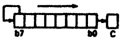
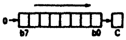
Operation Name	Mnemonics	Opcode	Addressing							Bytes	States	Operation	Flag					
													7	6	4	2	1	0
			IMMED	EXT	IND	REG	REGI	IMP	REL				S	Z	H	P/V	N	C
Rotate	RRD	11 101 101						S/D		2	16		↕	↕	R	P	R	·
and		01 100 111																
Shift	SLA g	11 001 011				S/D				2	7		↕	↕	R	P	R	↕
Data		00 100 g																
	SLA (HL)	11 001 011					S/D			2	13		↕	↕	R	P	R	↕
		00 100 110																
	SLA (IX+d)	11 011 101			S/D					4	19		↕	↕	R	P	R	↕
		11 001 011																
		〈 d 〉																
		00 100 110																
	SLA (IY+d)	11 111 101			S/D					4	19		↕	↕	R	P	R	↕
		11 001 011																
		〈 d 〉																
		00 100 110																
	SRA g	11 001 011				S/D				2	7		↕	↕	R	P	R	↕
		00 101 g																
	SRA (HL)	11 001 011					S/D			2	13		↕	↕	R	P	R	↕
		00 101 110																
	SRA (IX+d)	11 011 101			S/D					4	19		↕	↕	R	P	R	↕
		11 001 011																
		〈 d 〉																
		00 101 110																
	SRA (IY+d)	11 111 101			S/D					4	19		↕	↕	R	P	R	↕
		11 001 011																
		〈 d 〉																
		00 101 110																
	SRL g	11 001 011				S/D				2	7		↕	↕	R	P	R	↕
		00 111 g																
	SRL (HL)	11 001 011					S/D			2	3		↕	↕	R	P	R	↕
		00 111 110																
	SRL (IX+d)	11 011 101			S/D					4	19		↕	↕	R	P	R	↕
		11 001 011																
		〈 d 〉																
		00 111 110																
	SRL (IY+d)	11 111 101			S/D					4	19		↕	↕	R	P	R	↕
		11 001 011																
		〈 d 〉																
		00 111 110																

Table 13 Bit Manipulation Instructions

Operation Name	Mnemonics	Opcode	Addressing							Bytes	States	Operation	Flag					
													7	6	4	2	1	0
			IMMED	EXT	IND	REG	REGI	IMP	REL				S	Z	H	P/V	N	C
Bit Set	SET b,g	11 001 011 11 b g				S/D				2	7	$1\rightarrow b\cdot gr$	·	·	·	·	·	·
	SET b,(HL)	11 001 011 11 b 110					S/D			2	13	$1\rightarrow b\cdot(HL)_M$		·	·	·	·	·
	SET b,(IX+d)	11 011 101 11 001 011 〈 d 〉 11 b 110			S/D					4	19	$1\rightarrow b\cdot(IX+d)_M$	·	·	·	·	·	·
	SET b,(IY+d)	11 111 101 11 001 011 〈 d 〉 11 b 110			S/D					4	19	$1\rightarrow b\cdot(IY+d)_M$	·	·	·	·	·	·
Bit Reset	RES b,g	11 001 011 10 b g				S/D				2	7	$0\rightarrow b\cdot gr$		·	·	·	·	·
	RES b,(HL)	11 001 011 10 b 110					S/D			2	13	$0\rightarrow b\cdot(HL)_M$	·	·	·	·	·	·
	RES b,(IX+d)	11 011 101 11 001 011 〈 d 〉 10 b 110			S/D					4	19	$0\rightarrow b\cdot(IX+d)_M$		·	·	·	·	
	RES b,(IY+d)	11 111 101 11 001 011 〈 d 〉 10 b 110			S/D					4	19	$0\rightarrow b\cdot(IY+d)_M$	·	·	·	·		·
Bit Test	BIT b,g	11 001 011 01 b g				S				2	6	$\overline{b\cdot gr}\rightarrow z$	X	↕	S	X	R	
	BIT b,(HL)	11 001 011 01 b 110					S			2	9	$\overline{b\cdot(HL)_M}\rightarrow z$	X	↕	S	X	R	·
	BIT b,(IX+d)	11 011 101 11 001 011 〈 d 〉 01 b 110			S					4	15	$\overline{b\cdot(IX+d)_M}\rightarrow z$	X	↕	S	X	R	·
	BIT b,(IY+d)	11 111 101 11 001 011 〈 d 〉 01 b 110			S					4	15	$\overline{b\cdot(IY+d)_M}\rightarrow z$	X	↕	S	X	R	·

Table 14 Arithmetic Instructions (16 Bit)

			Addressing										Flag					
													7	6	4	2	1	0
Operation Name	Mnemonics	Opcode	IMMED	EXT	IND	REG	REGI	IMP	REL	Bytes	States	Operation	S	Z	H	P/V	N	C
ADD	ADD HL,ww	00 ww1 001				S		D		1	7	$HL_R+ww_R \rightarrow HL_R$	·	·	X	·	R	↕
	ADD IX,xx	11 011 101				S		D		2	10	$IX_R+xx_R \rightarrow IX_R$	·	·	X	·	R	↕
		00 xx1 001																
	ADD IY,yy	11 111 101				S		D		2	10	$IY_R+yy_R \rightarrow IY_R$	·	·	X	·	R	↕
		00 yy1 001																
ADC	ADC HL,ww	11 101 101				S		D		2	10	$HL_R+ww_R+c \rightarrow HL_R$	↕	↕	X	V	R	↕
		01 ww1 010																
DEC	DEC ww	00 ww1 011				S/D				1	4	$ww_R-1 \rightarrow ww_R$	·	·	·	·	·	·
	DEC IX	11 011 101						S/D		2	7	$IX_R-1 \rightarrow IX_R$	·	·	·	·	·	·
		00 101 011																
	DEC IY	11 111 101						S/D		2	7	$IY_R-1 \rightarrow IY_R$	·	·	·	·	·	·
		00 101 011																
INC	INC ww	00 ww0 011				S/D				1	4	$ww_R+1 \rightarrow ww_R$	·	·	·	·	·	·
	INC IX	11 011 101						S/D		2	7	$IX_R+1 \rightarrow IX_R$	·	·	·	·	·	·
		00 100 011																
	INC IY	11 111 101						S/D		2	7	$IY_R+1 \rightarrow IY_R$	·	·	·	·	·	·
		00 100 011																
SBC	SBC HL,ww	11 101 101				S		D		2	10	$HL_R-ww_R-c \rightarrow HL_R$	↕	↕	X	V	S	↕
		01 ww0 010																

Data Transfer Instructions

Table 15 8-Bit Load

Operation Name	Mnemonics	Opcode	Addressing							Bytes	States	Operation	Flag					
													7	6	4	2	1	0
			IMMED	EXT	IND	REG	REGI	IMP	REL				S	Z	H	P/V	N	C
Load 8-Bit Data	LD A,I	11 101 101 01 010 111						S/D		2	6	Ir→Ar	↕	↕	R	IEF_2	R	·
	LD A,R	11 101 101 01 011 111						S/D		2	6	Rr→Ar	↕	↕	R	IEF_2	R	·
	LD A,(BC)	00 001 010					S	D		1	6	$(BC)_M$→Ar (Note 1)	·	·	·	·	·	·
	LD A,(DE)	00 011 010					S	D		1	6	$(DE)_M$→Ar	·	·	·	·	·	·
	LD A,(mn)	00 111 010 ‹ n › ‹ m ›		S				D		3	12	$(mn)_M$→Ar	·	·	·	·	·	·
	LD I,A	11 101 101 01 000 111						S/D		2	6	Ar→Ir	·	·	·	·	·	·
	LD R,A	11 101 101 01 001 111						S/D		2	6	Ar→Rr	·	·	·	·	·	·
	LD (BC),A	00 000 010					D	S		1	7	Ar→$(BC)_M$	·	·	·	·	·	·
	LD (DE),A	00 010 010					D	S		1	7	Ar→$(DE)_M$	·	·	·	·	·	·
	LD (mn),A	00 110 010 ‹ n › ‹ m ›		D				S		3	13	Ar→$(mn)_M$	·	·	·	·	·	·
	LD g,g'	01 g g'				S/D				1	4	gr'→gr	·	·	·	·	·	·
	LD g,(HL)	01 g 110				D	S			1	6	$(HL)_M$→gr	·	·	·	·	·	·
	LD g,m	00 g 110 ‹ m ›	S			D				2	6	m→gr	·	·	·	·	·	·
	LD g,(IX+d)	11 011 101 01 g 110 ‹ d ›			S	D				3	14	$(IX+d)_M$→gr	·	·	·	·	·	·
	LD g,(IY+d)	11 111 101 01 g 110 ‹ d ›			S	D				3	14	$(IY+d)_M$→gr	·	·	·	·	·	·
	LD (HL),m	00 110 110 ‹ m ›	S				D			2	9	m→$(HL)_M$	·	·	·	·	·	·
	LD (IX+d),m	11 011 101 00 110 110 ‹ d › ‹ m ›	S		D					4	15	m→$(IX+d)_M$	·	·	·	·	·	·
	LD (IY+d),m	11 111 101 00 110 110 ‹ d › ‹ m ›	S		D					4	15	m→$(IY+d)_M$	·	·	·	·	·	·
	LD (HL),g	01 110 g				S	D			1	7	gr→$(HL)_M$	·	·	·	·	·	·
	LD (IX+d),g	11 011 101 01 110 g ‹ d ›			D	S				3	15	gr→$(IX+d)_M$	·	·	·	·	·	·
	LD (IY+d),g	11 111 101 01 110 g ‹ d ›			D	S				3	15	gr→$(IY+d)_M$	·	·	·	·	·	·

Note: 1 Interrupts are not sampled at the end of LD A, I or LD A, R.

Table 16 16-Bit Load

Operation Name	Mnemonics	Opcode	Addressing							Bytes	States	Operation	Flag					
													7	6	4	2	1	0
			IMMED	EXT	IND	REG	REGI	IMP	REL				S	Z	H	P/V	N	C
Load 16-Bit Data	LD ww,mn	00 ww0 001 ⟨ n ⟩ ⟨ m ⟩	S			D				3	9	mn→ww$_R$	·	·	·	·	·	·
	LD IX,mn	11 011 101 00 100 001 ⟨ n ⟩ ⟨ m ⟩	S					D		4	12	mn→IX$_R$	·	·	·	·	·	·
	LD IY,mn	11 111 101 00 100 001 ⟨ n ⟩ ⟨ m ⟩	S					D		4	12	mn→IY$_R$	·	·	·	·	·	·
	LD SP,HL	11 111 001						S/D		1	4	HL$_R$→SP$_R$	·	·	·	·	·	·
	LD SP,IX	11 011 101 11 111 001						S/D		2	7	IX$_R$→SP$_R$	·	·	·	·	·	·
	LD SP,IY	11 111 101 11 111 001						S/D		2	7	IY$_R$→SP$_R$	·	·	·	·	·	·
	LD ww,(mn)	11 101 101 01 ww1 011 ⟨ n ⟩ ⟨ m ⟩		S		D				4	18	(mn+1)$_M$→wwHr (mn)$_M$→wwLr	·	·	·	·	·	·
	LD HL,(mn)	00 101 010 ⟨ n ⟩ ⟨ m ⟩		S				D		3	15	(mn+1)$_M$→Hr (mn)$_M$→Lr	·	·	·	·	·	·
	LD IX,(mn)	11 011 101 00 101 010 ⟨ n ⟩ ⟨ m ⟩		S				D		4	18	(mn+1)$_M$→IXHr (mn)$_M$→IXLr	·	·	·	·	·	·
	LD IY,(mn)	11 111 101 00 101 010 ⟨ n ⟩ ⟨ m ⟩		S				D		4	18	(mn+1)$_M$→IYHr (mn)$_M$→IYLr	·	·	·	·	·	·
	LD (mn),ww	11 101 101 01 ww0 011 ⟨ n ⟩ ⟨ m ⟩		D		S				4	19	wwHr→(mn+1)$_M$ wwLr→(mn)$_M$	·	·	·	·	·	·
	LD (mn),HL	00 100 010 ⟨ n ⟩ ⟨ m ⟩		D				S		3	16	Hr→(mn+1)$_M$ Lr→(mn)$_M$	·	·	·	·	·	·
	LD (mn),IX	11 011 101 00 100 010 ⟨ n ⟩ ⟨ m ⟩		D				S		4	19	IXHr→(mn+1)$_M$ IXLr→(mn)$_M$	·	·	·	·	·	·
	LD (mn),IY	11 111 101 00 100 010 ⟨ n ⟩ ⟨ m ⟩		D				S		4	19	IYHr→(mn+1)$_M$ IYLr→(mn)$_M$	·	·	·	·	·	·

Table 17 Block Transfer

Operation Name	Mnemonics	Opcode	Addressing							Bytes	States	Operation	Flag					
													7	6	4	2	1	0
			IMMED	EXT	IND	REG	REGI	IMP	REL				S	Z	H	P/V	N	C
Block Transfer Search Data	CPD	11 101 101 10 101 001					S	S		2	12	$Ar-(HL)_M$ $BC_R-1\rightarrow BC_R$ $HL_R-1\rightarrow HL_R$	↕	↕[3]	↕	↕[2]	S	·
	CPDR	11 101 101 10 111 001					S	S		2	14 12	$BC_R\neq 0$ $Ar\neq(HL)_M$ $BC_R=0$ or $Ar=(HL)_M$ $Q\left\{\begin{array}{l}Ar-(HL\ )_M\\ BC_R-1\rightarrow BC_R\\ HL_R-1\rightarrow HL_R\end{array}\right.$ Repeat Q until $Ar=(HL)_M$ or $BC_R=0$	↕	↕[3]	↕	↕[2]	S	·
	CPI	11 101 101 10 100 001					S	S		2	12	$Ar-(HL)_M$ $BC_R-1\rightarrow BC_R$ $HL_R+1\rightarrow HL_R$	↕	↕[3]	↕	↕[2]	S	·
	CPIR	11 101 101 10 110 001					S	S		2	14 12	$BC_R\neq 0$ $Ar\neq(HL)_M$ $BC_R=0$ or $Ar=(HL)_M$ $Q\left\{\begin{array}{l}Ar-(HL\ )_M\\ BC_R-1\rightarrow BC_R\\ HL_R+1\rightarrow HL_R\end{array}\right.$ Repeat Q until $Ar=(HL)_M$ or $BC_R=0$	↕	↕[3]	↕	↕[2]	S	·
	LDD	11 101 101 10 101 000					S/D			2	12	$(HL)_M\rightarrow(DE)_M$ $BC_R-1\rightarrow BC_R$ $DE_R-1\rightarrow DE_R$ $HL_R-1\rightarrow HL_R$	·	·	R	↕[2]	R	·
	LDDk	11 101 101 10 111 000					S/D			2	14 $(BC_R\neq 0)$ 12 $(BC_R=0)$	$Q\left\{\begin{array}{l}(HL)_M\rightarrow(DE)_M\\ BC_R-1\rightarrow BC_R\\ DE_R-1\rightarrow DE_R\\ HL_R-1\rightarrow HL_R\end{array}\right.$ Repeat Q until $BC_R=0$	·	·	R	R	R	·
	LDI	11 101 101 10 100 000					S/D			2	12	$(HL)_M\rightarrow(DE)_M$ $BC_R-1\rightarrow BC_R$ $DE_R+1\rightarrow DE_R$ $HL_R+1\rightarrow HL_R$	·	·	R	↕[2]	R	·
	LDIR	11 101 101 10 110 000					S/D			2	14 $(BC_R\neq 0)$ 12 $(BC_R=0)$	$Q\left\{\begin{array}{l}(HL)_M\rightarrow(DE)_M\\ BC_R-1\rightarrow BC_R\\ DE_R+1\rightarrow DE_R\\ HL_R+1\rightarrow HL_R\end{array}\right.$ Repeat Q until $BC_R=0$	·	·	R	R	R	·

Note: 2 $P/V = 0.\ BC_R - 1 = 0$
$P/V = 1\cdot\ BC_R - 1 \neq 0$
3 $Z = 1\cdot\ Ar = (HL)_M$
$Z = 0\cdot\ Ar \neq (HL)_M$

Table 18 Stack and Exchange

Operation Name	Mnemonics	Opcode	Addressing							Bytes	States	Operation	Flag					
													7	6	4	2	1	0
			IMMED	EXT	IND	REG	REGI	IMP	REL				S	Z	H	P/V	N	C
PUSH	PUSH zz	11 zz0 101				S		D		1	11	$zzLr \rightarrow (SP-2)_M$ $zzHr \rightarrow (SP-1)_M$ $SP_R-2 \rightarrow SP_R$	·	·	·	·	·	·
	PUSH IX	11 011 101 11 100 101						S/D		2	14	$IXLr \rightarrow (SP-2)_M$ $IXHr \rightarrow (SP-1)_M$ $SP_R-2 \rightarrow SP_R$	·	·	·	·	·	·
	PUSH IY	11 111 101 11 100 101						S/D		2	14	$IYLr \rightarrow (SP-2)_M$ $IYHr \rightarrow (SP-1)_M$ $SP_R-2 \rightarrow SP_R$	·	·	·	·	·	·
POP	POP zz	11 zz0 001				D		S		1	9	$(SP+1)_M \rightarrow zzHr$ 4 $(SP)_M \rightarrow zzLr$ $SP_R+2 \rightarrow SP_R$	·	·	·	·	·	·
	POP IX	11 011 101 11 100 001						S/D		2	12	$(SP+1)_M \rightarrow IXHr$ $(SP)_M \rightarrow IXLr$ $SP_R+2 \rightarrow SP_R$	·	·	·	·	·	·
	POP IY	11 111 101 11 100 001						S/D		2	12	$(SP+1)_M \rightarrow IYHr$ $(SP)_M \rightarrow IYLr$ $SP_R+2 \rightarrow SP_R$	·	·	·	·	·	·
Exchange	EX AF,AF′	00 001 000						S/D		1	4	$AF_R \leftrightarrow AF_R'$	·	·	·	·	·	·
	EX DE,HL	11 101 011						S/D		1	3	$DE_R \leftrightarrow HL_R$	·	·	·	·	·	·
	EXX	11 011 001						S/D		1	3	$BC_R \leftrightarrow BC_R'$ $DE_R \leftrightarrow DE_R'$ $HL_R \leftrightarrow HL_R'$	·	·	·	·	·	·
	EX (SP),HL	11 100 011						S/D		1	16	$Hr \leftrightarrow (SP+1)_M$ $Lr \leftrightarrow (SP)_M$	·	·	·	·	·	·
	EX (SP),IX	11 011 101 11 100 011						S/D		2	19	$IXHr \leftrightarrow (SP+1)_M$ $IXLr \leftrightarrow (SP)_M$	·	·	·	·	·	·
	EX (SP),IY	11 111 101 11 100 011						S/D		2	19	$IYHr \leftrightarrow (SP+1)_M$ $IYLr \leftrightarrow (SP)_M$	·	·	·	·	·	·

Note 4 In the case of POP AF, Flag is written a current contents of the stack.

Program Control Instructions

Table 19 Program Control

Operation Name	Mnemonics	Opcode	Addressing							Bytes	States	Operation	Flag					
													7	6	4	2	1	0
			IMMED	EXT	IND	REG	REGI	IMP	REL				S	Z	H	P/V	N	C
Call	CALL mn	11 001 101 〈 n 〉 〈 m 〉		D						3	16	PCHr→$(SP-1)_M$ PCLr→$(SP-2)_M$ mn→PC_R $SP_R-2 \rightarrow SP_R$	·	·	·	·	·	·
	CALL f,mn	11 f 100 〈 n 〉 〈 m 〉		D						3	6 (f false) 16 (f ˙ true)	continue f is false CALL mn f is true	·	·	·	·	·	·
Jump	DJNZ j	00 010 000 〈j-2〉							D	2 2	9 (Br≠0) 7 (Br=0)	Br−1→Br continue . Br=0 $PC_R+j \rightarrow PC_R$. Br≠0	·	·	·	·	·	·
	JP f,mn	11 f 010 〈 n 〉 〈 m 〉		D						3 3	6 (f false) 9 (f true)	mn→PC_R f is true continue f is false	·	·	·	·	·	·
	JP mn	11 000 011 〈 n 〉 〈 m 〉		D						3	9	mn→PC_R	·	·	·	·	·	·
	JP (HL)	11 101 001					D			1	3	$HL_R \rightarrow PC_R$	·	·	·	·	·	·
	JP (IX)	11 011 101 11 101 001					D			2	6	$IX_R \rightarrow PC_R$	·	·	·	·	·	·
	JP (IY)	11 111 101 11 101 001					D			2	6	$IY_R \rightarrow PC_R$	·	·	·	·	·	·
	JR j	00 011 000 〈j-2〉							D	2	8	$PC_R+j \rightarrow PC_R$	·	·	·	·	·	·
	JR C,j	00 111 000 〈j-2〉							D	2 2	6 8	continue C=0 $PC_R+j \rightarrow PC_R$ C=1	·	·	·	·	·	·
	JR NC,j	00 110 000 〈j-2〉							D	2 2	6 8	continue . C=1 $PC_R+j \rightarrow PC_R$ C=0	·	·	·	·	·	·
	JR Z,j	00 101 000 〈j-2〉							D	2 2	6 8	continue Z=0 $PC_R+j \rightarrow PC_R$ Z=1	·	·	·	·	·	·
	JR NZ,j	00 100 000 〈j-2〉							D	2 2	6 8	continue ˙ Z=1 $PC_R+j \rightarrow PC_R$ Z=0	·	·	·	·	·	·
Return	RET	11 001 001						D		1	9	$(SP)_M \rightarrow$ PCLr $(SP+1)_M \rightarrow$ PCHr $SP_R+2 \rightarrow SP_R$	·	·	·	·	·	·
	RET f	11 f 000							D	1 1	5 (f false) 10 (f true)	continue f is false RET f is true	·	·	·	·	·	·
	RETI	11 101 101 01 001 101						D		2	22	$(SP)_M \rightarrow$ PCLr $(SP+1)_M \rightarrow$ PCHr $SP_R+2 \rightarrow SP_R$	·	·	·	·	·	·
	RETN	11 101 101 01 000 101						D		2	12	$(SP)_M \rightarrow$ PCLr $(SP+1)_M \rightarrow$ PCHr $SP_R+2 \rightarrow SP_R$ $IEF_2 \rightarrow IEF_1$	·	·	·	·	·	·
Restart	RST v	11 v 111						D		1	11	PCHr→$(SP-1)_M$ PCLr→$(SP-2)_M$ 0→PCHr v→PCLr $SP_R-2 \rightarrow SP_R$	·	·	·	·	·	·

I/O Instructions

Table 20 I/O

Operation Name	Mnemonics	Opcode	Addressing							Bytes	States	Operation	Flag					
													7	6	4	2	1	0
			IMMED	EXT	IND	REG	REGI	IMP	I/O				S	Z	H	P/V	N	C
Input	IN A,(m)	11 011 011 〈 m 〉						D	S	2	9	$(Am)_I \rightarrow Ar$ $m \rightarrow A_0 \sim A_7$ $Ar \rightarrow A_8 \sim A_{15}$	·	·	·	·	·	·
	IN g,(C)	11 101 101 01 g 000				D			S	2	9	$(BC)_I \rightarrow gr$ g=110 Only the flags will change $Cr \rightarrow A_0 \sim A_7$ $Br \rightarrow A_8 \sim A_{15}$	↕	↕	R	P	R	·
	IN0 g,(m) **	11 101 101 00 g 000 〈 m 〉				D			S	3	12	$(00m)_x \rightarrow gr$ g=110 Only the flags will change $m \rightarrow A_0 \sim A_7$ $00 \rightarrow A_8 \sim A_{15}$	↕	↕	R	P	R	·
	IND	11 101 101 10 101 010					D		S	2	12	$(BC)_I \rightarrow (HL)_M$ $HL_R - 1 \rightarrow HL_R$ $Br - 1 \rightarrow Br$ $Cr \rightarrow A_0 \sim A_7$ $Br \rightarrow A_8 \sim A_{15}$	X	[5] ↕	X	X	[6] ↕	X
	INDR	11 101 101 10 111 010					D		S	2	14(Br≠0) 12(Br=0)	$Q\begin{cases}(BC)_I \rightarrow (HL)_M \\ HL_R - 1 \rightarrow HL_R \\ Br - 1 \rightarrow Br\end{cases}$ Repeat Q until Br=0 $Cr \rightarrow A_0 \sim A_7$ $Br \rightarrow A_8 \sim A_{15}$	X	S	X	X	[6] ↕	X
	INI	11 101 101 10 100 010					D		S	2	12	$(BC)_I \rightarrow (HL)_M$ $HL_R + 1 \rightarrow HL_R$ $Br - 1 \rightarrow Br$ $Cr \rightarrow A_0 \sim A_7$ $Br \rightarrow A_8 \sim A_{15}$	X	[5] ↕	X	X	[6] ↕	X
	INIR	11 101 101 10 110 010					D		S	2	14(Br≠0) 12(Br=0)	$Q\begin{cases}(BC)_I \rightarrow (HL)_M \\ HL_R + 1 \rightarrow HL_R \\ Br - 1 \rightarrow Br\end{cases}$ Repeat Q until Br=0 $Cr \rightarrow A_0 \sim A_7$ $Br \rightarrow A_8 \sim A_{15}$	X	S	X	X	[6] ↕	X

Note 5 Z = 1. Br − 1 = 0
Z = 0· Br − 1 ≠ 0
6 N = 1 MSB of Data = 1
N = 0 MSB of Data = 0

(continued)

Table 20 I/O (cont)

Operation Name	Mnemonics	Opcode	Addressing							Bytes	States	Operation	Flag					
													7	6	4	2	1	0
			IMMED	EXT	IND	REG	REGI	IMP	I/O				S	Z	H	P/V	N	C
Output	OUT (m),A	11 010 011 ⟨ m ⟩						S	D	2	10	$Ar \to (Am)_1$ $m \to A_0 \sim A_7$ $Ar \to A_8 \sim A_{15}$	·	·	·	·	·	·
	OUT (C),g	11 101 101 01 g 001				S			D	2	10	$gr \to (BC)_1$ $Cr \to A_0 \sim A_7$ $Br \to A_8 \sim A_{15}$	·	·	·	·	·	·
	OUT0 (m),g **	11 101 101 00 g 001 ⟨ m ⟩				S			D	3	13	$gr \to (00m)_1$ $m \to A_0 \sim A_7$ $00 \to A_8 \sim A_{15}$	·	·	·	·	·	·
	OTDM **	11 101 101 10 001 011					S		D	2	14	$(HL)_M \to (00C)_1$ $HL_R-1 \to HL_R$ $Cr-1 \to Cr$ $Br-1 \to Br$ $Cr \to A_0 \sim A_7$ $00 \to A_8 \sim A_{15}$	↕	5 ↕	↕	P	6 ↕	↕
	OTDMR **	11 101 101 10 011 011					S		D	2	16(Br≠0) 14(Br=0)	$Q\left\{\begin{array}{l}(HL)_M \to (00C)_1\\ HL_R-1 \to HL_R\\ Cr-1 \to Cr\\ Br-1 \to Br\end{array}\right.$ Repeat Q until Br=0 $Cr \to A_0 \sim A_7$ $00 \to A_8 \sim A_{15}$	R	S	R	S	6 ↕	R
	OTDR	11 101 101 10 111 011					S		D	2	14(Br≠0) 12(Br=0)	$Q\left\{\begin{array}{l}(HL)_M \to (BC)_1\\ HL_R-1 \to HL_R\\ Br-1 \to Br\end{array}\right.$ Repeat Q until Br=0 $Cr \to A_0 \sim A_7$ $Br \to A_8 \sim A_{15}$	X	S	X	X	6 ↕	X
	OUTI	11 101 101 10 100 011					S		D	2	12	$(HL)_M \to (BC)_1$ $HL_R+1 \to HL_R$ $Br-1 \to Br$ $Cr \to A_0 \sim A_7$ $Br \to A_8 \sim A_{15}$	X	5 ↕	X	X	6 ↕	X
	OTIR	11 101 101 10 110 011					S		D	2	14(Br≠0) 12(Br=0)	$Q\left\{\begin{array}{l}(HL)_M \to (BC)_1\\ HL_R+1 \to HL_R\\ Br-1 \to Br\end{array}\right.$ Repeat Q until Br=0 $Cr \to A_0 \sim A_7$ $Br \to A_8 \sim A_{15}$	X	S	X	X	6 ↕	X
	TSTIO m **	11 101 101 01 110 100 ⟨ m ⟩	S						S	3	12	$(00C)_1 \cdot m$ $Cr \to A_0 \sim A_7$ $00 \to A_8 \sim A_{15}$	↕	↕	S	P	R	R
	OTIM **	11 101 101 10 000 011					S		D	2	14	$(HL)_M \to (00C)_1$ $HL_R+1 \to HL_R$ $Cr+1 \to Cr$ $Br-1 \to Br$ $Cr \to A_0 \sim A_7$ $00 \to A_8 \sim A_{15}$	↕	5 ↕	↕	P	6 ↕	↕
	OTIMR **	11 101 101 10 010 011					S		D	2	16(Br≠0) 14(Br=0)	$Q\left\{\begin{array}{l}(HL)_M \to (00C)_1\\ HL_R+1 \to HL_R\\ Cr+1 \to Cr\\ Br-1 \to Br\end{array}\right.$ Repeat Q until Br=0 $Cr \to A_0 \sim A_7$ $00 \to A_8 \sim A_{15}$	R	S	R	S	6 ↕	R
	OUTD	11 101 101 10 101 011					S		D	2	12	$(HL)_M \to (BC)_1$ $HL_R-1 \to HL_R$ $Br-1 \to Br$ $Cr \to A_0 \sim A_7$ $Br \to A_8 \sim A_{15}$	X	5 ↕	X	X	6 ↕	X

Note: 5 $Z = 1.\ Br-1 = 0$
$Z = 0.\ Br-1 \neq 0$
6 $N = 1$: MSB of Data = 1
$N = 0$: MSB of Data = 0

Special Control Instructions

Table 21 Special Control

			Addressing										Flag					
													7	6	4	2	1	0
Operation Name	Mnemonics	Opcode	IMMED	EXT	IND	REG	REGI	IMP	REL	Bytes	States	Operation	S	Z	H	P/V	N	C
Special Function	DAA	00 100 111						S/D		1	4	Decimal Adjust Accumulator	↕	↕	↕	P	·	↕
Carry Control	CCF	00 111 111								1	3	$\overline{C}$→C	·	·	R	·	R	↕
	SCF	00 110 111								1	3	1→C	·	·	R	·	R	S
CPU Control	DI	11 110 011								1	3	0→IEF$_1$, 0→IEF$_2$ 7	·	·	·	·	·	·
	EI	11 111 011								1	3	1→IEF$_1$, 1→IEF$_2$ 7	·	·	·	·	·	·
	HALT	01 110 110								1	3	CPU halted	·	·	·	·	·	·
	IM 0	11 101 101 01 000 110								2	6	Interrupt mode 0	·	·	·	·	·	·
	IM 1	11 101 101 01 010 110								2	6	Interrupt mode 1	·	·	·	·	·	·
	IM 2	11 101 101 01 011 110								2	6	Interrupt mode 2	·	·	·	·	·	·
	NOP	00 000 000								1	3	No operation	·	·	·	·	·	·
	SLP **	11 101 101 01 110 110								2	8	Sleep	·	·	·	·	·	·

Note 7 Interrupts are not sampled at the end of DI or EI.

■ INSTRUCTION SUMMARY IN ALPHABETICAL ORDER

MNEMONICS	Bytes	Machine Cycles	States
ADC A,m	2	2	6
ADC A,g	1	2	4
ADC A, (HL)	1	2	6
ADC A, (IX+d)	3	6	14
ADC A, (IY+d)	3	6	14
ADD A,m	2	2	6
ADD A,g	1	2	4
ADD A, (HL)	1	2	6
ADD A, (IX+d)	3	6	14
ADD A, (IY+d)	3	6	14
ADC HL,ww	2	6	10
ADD HL,ww	1	5	7
ADD IX,xx	2	6	10
ADD IY,yy	2	6	10
AND m	2	2	6
AND g	1	2	4
AND (HL)	1	2	6
AND (IX+d)	3	6	14
AND (IY+d)	3	6	14
BIT b, (HL)	2	3	9
BIT b, (IX+d)	4	5	15
BIT b, (IY+d)	4	5	15
BIT b,g	2	2	6
CALL f,mn	3	2	6 (If condition is false)
	3	6	16 (If condition is true)

(continued)

Note ** : New instructions added to Z80

MNEMONICS	Bytes	Machine Cycles	States
CALL mn	3	6	16
CCF	1	1	3
CPD	2	6	12
CPDR	2	8	14
			(If $BC_R \neq 0$ and $Ar \neq (HL)_M$)
	2	6	12
			(If $BC_R = 0$ or $Ar = (HL)_M$)
CP (HL)	1	2	6
CPI	2	6	12
CPIR	2	8	14
			(If $BC_R \neq 0$ and $Ar \neq (HL)_M$)
	2	6	12
			(If $BC_R = 0$ or $Ar = (HL)_M$)
CP (IX+d)	3	6	14
CP (IY+d)	3	6	14
CPL	1	1	3
CP m	2	2	6
CP g	1	2	4
DAA	1	2	4
DEC (HL)	1	4	10
DEC IX	2	3	7
DEC IY	2	3	7
DEC (IX+d)	3	8	18
DEC (IY+d)	3	8	18
DEC g	1	2	4
DEC ww	1	2	4
DI	1	1	3

(continued)

MNEMONICS	Bytes	Machine Cycles	States
DJNZ j	2	5	9 (If Br≠0)
	2	3	7 (If Br=0)
EI	1	1	3
EX AF,AF'	1	2	4
EX DE,HL	1	1	3
EX (SP),HL	1	6	16
EX (SP),IX	2	7	19
EX (SP),IY	2	7	19
EXX	1	1	3
HALT	1	1	3
IM 0	2	2	6
IM 1	2	2	6
IM 2	2	2	6
INC g	1	2	4
INC (HL)	1	4	10
INC (IX+d)	3	8	18
INC (IY+d)	3	8	18
INC ww	1	2	4
INC IX	2	3	7
INC IY	2	3	7
IN A,(m)	2	3	9
IN g,(C)	2	3	9
INI	2	4	12
INIR	2	6	14 (If Br≠0)
	2	4	12 (If Br=0)
IND	2	4	12
INDR	2	6	14 (If Br≠0)

(continued)

MNEMONICS	Bytes	Machine Cycles	States
INDR	2	4	12 (If Br=0)
INO g,(m)**	3	4	12
JP f,mn	3	2	6
			(If f is false)
	3	3	9
			(If f is true)
JP (HL)	1	1	3
JP (IX)	2	2	6
JP (IY)	2	2	6
JP mn	3	3	9
JR j	2	4	8
JR C,j	2	2	6
			(If condition is false)
	2	4	8
			(If condition is true)
JR NC,j	2	2	6
			(If condition is false)
	2	4	8
			(If condition is true)
JR Z,j	2	2	6
			(If condition is false)
	2	4	8
			(If condition is true)
JR NZ,j	2	2	6
			(If condition is false)
	2	4	8
			(If condition is true)

(continued)

MNEMONICS	Bytes	Machine Cycles	States
LD A, (BC)	1	2	6
LD A, (DE)	1	2	6
LD A,I	2	2	6
LD A, (mn)	3	4	12
LD A,R	2	2	6
LD (BC),A	1	3	7
LDD	2	4	12
LD (DE),A	1	3	7
LD ww,mn	3	3	9
LD ww,(mn)	4	6	18
LDDR	2	6	14 (If $BC_R \neq 0$)
	2	4	12 (If $BC_R = 0$)
LD (HL),m	2	3	9
LD HL,(mn)	3	5	15
LD (HL),g	1	3	7
LDI	2	4	12
LD I,A	2	2	6
LDIR	2	6	14 (If $BC_R \neq 0$)
	2	4	12 (If $BC_R = 0$)
LD IX,mn	4	4	12
LD IX,(mn)	4	6	18
LD (IX+d),m	4	5	15
LD (IX+d),g	3	7	15
LD IY,mn	4	4	12
LD IY,(mn)	4	6	18
LD (IY+d),m	4	5	15
LD (IY+d),g	3	7	15

(continued)

MNEMONICS	Bytes	Machine Cycles	States
LD (mn),A	3	5	13
LD (mn),ww	4	7	19
LD (mn),HL	3	6	16
LD (mn),IX	4	7	19
LD (mn),IY	4	7	19
LD R,A	2	2	6
LD g,(HL)	1	2	6
LD g,(IX+d)	3	6	14
LD g,(IY+d)	3	6	14
LD g,m	2	2	6
LD g,g'	1	2	4
LD SP,HL	1	2	4
LD SP,IX	2	3	7
LD SP,IY	2	3	7
MLT ww**	2	13	17
NEG	2	2	6
NOP	1	1	3
OR (HL)	1	2	6
OR (IX+d)	3	6	14
OR (IY+d)	3	6	14
OR m	2	2	6
OR g	1	2	4
OTDM**	2	6	14
OTDMR**	2	8	16 (If Br≠0)
	2	6	14 (If Br=0)
OTDR	2	6	14 (If Br≠0)
	2	4	12 (If Br=0)

(continued)

MNEMONICS	Bytes	Machine Cycles	States
OTIM**	2	6	14
OTIMR**	2	8	16 (If Br≠0)
	2	6	14 (If Br=0)
OTIR	2	6	14 (If Br≠0)
	2	4	12 (If Br=0)
OUTD	2	4	12
OUTI	2	4	12
OUT (m),A	2	4	10
OUT (C),g	2	4	10
OUT0 (m),g**	3	5	13
POP IX	2	4	12
POP IY	2	4	12
POP zz	1	3	9
PUSH IX	2	6	14
PUSH IY	2	6	14
PUSH zz	1	5	11
RES b,(HL)	2	5	13
RES b,(IX+d)	4	7	19
RES b,(IY+d)	4	7	19
RES b,g	2	3	7
RET	1	3	9
RET f	1	3	5
			(If condition is false)
	1	4	10
			(If condition is true)
RETI	2	10	22
RETN	2	4	12

(continued)

MNEMONICS	Bytes	Machine Cycles	States
RLA	1	1	3
RLCA	1	1	3
RLC (HL)	2	5	13
RLC (IX+d)	4	7	19
RLC (IY+d)	4	7	19
RLC g	2	3	7
RLD	2	8	16
RL (HL)	2	5	13
RL (IX+d)	4	7	19
RL (IY+d)	4	7	19
RL g	2	3	7
RRA	1	1	3
RRCA	1	1	3
RRC (HL)	2	5	13
RRC (IX+d)	4	7	19
RRC (IY+d)	4	7	19
RRC g	2	3	7
RRD	2	8	16
RR (HL)	2	5	13
RR (IX+d)	4	7	19
RR (IY+d)	4	7	19
RR g	2	3	7
RST v	1	5	11
SBC A,(HL)	1	2	6
SBC A,(IX+d)	3	6	14
SBC A,(IY+d)	3	6	14
SBC A,m	2	2	6

(continued)

MNEMONICS	Bytes	Machine Cycles	States
SBC A,g	1	2	4
SBC HL,ww	2	6	10
SCF	1	1	3
SET b,(HL)	2	5	13
SET b,(IX+d)	4	7	19
SET b,(IY+d)	4	7	19
SET b,g	2	3	7
SLA (HL)	2	5	13
SLA (IX+d)	4	7	19
SLA (IY+d)	4	7	19
SLA g	2	3	7
SLP**	2	2	8
SRA (HL)	2	5	13
SRA (IX+d)	4	7	19
SRA (IY+d)	4	7	19
SRA g	2	3	7
SRL (HL)	2	5	13
SRL (IX+d)	4	7	19
SRL (IY+d)	4	7	19
SRL g	2	3	7
SUB (HL)	1	2	6
SUB (IX+d)	3	6	14
SUB (IY+d)	3	6	14
SUB m	2	2	6
SUB g	1	2	4
**TSTIO m	3	4	12
**TST g	2	3	7

(continued)

MNEMONICS	Bytes	Machine Cycles	States
TST m**	3	3	9
TST (HL)**	2	4	10
XOR (HL)	1	2	6
XOR (IX+d)	3	6	14
XOR (IY+d)	3	6	14
XOR m	2	2	6
XOR g	1	2	4

■ OPCODE MAP

Table 22 First Opcode Map

Instruction format: XX

				ww(Lo = All)												Lo = 0 – 7				
				BC	DE	HL	SP									BC	DE	HL	AF	zz
				g (Lo = 0 – 7)												NZ	NC	PO	P	f
				B	D	H	(HL)	B	D	H	(HL)					00H	10H	20H	30H	v
			HI	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111	
		LO		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
S (Hi = All)	B	0000	0	NOP	DJNZ j	JR NZ, j	JR NC, j	LD g, s			(Note 1)	ADD A, s	SUB s	AND s	OR s	RET f				0
	C	0001	1	LD ww, mn												POP zz				1
	D	0010	2	LD (ww), A		LD (mn), HL	LD (mn), A									JP f, mn				2
	E	0011	3	INC ww												JP mn	OUT (m), A	EX (SP), HL	DI	3
	H	0100	4	INC g			(Note 1)									CALL f, mn				4
	L	0101	5	DEC g			(Note 1)									PUSH zz				5
	(HL)	0110	6	LD g, m			(Note 1)	(Note 2)			HALT	(Note 2)	(Note 2)	(Note 2)	(Note 2)	ADD A, m	SUB m	AND m	OR m	6
	A	0111	7	RLCA	RLA	DAA	SCF									RST v				7
	B	1000	8	EXAF, AF'	JR j	JR Z, j	JR C, j	LD g, s				ADC A, s	SBC A, s	XOR s	CP s	RET f				8
	C	1001	9	ADD HL, ww												RET	EXX	JP (HL)	LD SP, HL	9
	D	1010	A	LD A, (ww)		LD HL, (mn)	LD A, (mn)									JP f, mn				A
	E	1011	B	DEC ww												Table 2	IN A, (m)	EX DE, HL	EI	B
	H	1100	C	INC g												CALL f, mn				C
	L	1101	D	DEC g												CALL mn	(Note 3)	Table 3	(Note 3)	D
	(HL)	1110	E	LD g, m				(Note 2)				(Note 2)	(Note 2)	(Note 2)	(Note 2)	ADC A, m	SBC A, m	XOR m	CP m	E
	A	1111	F	RRCA	RRA	CPL	CCF									RST v				F
				0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
				C	E	L	A	C	E	L	A					Z	C	PE	M	f
				g (Lo = 8 – F)												08H	18H	28H	38H	v
																Lo = 8 – F				

Notes: 1. (HL) replaces g.

2. (HL) replaces s.

3. If DDH is added as first opcode for the instructions which have HL or (HL) as an operand in table 1, the instructions are executed replacing HL with IX and (HL) with (IX + d).

```
ex:      22H: LD (mn), HL
     DDH 22H: LD (mn), IX
```

If FDH is added as first opcode for the instructions which have HL or (HL) as an operand in table 1, the instructions are executed replacing HL with IY and (HL) with (IY + d).

```
ex:      34H: INC (HL)
     FDH 34H: INC (IY + d)
```

However, JP (HL) and EX DE, HL are exceptions. Note the followings:
If DDH is added as first opcode for JP (HL), (IX) replaces (HL) as operand and JP (IX) is executed.
If FDH is added as first opcode for JP (HL), (IY) replaces (HL) as operand and JP (IY) is executed.
Even if DDH or FDH is added as first opcode for EX DE, HL, HL is not replaced and the instruction is regarded as illegal.

Table 23 Second Opcode Map
Instruction format: CB XX

								b (Lo = 0 – 7)												
								0	2	4	6	0	2	4	6	0	2	4	6	
			HI	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111	
		LO		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
g (Hi = All)	B	0000	0	RLC g	RL g	SLA g		BIT b,g				RES b,g				SET b,g				0
	C	0001	1																	1
	D	0010	2																	2
	E	0011	3																	3
	H	0100	4																	4
	L	0101	5																	5
	(HL)	0110	6	(Note 1)	(Note 1)	(Note 1)		(Note 1)				(Note 1)				(Note 1)				6
	A	0111	7																	7
	B	1000	8	RRC g	RR g	SRA g	SRL g	BIT b,g				RES b,g				SET b,g				8
	C	1001	9																	9
	D	1010	A																	A
	E	1011	B																	B
	H	1100	C																	C
	L	1101	D																	D
	(HL)	1110	E	(Note 1)	(Note 1)	(Note 1)	(Note 1)	(Note 1)				(Note 1)				(Note 1)				E
	A	1111	F																	F
				0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
								1	3	5	7	1	3	5	7	1	3	5	7	
								b (Lo = 8 – nF)												

Note 1 If DDH is added as first opcode for the instructions which have (HL) as operand in table 2, the instructions are executed replacing (HL) with (IX + d)

If FDH is added as first opcode for the instructions which have (HL) as operand in table 2, the instructions are executed replacing (HL) with (IY + d)

Table 24 Second Opcode Map
Instruction format: ED XX

						ww (Lo = All)												
						BC	DE	HL	SP									
		g (Lo = 0 – 7)																
		B	D	H		B	D	H										
	Hi	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111	
Lo		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0000	0	IN0 g, (m)				IN g, (C)						LDI	LDIR					0
0001	1	OUT0 (m),g				OUT (C),g						CPI	CPIR					1
0010	2					SBC HL, ww						INI	INIR					2
0011	3					LD (mn), ww				OTIM	OTIMR	OUTI	OTIR					3
0100	4	TST g			TST (HL)	NEG		TST m	TSTIO m									4
0101	5					RETN												5
0110	6					IM 0	IM 1		SLP									6
0111	7					LD I,A	LD A,I	RRD										7
1000	8	IN0 g, (m)				IN g, (C)						LDD	LDDR					8
1001	9	OUT0 (m),g				OUT (C),g						CPD	CPDR					9
1010	A					ADC HL,ww						IND	INDR					A
1011	B					LD ww, (mn)				OTDM	OTDMR	OUTD	OTDR					B
1100	C	TST g				MLT ww												C
1101	D					RETI												D
1110	E						IM 2											E
1111	F					LD R,A	LD A,R	RLD										F
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
		C	E	L	A	C	E	L	A									
		g (Lo = 8 – F)																

■ BUS AND CONTROL SIGNAL CONDITION IN EACH MACHINE CYCLE

Instruction	Machine Cycle	States	Address	Data	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{LIR}$	$\overline{HALT}$	ST
ADD HL,ww	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC_2– MC_5	TiTiTiTi	•	Z	1	1	1	1	1	1	1
ADD IX,xx ADD IY,yy	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd opcode Address	2nd opcode	0	1	0	1	0	1	1
	MC_3– MC_6	TiTiTiTi	•	Z	1	1	1	1	1	1	1
ADC HL,ww SBC HL,ww	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd opcode Address	2nd opcode	0	1	0	1	0	1	1
	MC_3– MC_6	TiTiTiTi	•	Z	1	1	1	1	1	1	1
ADD A,g ADC A,g SUB g SBC A,g AND g OR g XOR g CP g	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC_2	Ti	•	Z	1	1	1	1	1	1	1
ADD A,m ADC A,m SUB m SBC A,m AND m OR m XOR m CP m	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	1st operand Address	m	0	1	0	1	1	1	1
ADD A, (HL) ADC A, (HL) SUB (HL) SBC A, (HL) AND (HL) OR (HL) XOR (HL) CP (HL)	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	HL	Data	0	1	0	1	1	1	1
ADD A, (IX+d) ADD A, (IY+d) ADC A, (IX+d) ADC A, (IY+d) SUB (IX+d) SUB (IY+d) SBC A, (IX+d)	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd opcode Address	2nd opcode	0	1	0	1	0	1	1

(continued)

Note: • (Address): Invalid
Z (Data): High impedance.
**: New instructions added to Z80

Instruction	Machine Cycle	States	Address	Data	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{LIR}$	$\overline{HALT}$	ST
SBC A, (IY+d) AND (IX+d) AND (IY+d) OR (IX+d) OR (IY+d) XOR (IX+d) XOR (IY+d) CP (IX+d) CP (IY+d)	MC_3	$T_1T_2T_3$	1st operand Address	d	0	1	0	1	1	1	1
	MC_4– MC_5	T_iT_i	*	Z	1	1	1	1	1	1	1
	MC_6	$T_1T_2T_3$	IX+d IY+d	Data	0	1	0	1	1	1	1
BIT b,g	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd opcode Address	2nd opcode	0	1	0	1	0	1	1
BIT b, (HL)	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd opcode Address	2nd opcode	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	HL	Data	0	1	0	1	1	1	1
BIT b, (IX+d) BIT b, (IY+d)	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd opcode Address	2nd opcode	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	1st operand Address	d	0	1	0	1	1	1	1
	MC_4	$T_1T_2T_3$	3rd opcode Address	3rd opcode	0	1	0	1	0	1	1
	MC_5	$T_1T_2T_3$	IX+d IY+d	Data	0	1	0	1	1	1	1
CALL mn	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	1st operand Address	n	0	1	0	1	1	1	1
	MC_3	$T_1T_2T_3$	2nd operand Address	m	0	1	0	1	1	1	1
	MC_4	T_i	*	Z	1	1	1	1	1	1	1
	MC_5	$T_1T_2T_3$	SP−1	PCH	1	0	0	1	1	1	1
	MC_6	$T_1T_2T_3$	SP−2	PCL	1	0	0	1	1	1	1
CALL f,mn (If condition is false)	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	1st operand Address	n	0	1	0	1	1	1	1

(continued)

Instruction	Machines										
	Cycle	States	Address	Data	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{LIR}$	$\overline{HALT}$	ST
CALL f,mn (If condition is true)	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	1st operand Address	n	0	1	0	1	1	1	1
	MC_3	$T_1T_2T_3$	2nd operand Address	m	0	1	0	1	1	1	1
	MC_4	Ti	*	Z	1	1	1	1	1	1	1
	MC_5	$T_1T_2T_3$	SP−1	PCH	1	0	0	1	1	1	1
	MC_6	$T_1T_2T_3$	SP−2	PCL	1	0	0	1	1	1	1
CCF	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
CPI CPD	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd opcode Address	2nd opcode	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	HL	Data	0	1	0	1	1	1	1
	MC_4– MC_6	$T_iT_iT_i$	*	Z	1	1	1	1	1	1	1
CPIR CPDR (If $BC_R \neq 0$ and $Ar \neq (HL)_M$)	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd opcode Address	2nd opcode	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	HL	Data	0	1	0	1	1	1	1
	MC_4– MC_8	$T_iT_iT_iT_iT_i$	*	Z	1	1	1	1	1	1	1
CPIR CPDR (If $BC_R = 0$ or $Ar = (HL)_M$)	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd opcode Address	2nd opcode	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	HL	Data	0	1	0	1	1	1	1
	MC_4– MC_6	TiTiTi	*	Z	1	1	1	1	1	1	1
CPL	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
DAA	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC_2	T_i	*	Z	1	1	1	1	1	1	1
DI (Note 1)	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0

(continued)

Note· 1 Interrupt request is not sampled

Instruction	Machine Cycle	States	Address	Data	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{LIR}$	$\overline{HALT}$	ST
DJNZ j (If Br≠0)	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC_2	T_i (Note 2)	*	Z	1	1	1	1	1	1	1
	MC_3	$T_1T_2T_3$	1st operand Address	j-2	0	1	0	1	1	1	1
	MC_4– MC_5	T_iT_i	*	Z	1	1	1	1	1	1	1
DJNZ j (If Br=0)	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC_2	T_i (Note 1)	*	Z	1	1	1	1	1	1	1
	MC_3	$T_1T_2T_3$	1st operand Address	j-2	0	1	0	1	1	1	1
EI (Note 3)	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
EX DE, HL EXX	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
EX AF, AF′	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC_2	T_i	*	Z	1	1	1	1	1	1	1
EX (SP), HL	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	SP	Data	0	1	0	1	1	1	1
	MC_3	$T_1T_2T_3$	SP+1	Data	0	1	0	1	1	1	1
	MC_4	T_i	*	Z	1	1	1	1	1	1	1
	MC_5	$T_1T_2T_3$	SP+1	H	1	0	0	1	1	1	1
	MC_6	$T_1T_2T_3$	SP	L	1	0	0	1	1	1	1
EX (SP),IX EX (SP),IY	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd opcode Address	2nd opcode	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	SP	Data	0	1	0	1	1	1	1
	MC_4	$T_1T_2T_3$	SP+1	Data	0	1	0	1	1	1	1
	MC_5	T_i	*	Z	1	1	1	1	1	1	1

(continued)

Note 2 DMA, refresh, or bus release cannot be executed after this state (Request is ignored)

3 Interrupt request is not sampled

Instruction	Machine Cycle	States	Address	Data	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{LIR}$	$\overline{HALT}$	ST
EX (SP), IX EX (SP), IY	MC$_6$	$T_1T_2T_3$	SP+ 1	IXH IYH	1	0	0	1	1	1	1
	MC$_7$	$T_1T_2T_3$	SP	IXL IYL	1	0	0	1	1	1	1
HALT	MC$_1$	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	——	——	Next opcode Address	Next opcode	0	1	0	1	0	0	0
IM 0 IM 1	MC$_1$	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
IM 2	MC$_2$	$T_1T_2T_3$	2nd opcode Address	2nd opcode	0	1	0	1	0	1	1
INC g DEC g	MC$_1$	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC$_2$	Ti	•	Z	1	1	1	1	1	1	1
INC (HL) DEC (HL)	MC$_1$	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC$_2$	$T_1T_2T_3$	HL	Data	0	1	0	1	1	1	1
	MC$_3$	Ti	•	Z	1	1	1	1	1	1	1
	MC$_4$	$T_1T_2T_3$	HL	Data	1	0	0	1	1	1	1
INC (IX+ d) INC (IY+ d)	MC$_1$	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
DEC (IX+ d) DEC (IY+ d)	MC$_2$	$T_1T_2T_3$	2nd opcode Address	2nd opcode	0	1	0	1	0	1	1
	MC$_3$	$T_1T_2T_3$	1st operand Address	d	0	1	0	1	1	1	1
	MC$_4$– MC$_5$	TiTi	•	Z	1	1	1	1	1	1	1
	MC$_6$	$T_1T_2T_3$	IX+ d IY+ d	Data	0	1	0	1	1	1	1
	MC$_7$	Ti	•	Z	1	1	1	1	1	1	1
	MC$_8$	$T_1T_2T_3$	IX+ d IY+ d	Data	1	0	0	1	1	1	1
INC ww DEC ww	MC$_1$	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC$_2$	Ti	•	Z	1	1	1	1	1	1	1
INC IX INC IY	MC$_1$	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
DEC IX DEC IY	MC$_2$	$T_1T_2T_3$	2nd opcode Address	2nd opcode	0	1	0	1	0	1	1
	MC$_3$	Ti	•	Z	1	1	1	1	1	1	1

(continued)

Instruction	Machine Cycle	States	Address	Data	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{LIR}$	$\overline{HALT}$	ST
IN A,(m)	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	1st operand Address	m	0	1	0	1	1	1	1
	MC_3	$T_1T_2T_3$	m to A_0—A_7 A to A_8—A_{15}	Data	0	1	1	0	1	1	1
IN g,(C)	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd opcode Address	2nd opcode	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	BC	Data	0	1	1	0	1	1	1
INO g,(m)**	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd opcode Address	2nd opcode	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	1st operand Address	m	0	1	0	1	1	1	1
	MC_4	$T_1T_2T_3$	m to A_0—A_7 00H to A_8—A_{15}	Data	0	1	1	0	1	1	1
INI IND	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd opcode Address	2nd opcode	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	BC	Data	0	1	1	0	1	1	1
	MC_4	$T_1T_2T_3$	HL	Data	1	0	0	1	1	1	1
INIR INDR (If Br≠0)	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd opcode Address	2nd opcode	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	BC	Data	0	1	1	0	1	1	1
	MC_4	$T_1T_2T_3$	HL	Data	1	0	0	1	1	1	1
	MC_5—MC_6	T_iT_i	*	Z	1	1	1	1	1	1	1
INIR INDR (If Br=0)	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd opcode Address	2nd opcode	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	BC	Data	0	1	1	0	1	1	1
	MC_4	$T_1T_2T_3$	HL	Data	1	0	0	1	1	1	1

(continued)

Instruction	Machine Cycle	States	Address	Data	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{LIR}$	$\overline{HALT}$	ST
JP mn	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	1st operand Address	n	0	1	0	1	1	1	1
	MC_3	$T_1T_2T_3$	2nd operand Address	m	0	1	0	1	1	1	1
JP f,mn (If f is false)	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	1st operand Address	n	0	1	0	1	1	1	1
JP f,mn (If f is true)	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	1st operand Address	n	0	1	0	1	1	1	1
	MC_3	$T_1T_2T_3$	2nd operand Address	m	0	1	0	1	1	1	1
JP (HL)	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
JP (IX) JP (IY)	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd opcode Address	2nd opcode	0	1	0	1	0	1	1
JR j	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	1st operand Address	j-2	0	1	0	1	1	1	1
	MC_3— MC_4	T_iT_i	*	Z	1	1	1	1	1	1	1
JR C,j JR NC,j JR Z,j JR NZ,j (If condition is false)	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	1st operand Address	j-2	0	1	0	1	1	1	1
JR C,j JR NC,j JR Z,j JR NZ,j (If condition is true)	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	1st operand Address	j-2	0	1	0	1	1	1	1
	MC_3— MC_4	T_iT_i	*	Z	1	1	1	1	1	1	1
LD g,g'	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC_2	T_i	*	Z	1	1	1	1	1	1	1
LD g,m	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	1st operand Address	m	0	1	0	1	1	1	1

(continued)

Instruction	Machine Cycle	States	Address	Data	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{LIR}$	$\overline{HALT}$	ST
LD g, (HL)	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	HL	Data	0	1	0	1	1	1	1
LD g, (IX+d) LD g, (IY+d)	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd opcode Address	2nd opcode	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	1st operand Address	d	0	1	0	1	1	1	1
	MC_4– MC_5	TiTi	•	Z	1	1	1	1	1	1	1
	MC_6	$T_1T_2T_3$	IX+d IY+d	Data	0	1	0	1	1	1	1
LD (HL),g	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC_2	Ti	•	Z	1	1	1	1	1	1	1
	MC_3	$T_1T_2T_3$	HL	g	1	0	0	1	1	1	1
LD (IX+d),g LD (IY+d),g	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd opcode Address	2nd opcode	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	1st operand Address	d	0	1	0	1	1	1	1
	MC_4– MC_6	TiTiTi	•	Z	1	1	1	1	1	1	1
	MC_7	$T_1T_2T_3$	IX+d IY+d	g	1	0	0	1	1	1	1
LD (HL),m	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	1st operand Address	m	0	1	0	1	1	1	1
	MC_3	$T_1T_2T_3$	HL	Data	1	0	0	1	1	1	1
LD (IX+d),m LD (IY+d),m	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd opcode Address	2nd opcode	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	1st operand Address	d	0	1	0	1	1	1	1
	MC_4	$T_1T_2T_3$	2nd operand Address	m	0	1	0	1	1	1	1
	MC_5	$T_1T_2T_3$	IX+d IY+d	Data	1	0	0	1	1	1	1
LD A, (BC) LD A, (DE)	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0

(continued)

Instruction	Machine Cycle	States	Address	Data	$\overline{\text{RD}}$	$\overline{\text{WR}}$	$\overline{\text{ME}}$	$\overline{\text{IOE}}$	$\overline{\text{LIR}}$	$\overline{\text{HALT}}$	ST
LD A, (BC) LD A, (DE)	MC$_2$	$T_1T_2T_3$	BC DE	Data	0	1	0	1	1	1	1
LD A,(mn)	MC$_1$	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC$_2$	$T_1T_2T_3$	1st operand Address	n	0	1	0	1	1	1	1
	MC$_3$	$T_1T_2T_3$	2nd operand Address	m	0	1	0	1	1	1	1
	MC$_4$	$T_1T_2T_3$	mn	Data	0	1	0	1	1	1	1
LD (BC),A LD (DE),A	MC$_1$	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC$_2$	T_i	*	Z	1	1	1	1	1	1	1
	MC$_3$	$T_1T_2T_3$	BC DE	A	1	0	0	1	1	1	1
LD (mn),A	MC$_1$	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC$_2$	$T_1T_2T_3$	1st operand Address	n	0	1	0	1	1	1	1
	MC$_3$	$T_1T_2T_3$	2nd operand Address	m	0	1	0	1	1	1	1
	MC$_4$	T_i	*	Z	1	1	1	1	1	1	1
	MC$_5$	$T_1T_2T_3$	mn	A	1	0	0	1	1	1	1
LD A,I (Note 4) LD A,R	MC$_1$	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
LD I,A LD R,A	MC$_2$	$T_1T_2T_3$	2nd opcode Address	2nd opcode	0	1	0	1	0	1	1
LD ww, mn	MC$_1$	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC$_2$	$T_1T_2T_3$	1st operand Address	n	0	1	0	1	1	1	1
	MC$_3$	$T_1T_2T_3$	2nd operand Address	m	0	1	0	1	1	1	1
LD IX,mn LD IY,mn	MC$_1$	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC$_2$	$T_1T_2T_3$	2nd opcode Address	2nd opcode	0	1	0	1	0	1	1
	MC$_3$	$T_1T_2T_3$	1st operand Address	n	0	1	0	1	1	1	1
	MC$_4$	$T_1T_2T_3$	2nd operand Address	m	0	1	0	1	1	1	1
LD HL, (mn)	MC$_1$	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC$_2$	$T_1T_2T_3$	1st operand Address	n	0	1	0	1	1	1	1

(continued)

Note 4 Interrupt request is not sampled

Instruction	Machine Cycle	States	Address	Data	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{LIR}$	$\overline{HALT}$	ST
LD HL, (mn)	MC_3	$T_1T_2T_3$	2nd operand Address	m	0	1	0	1	1	1	1
	MC_4	$T_1T_2T_3$	mn	Data	0	1	0	1	1	1	1
	MC_5	$T_1T_2T_3$	mn+ 1	Data	0	1	0	1	1	1	1
LD ww,(mn)	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd opcode Address	2nd opcode	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	1st operand Address	n	0	1	0	1	1	1	1
	MC_4	$T_1T_2T_3$	2nd operand Address	m	0	1	0	1	1	1	1
	MC_5	$T_1T_2T_3$	mn	Data	0	1	0	1	1	1	1
	MC_6	$T_1T_2T_3$	mn+ 1	Data	0	1	0	1	1	1	1
LD IX,(mn) LD IY,(mn)	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd opcode Address	2nd opcode	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	1st operand Address	n	0	1	0	1	1	1	1
	MC_4	$T_1T_2T_3$	2nd operand Address	m	0	1	0	1	1	1	1
	MC_5	$T_1T_2T_3$	mn	Data	0	1	0	1	1	1	1
	MC_6	$T_1T_2T_3$	mn+ 1	Data	0	1	0	1	1	1	1
LD (mn),HL	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	1st operand Address	n	0	1	0	1	1	1	1
	MC_3	$T_1T_2T_3$	2nd operand Address	m	0	1	0	1	1	1	1
	MC_4	T_I	•	Z	1	1	1	1	1	1	1
	MC_5	$T_1T_2T_3$	mn	L	1	0	0	1	1	1	1
	MC_5	$T_1T_2T_3$	mn+ 1	H	1	0	0	1	1	1	1

(continued)

Instruction	Machine Cycle	States	Address	Data	$\overline{\text{RD}}$	$\overline{\text{WR}}$	$\overline{\text{ME}}$	$\overline{\text{IOE}}$	$\overline{\text{LIR}}$	$\overline{\text{HALT}}$	ST
LD (mn),ww	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd opcode Address	2nd opcode	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	1st operand Address	n	0	1	0	1	1	1	1
	MC_4	$T_1T_2T_3$	2nd operand Address	m	0	1	0	1	1	1	1
	MC_5	T_i	*	Z	1	1	1	1	1	1	1
	MC_6	$T_1T_2T_3$	mn	wwL	1	0	0	1	1	1	1
	MC_7	$T_1T_2T_3$	mn+1	wwH	1	0	0	1	1	1	1
LD (mn),IX LD (mn),IY	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd opcode Address	2nd opcode	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	1st operand Address	n	0	1	0	1	1	1	1
	MC_4	$T_1T_2T_3$	2nd operand Address	m	0	1	0	1	1	1	1
	MC_5	T_i	*	Z	1	1	1	1	1	1	1
	MC_6	$T_1T_2T_3$	mn	IXL IYL	1	0	0	1	1	1	1
	MC_7	$T_1T_2T_3$	mn+1	IXH IYH	1	0	0	1	1	1	1
LD SP, HL	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC_2	T_i	*	Z	1	1	1	1	1	1	1
LD SP,IX LD SP,IY	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd opcode Address	2nd opcode	0	1	0	1	0	1	1
	MC_3	T_i	*	Z	1	1	1	1	1	1	1
LDI LDD	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd opcode Address	2nd opcode	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	HL	Data	0	1	0	1	1	1	1
	MC_4	$T_1T_2T_3$	DE	Data	1	0	0	1	1	1	1

(continued)

Instruction	Machine Cycle	States	Address	Data	$\overline{\text{RD}}$	$\overline{\text{WR}}$	$\overline{\text{ME}}$	$\overline{\text{IOE}}$	$\overline{\text{LIR}}$	$\overline{\text{HALT}}$	ST
LDIR LDDR (If $BC_R \neq 0$)	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd opcode Address	2nd opcode	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	HL	Data	0	1	0	1	1	1	1
	MC_4	$T_1T_2T_3$	DE	Data	1	0	0	1	1	1	1
	MC_5– MC_6	TiTi	•	Z	1	1	1	1	1	1	1
LDIR LDDR (If $BC_R = 0$)	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd opcode Address	2nd opcode	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	HL	Data	0	1	0	1	1	1	1
	MC_4	$T_1T_2T_3$	DE	Data	1	0	0	1	1	1	1
MLT ww**	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd opcode Address	2nd opcode	0	1	0	1	0	1	1
	MC_3– MC_{13}	TiTiTiTi TiTiTiTi TiTiTi	•	Z	1	1	1	1	1	1	1
NEG	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd opcode Address	2nd opcode	0	1	0	1	0	1	1
NOP	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
OUT (m),A	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	1st operand Address	m	0	1	0	1	1	1	1
	MC_3	Ti	•	Z	1	1	1	1	1	1	1
	MC_4	$T_1T_2T_3$	m to A_0–A_7 A to A_8–A_{15}	A	1	0	1	0	1	1	1

(continued)

Instruction	Machine Cycle	States	Address	Data	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{LIR}$	$\overline{HALT}$	ST
OUT (C),g	MC$_1$	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC$_2$	$T_1T_2T_3$	2nd opcode Address	2nd opcode	0	1	0	1	0	1	1
	MC$_3$	T_i	*	Z	1	1	1	1	1	1	1
	MC$_4$	$T_1T_2T_3$	BC	g	1	0	1	0	1	1	1
OUT0 (m),g**	MC$_1$	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC$_2$	$T_1T_2T_3$	2nd opcode Address	2nd opcode	0	1	0	1	0	1	1
	MC$_3$	$T_1T_2T_3$	1st operand Address	m	0	1	0	1	1	1	1
	MC$_4$	T_i	*	Z	1	1	1	1	1	1	1
	MC$_5$	$T_1T_2T_3$	m to A_0-A_7 00H to A_8-A_{15}	g	1	0	1	0	1	1	1
OTIM** OTDM**	MC$_1$	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC$_2$	$T_1T_2T_3$	2nd opcode Address	2nd opcode	0	1	0	1	0	1	1
	MC$_3$	T_i	*	Z	1	1	1	1	1	1	1
	MC$_4$	$T_1T_2T_3$	HL	Data	0	1	0	1	1	1	1
	MC$_5$	$T_1T_2T_3$	C to A_0-A_7 00H to A_8-A_{15}	Data	1	0	1	0	1	1	1
	MC$_6$	T_i	*	Z	1	1	1	1	1	1	1
OTIMR** OTDMR** (If Br≠0)	MC$_1$	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC$_2$	$T_1T_2T_3$	2nd opcode Address	2nd opcode	0	1	0	1	0	1	1
	MC$_3$	T_i	*	Z	1	1	1	1	1	1	1
	MC$_4$	$T_1T_2T_3$	HL	Data	0	1	0	1	1	1	1
	MC$_5$	$T_1T_2T_3$	C to A_0-A_7 00H to A_8-A_{15}	Data	1	0	1	0	1	1	1
	MC$_6$– MC$_8$	$T_iT_iT_i$	*	Z	1	1	1	1	1	1	1
OTIMR** OTDMR** (If Br=0)	MC$_1$	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC$_2$	$T_1T_2T_3$	2nd opcode Address	2nd opcode	0	1	0	1	0	1	1
	MC$_3$	T_i	*	Z	1	1	1	1	1	1	1
	MC$_4$	$T_1T_2T_3$	HL	Data	0	1	0	1	1	1	1
	MC$_5$	$T_1T_2T_3$	C to A_0-A_7 00H to A_8-A_{15}	Data	1	0	1	0	1	1	1
	MC$_6$	T_i	*	Z	1	1	1	1	1	1	1

(continued)

Instruction	Machine Cycle	States	Address	Data	$\overline{\text{RD}}$	$\overline{\text{WR}}$	$\overline{\text{ME}}$	$\overline{\text{IOE}}$	$\overline{\text{LIR}}$	$\overline{\text{HALT}}$	ST
OUTI OUTD	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd opcode Address	2nd opcode	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	HL	Data	0	1	0	1	1	1	1
	MC_4	$T_1T_2T_3$	BC	Data	1	0	1	0	1	1	1
OTIR OTDR (If $Br \neq 0$)	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd opcode Address	2nd opcode	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	HL	Data	0	1	0	1	1	1	1
	MC_4	$T_1T_2T_3$	BC	Data	1	0	1	0	1	1	1
	MC_5– MC_6	TiTi	•	Z	1	1	1	1	1	1	1
OTIR OTDR (If $Br = 0$)	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd opcode Address	2nd opcode	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	HL	Data	0	1	0	1	1	1	1
	MC_4	$T_1T_2T_3$	BC	Data	1	0	1	0	1	1	1
POP zz	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	SP	Data	0	1	0	1	1	1	1
	MC_3	$T_1T_2T_3$	SP+1	Data	0	1	0	1	1	1	1
POP IX POP IY	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0

(continued)

Instruction	Machine Cycle	States	Address	Data	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{LIR}$	$\overline{HALT}$	ST
POP IX POP IY	MC_2	$T_1T_2T_3$	2nd opcode Address	2nd opcode	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	SP	Data	0	1	0	1	1	1	1
	MC_4	$T_1T_2T_3$	SP+1	Data	0	1	0	1	1	1	1
PUSH zz	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC_2– MC_3	TiTi	•	Z	1	1	1	1	1	1	1
	MC_4	$T_1T_2T_3$	SP−1	zzH	1	0	0	1	1	1	1
	MC_5	$T_1T_2T_3$	SP−2	zzL	1	0	0	1	1	1	1
PUSH IX PUSH IY	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd opcode Address	2nd opcode	0	1	0	1	0	1	1
	MC_3– MC_4	TiTi	•	Z	1	1	1	1	1	1	1
	MC_5	$T_1T_2T_3$	SP−1	IXH IYH	1	0	0	1	1	1	1
	MC_6	$T_1T_2T_3$	SP−2	IXL IYL	1	0	0	1	1	1	1
RET	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	SP	Data	0	1	0	1	1	1	1
	MC_3	$T_1T_2T_3$	SP+1	Data	0	1	0	1	1	1	1
RET f (If condition is false)	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC_2– MC_3	TiTi	•	Z	1	1	1	1	1	1	1
RET f (If condition is true)	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC_2	Ti	•	Z	1	1	1	1	1	1	1
	MC_3	$T_1T_2T_3$	SP	Data	0	1	0	1	1	1	1
	MC_4	$T_1T_2T_3$	SP+1	Data	0	1	0	1	1	1	1

(continued)

Instruction	Machine Cycle	States	Address	Data	$\overline{\text{RD}}$	$\overline{\text{WR}}$	$\overline{\text{ME}}$	$\overline{\text{IOE}}$	$\overline{\text{LIR}}$	$\overline{\text{HALT}}$	ST
RETI	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0 [5] 1	1	0
	MC_2	$T_1T_2T_3$	2nd opcode Address	2nd opcode	0	1	0	1	0 [5] 1	1	1
	MC_3– MC_5	$T_iT_iT_i$	*	Z	1	1	1	1	1 [5] 1	1	1
	MC_6	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0 [5] 0	1	1
	MC_7	T_i	*	Z	1	1	1	1	1 [5] 1	1	1
	MC_8	$T_1T_2T_3$	2nd opcode Address	2nd opcode	0	1	0	1	0 [5] 1	1	1
	MC_9	$T_1T_2T_3$	SP	Data	0	1	0	1	1 [5] 1	1	1
	MC_{10}	$T_1T_2T_3$	SP+1	Data	0	1	0	1	1 [5] 1	1	1
RLCA RLA RRCA RRA	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
RLC g RL g RRC g RR g SLA g SRA g SRL g	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd opcode Address	2nd opcode	0	1	0	1	0	1	1
	MC_3	T_i	*	Z	1	1	1	1	1	1	1
RLC (HL) RL (HL) RRC (HL) RR (HL) SLA (HL) SRA (HL) SRL (HL)	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd opcode Address	2nd opcode	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	HL	Data	0	1	0	1	1	1	1
	MC_4	T_i	*	Z	1	1	1	1	1	1	1
	MC_5	$T_1T_2T_3$	HL	Data	1	0	0	1	1	1	1

(continued)

Note 5 The upper and lower data show the state of LIR when LIRE = 1 and LIRE = 0 respectively

	Machine										
Instruction	Cycle	States	Address	Data	$\overline{\text{RD}}$	$\overline{\text{WR}}$	$\overline{\text{ME}}$	$\overline{\text{IOE}}$	$\overline{\text{LIR}}$	$\overline{\text{HALT}}$	ST
RLC (IX + d) RLC (IY + d) RL (IX + d) RL (IY + d) RRC (IX + d) RRC (IY + d) RR (IX + d) RR (IY + d) SLA (IX + d) SLA (IY + d) SRA (IX + d) SRA (IY + d) SRL (IX + d) SRL (IY + d)	MC$_1$	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC$_2$	$T_1T_2T_3$	2nd opcode Address	2nd opcode	0	1	0	1	0	1	1
	MC$_3$	$T_1T_2T_3$	1st operand Address	d	0	1	0	1	1	1	1
	MC$_4$	$T_1T_2T_3$	3rd opcode Address	3rd opcode	0	1	0	1	0	1	1
	MC$_5$	$T_1T_2T_3$	IX+d IY+d	Data	0	1	0	1	1	1	1
	MC$_6$	Ti	•	Z	1	1	1	1	1	1	1
	MC$_7$	$T_1T_2T_3$	IX+d IY+d	Data	1	0	0	1	1	1	1
RLD RRD	MC$_1$	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC$_2$	$T_1T_2T_3$	2nd opcode Address	2nd opcode	0	1	0	1	0	1	1
	MC$_3$	$T_1T_2T_3$	HL	Data	0	1	0	1	1	1	1
	MC$_4$– MC$_7$	TiTiTiTi	•	Z	1	1	1	1	1	1	1
	MC$_8$	$T_1T_2T_3$	HL	Data	1	0	0	1	1	1	1
RST v	MC$_1$	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC$_2$– MC$_3$	TiTi	•	Z	1	1	1	1	1	1	1
	MC$_4$	$T_1T_2T_3$	SP−1	PCH	1	0	0	1	1	1	1
	MC$_5$	$T_1T_2T_3$	SP−2	PCL	1	0	0	1	1	1	1
SCF	MC$_1$	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
SET b,g RES b,g	MC$_1$	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC$_2$	$T_1T_2T_3$	2nd opcode Address	2nd opcode	0	1	0	1	0	1	1
	MC$_3$	Ti	•	Z	1	1	1	1	1	1	1
SET b, (HL) RES b, (HL)	MC$_1$	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC$_2$	$T_1T_2T_3$	2nd opcode Address	2nd opcode	0	1	0	1	0	1	1
	MC$_3$	$T_1T_2T_3$	HL	Data	0	1	0	1	1	1	1
	MC$_4$	Ti	•	Z	1	1	1	1	1	1	1
	MC$_5$	$T_1T_2T_3$	HL	Data	1	0	0	1	1	1	1

(continued)

Instruction	Machine Cycle	States	Address	Data	$\overline{\text{RD}}$	$\overline{\text{WR}}$	$\overline{\text{ME}}$	$\overline{\text{IOE}}$	$\overline{\text{LIR}}$	$\overline{\text{HALT}}$	ST

SET b, (IX+d)
SET b, (IY+d)
RES b, (IX+d)
RES b, (IY+d)
	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0

	MC_2	$T_1T_2T_3$	2nd opcode Address	2nd opcode	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	1st operand Address	d	0	1	0	1	1	1	1
	MC_4	$T_1T_2T_3$	3rd opcode Address	3rd opcode	0	1	0	1	0	1	1
	MC_5	$T_1T_2T_3$	IX+d IY+d	Data	0	1	0	1	1	1	1
	MC_6	T_I	*	Z	1	1	1	1	1	1	1
	MC_7	$T_1T_2T_3$	IX+d IY+d	Data	1	0	0	1	1	1	1
SLP**	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd opcode Address	2nd opcode	0	1	0	1	0	1	1
	–	–	FFFFFH	Z	1	1	1	1	1	0	1
TSTIO m**	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$ Address	2nd opcode opcode	2nd	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	1st operand Address	m	0	1	0	1	1	1	1
	MC_4	$T_1T_2T_3$	C to A_0-A_7 00H to A_8-A_{15}	Data	0	1	1	0	1	1	1
TST g**	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd opcode Address	2nd opcode	0	1	0	1	0	1	1
	MC_3	T_I	*	Z	1	1	1	1	1	1	1
TST m**	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd opcode Address	2nd opcode	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	1st operand Address	m	0	1	0	1	1	1	1
TST (HL)**	MC_1	$T_1T_2T_3$	1st opcode Address	1st opcode	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd opcode Address	2nd opcode	0	1	0	1	0	1	1
	MC_3	T_I	*	Z	1	1	1	1	1	1	1
	MC_4	$T_1T_2T_3$	HL	Data	0	1	0	1	1	1	1

(continued)

INTERRUPT

Instruction	Machine Cycle	States	Address	Data	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{LIR}$	$\overline{HALT}$	ST
$\overline{NMI}$	MC_1	$T_1T_2T_3$	Next opcode Address (PC)	Z	0	1	0	1	0	1	0
	MC_2– MC_3	TiTi	•	Z	1	1	1	1	1	1	1
	MC_4	$T_1T_2T_3$	SP−1	PCH	1	0	0	1	1	1	1
	MC_5	$T_1T_2T_3$	SP−2	PCL	1	0	0	1	1	1	1
$\overline{INT_0}$ Mode 0 (RST Inserted)	MC_1	$T_1T_2T_W$ T_WT_3	Next opcode Address (PC)	1st opcode	1	1	1	0	0	1	0
	MC_2– MC_3	TiTi	•	Z	1	1	1	1	1	1	1
	MC_4	$T_1T_2T_3$	SP−1	PCH	1	0	0	1	1	1	1
	MC_5	$T_1T_2T_3$	SP−2	PCL	1	0	0	1	1	1	1
$\overline{INT_0}$ Mode 0 (CALL Inserted)	MC_1	$T_1T_2T_W$ T_WT_3	Next opcode Address (PC)	1st opcode	1	1	1	0	0	1	0
	MC_2	$T_1T_2T_3$	PC	n	0	1	0	1	1	1	1
	MC_3	$T_1T_2T_3$	PC+1	m	0	1	0	1	1	1	1
	MC_4	Ti	•	Z	1	1	1	1	1	1	1
	MC_5	$T_1T_2T_3$	SP−1	PC+2(H)	1	0	0	1	1	1	1
	MC_6	$T_1T_2T_3$	SP−2	PC+2(L)	1	0	0	1	1	1	1
$\overline{INT_0}$ Mode 1	MC_1	$T_1T_2T_W$ T_WT_3	Next opcode Address (PC)	Z	1	1	1	0	0	1	0
	MC_2	$T_1T_2T_3$	SP−1	PCH	1	0	0	1	1	1	1
	MC_3	$T_1T_2T_3$	SP−2	PCL	1	0	0	1	1	1	1
$\overline{INT_0}$ Mode 2	MC_1	$T_1T_2T_W$ T_WT_3	Next opcode Address (PC)	Vector	1	1	1	0	0	1	0
	MC_2	Ti	•	Z	1	1	1	1	1	1	1
	MC_3	$T_1T_2T_3$	SP−1	PCH	1	0	0	1	1	1	1
	MC_4	$T_1T_2T_3$	SP−2	PCL	1	0	0	1	1	1	1
	MC_5	$T_1T_2T_3$	I, Vector	Data	0	1	0	1	1	1	1
	MC_6	$T_1T_2T_3$	I, Vector+1	Data	0	1	0	1	1	1	1
$\overline{INT_1}$ $\overline{INT_2}$ Internal Interrupts	MC_1	$T_1T_2T_W$ T_WT_3	Next opcode Address (PC)	Z	1	1	1	1	1	1	0
	MC_2	Ti	•	Z	1	1	1	1	1	1	1
	MC_3	$T_1T_2T_3$	SP−1	PCH	1	0	0	1	1	1	1
	MC_4	$T_1T_2T_3$	SP−2	PCL	1	0	0	1	1	1	1
	MC_5	$T_1T_2T_3$	I, Vector	Data	0	1	0	1	1	1	1
	MC_6	$T_1T_2T_3$	I, Vector+1	Data	0	1	0	1	1	1	1

■ OPERATING MODES

Request Acceptance in Each Operating Mode

Table 25 Request Acceptance

Request		Normal Operation (CPU mode) (I/O Stop mode)	Wait State	Refresh Cycle	Interrupt Acknowledge Cycle	DMA Cycle	Bus Release Mode	Sleep mode	System Stop Mode
$\overline{\text{WAIT}}$		Accepted	Accepted	Not accepted	Accepted	Accepted	Not accepted	Not accepted	Not accepted
Refresh Request (Request of Refresh by the on-chip Refresh Controller)		Refresh cycle begins at the end of MC	Not accepted	Not accepted	Refresh cycle begins at the end of MC	Refresh cycle begins at the end of MC	Not accepted	Not accepted	Not accepted
$\overline{\text{DREQ}_0}$ $\overline{\text{DREQ}_1}$		DMA cycle begins at the end of MC	DMA cycle begins at the end of MC	Accepted If refresh cycle precedes DMA cycle begins at the end of one MC	Accepted DMA cycle begins at the end of MC	Accepted Refer to Section 10 "DMA Controller" for details	Accepted *, After bus frelease cycle, DMA cycle begins at the end of one MC	Not accepted	Not accepted
$\overline{\text{BUSREQ}}$		Bus is released at the end of MC	Not accepted	Not accepted	Bus is released at the end of MC	Bus is released at the end of MC	Continue bus release mode.	Accepted	Accepted
Interrupt	$\overline{\text{INT}_0}$, $\overline{\text{INT}_1}$, $\overline{\text{INT}_2}$	Accepted after executing the current instruction.	Accepted after executing the current instruction	Not accepted	Not accepted	Not accepted	Not accepted	Accepted Return from sleep mode to normal operation	Accepted Return from system stop mode to normal operation
	Internal I/O Interrupt	Accepted after executing the current instruction	Accepted after executing the current instruction	Not accepted	Not accepted	Not accepted	Not accepted	Accepted Return from sleep mode to normal operation	Not accepted
	$\overline{\text{NMI}}$	Accepted after executing the current instruction.	Accepted after executing the current instruction	Not accepted	Not accepted Interrupt acknowledge cycle precedes $\overline{\text{NMI}}$ is accepted after executing the next instruction	Accepted DMA cycle stops	Not accepted	Accepted Return from sleep mode to normal operation	Acceptable Return from system stop mode to normal operation.

Notes *. not acceptable when DMA Request is in level sense
MC: Machine Cycle

Request Priority

The HD643180X/HD647180X has the following three types of requests.

Type 1: To be accepted in specified state .... WAIT

Type 2: To be accepted in each machine cycle .... Refresh Req.
DMA Req.
Bus Req.

Type 3: To be accepted in each instruction .... Interrupt Req.

Type 1, type 2, and type 3 request priority is as follows:

Highest priority Type 1 > Type 2 > Type 3 Lowest priority

Type 2 request priority is as follows:

Highest priority Bus Req. > Refresh Req. > DMA Req. Lowest priority

Note : If Bus Req. and Refresh Req. occurs simultaneously, Bus Req. is accepted.

Refer to "Section 8, Interrupts" for type 3 request priority.

Type 4: To be accepted in last machine cycle

Highest priority Bus Req. from Bus masters > Interrupt Req.

Operation Mode Transition

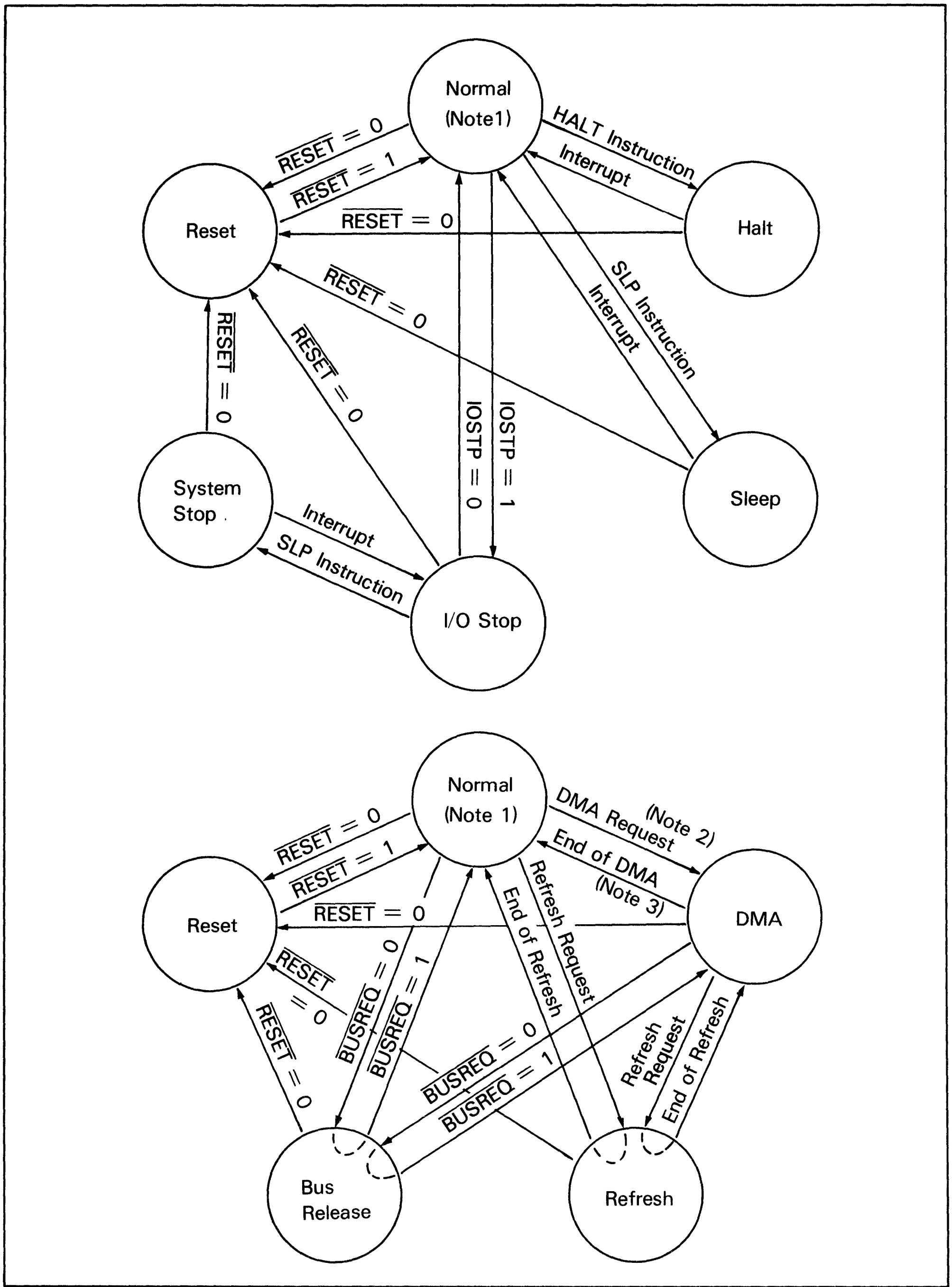


Figure 19. Operation Mode Transitions

Notes : 1 Normal. CPU executes instructions normally in normal mode.

2. DMA request: DMA is requested in the following cases.
(1) $\overline{DREQ_0}$, $\overline{DREQ_1}$ = 0 (memory to/from (memory-mapped) I/O DMA transfer)
(2) DE0 = 1 (memory to/from memory DMA transfer)

3. DMA end: DMA ends in the following cases.
(1) $\overline{DREQ_0}$, $\overline{DREQ_1}$ = 1 (memory to/from (memory-mapped) I/O DMA transfer)
(2) BCR0, BCR1 = 0000H (all DMA transfers)
(3) $\overline{NMI}$ = 0 (all DMA transfers)

The following operation mode transitions are also possible

$$\text{Halt} \rightleftarrows \begin{Bmatrix} \text{DMA} \\ \text{Refresh} \\ \text{Bus Release} \end{Bmatrix}$$

$$\text{I/O Stop} \rightleftarrows \begin{Bmatrix} \text{DMA} \\ \text{Refresh} \\ \text{Bus Release} \end{Bmatrix}$$

Sleep ⇄ Bus Release

System Stop ⇄ Bus Release

Status Signals

Table 26. shows pin outputs in each operating mode.

Table 26 Pin Outputs

Mode		$\overline{\text{LIR}}$	$\overline{\text{ME}}$	$\overline{\text{IOE}}$	$\overline{\text{RD}}$	$\overline{\text{WR}}$	$\overline{\text{REF}}$	$\overline{\text{HALT}}$	$\overline{\text{BUSACK}}$	ST	Address Bus	Data Bus
CPU operation	Opcode Fetch (1st opcode)	0	0	1	0	1	1	1	1	0	A	In
	Opcode Fetch (except 1st opcode)	0	0	1	0	1	1	1	1	1	A	In
	Memory Read	1	0	1	0	1	1	1	1	1	A	In
	Memory Write	1	0	1	1	0	1	1	1	1	A	Out
	I/O Read	1	1	0	0	1	1	1	1	1	A	In
	I/O Write	1	1	0	1	0	1	1	1	1	A	Out
	Internal Operation	1	1	1	1	1	1	1	1	1	A	In
Refresh		1	0	1	1	1	0	1	1	*	A	In
Interrupt Acknowledge Cycle (1st machine cycle)	$\overline{\text{NMI}}$	0	0	1	0	1	1	1	1	0	A	In
	$\overline{\text{INT}_0}$	0	1	0	1	1	1	1	1	0	A	In
	$\overline{\text{INT}_1}$, $\overline{\text{INT}_2}$ & Internal Interrupts	1	1	1	1	1	1	1	1	0	A	In
Bus Release		1	Z	Z	Z	Z	1	1	0	*	Z	In
Halt		0	0	1	0	1	1	0	1	0	A	In
Sleep		1	1	1	1	1	1	0	1	1	1	In
Internal DMA	Memory Read	1	0	1	0	1	1	*	1	0	A	IN
	Memory Write	1	0	1	1	0	1	*	1	0	A	Out
	I/O Read	1	1	0	0	1	1	*	1	0	A	In
	I/O Write	1	1	0	1	0	1	*	1	0	A	Out
Reset		1	1	1	1	1	1	1	1	1	Z	In

Note 1 : High
0 : Low
A : Programmable
Z : High Impedance
In : Input
Out : Output
* : Invalid

■ INTERNAL I/O REGISTERS

By programming IOA7 in the I/O control register, internal I/O register addresses are relocatable within ranges from 0000H to 00FFH in the I/O address space.

<table>
<tr><th>Register</th><th>Mnemonic</th><th>Address</th><th>Remarks</th></tr>
<tr><td>ASCI Control Register A Channel 0</td><td>(CNTLA0)</td><td>0 0</td><td>
<table>
<tr><td>bit</td><td>MPE</td><td>RE</td><td>TE</td><td>$\overline{RTS0}$</td><td>MPBR/ EFR</td><td>MOD2</td><td>MOD1</td><td>MOD0</td></tr>
<tr><td>During reset</td><td>0</td><td>0</td><td>0</td><td>1</td><td>invalid</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td></tr>
</table>
Mode Selection
Multi Processor Bit Receive/ Error Flag Reset
Request To Send
Transmit Enable
Receive Enable
Multi Processor Enable
</td></tr>
<tr><td>ASCI Control Register A Channel 1</td><td>(CNTLA1)</td><td>0 1</td><td>
<table>
<tr><td>bit</td><td>MPE</td><td>RE</td><td>TE</td><td>CKA1D</td><td>MPBR/ EFR</td><td>MOD2</td><td>MOD1</td><td>MOD0</td></tr>
<tr><td>During reset</td><td>0</td><td>0</td><td>0</td><td>1</td><td>invalid</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td></tr>
</table>
Mode Selection
Multi Processor Bit Receive/ Error Flag Reset
CKA1 Disable
Transmit Enable
Receive Enable
Multi Processor Enable
<table>
<tr><td>MOD2, 1, 0</td><td></td></tr>
<tr><td>0 0 0</td><td>Start + 7 bit Data + 1 Stop</td></tr>
<tr><td>0 0 1</td><td>Start + 7 bit Data + 2 Stop</td></tr>
<tr><td>0 1 0</td><td>Start + 7 bit Data + Parity + 1 Stop</td></tr>
<tr><td>0 1 1</td><td>Start + 7 bit Data + Parity + 2 Stop</td></tr>
<tr><td>1 0 0</td><td>Start + 8 bit Data + 1 Stop</td></tr>
<tr><td>1 0 1</td><td>Start + 8 bit Data + 2 Stop</td></tr>
<tr><td>1 1 0</td><td>Start + 8 bit Data + Parity + 1 Stop</td></tr>
<tr><td>1 1 1</td><td>Start + 8 bit Data + Parity + 2 Stop</td></tr>
</table>
</td></tr>
<tr><td>ASCI Control Register B Channel 0</td><td>(CNTLB0)</td><td>0 2</td><td>
<table>
<tr><td>bit</td><td>MPBT</td><td>MP</td><td>$\overline{CTS}$/ PS</td><td>PEO</td><td>DR</td><td>SS2</td><td>SS1</td><td>SS0</td></tr>
<tr><td>During reset</td><td>invalid</td><td>0</td><td>*</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td></tr>
<tr><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td></tr>
</table>
Clock Source and Speed Select
Divide Ratio
Parity Even or Odd
Clear To Send/Prescale
Multi Processor
Multi Processor Bit Transmit

* $\overline{CTS}$ Depends on the condition of $\overline{CTS}$ Pin
PS Cleared to 0
</td></tr>
</table>

(continued)

Register	Mnemonic	Address	Remarks
ASCI Control Register B Channel 1	(CNTLB1)	0 3	

bit	MPBT	MP	$\overline{\text{CTS}}$/PS	PEO	DR	SS2	SS1	SS0
During reset	invalid	0	0	0	0	1	1	1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

SS2, SS1, SS0 — Clock Source and Speed Select
DR — Divide Ratio
PEO — Parity Even or Odd
$\overline{\text{CTS}}$/PS — Clear To Send/Prescale
MP — Multi Processor
MPBT — Multi Processor Bit Transmit

General divide ratio	PS = 0 (divide ratio = 10)		PS = 1 (divide ratio = 30)	
SS2,1,0	DR = 0(× 16)	DR = 1(× 64)	DR = 0(× 16)	DR = 1(× 64)
000	φ÷ 160	φ÷ 640	φ÷ 480	φ÷ 1920
001	÷ 320	÷ 1280	÷ 960	÷ 3840
010	÷ 640	÷ 2560	÷ 1920	÷ 7680
011	÷ 1280	÷ 5120	÷ 3840	÷ 15360
100	÷ 2560	÷ 10240	÷ 7680	÷ 30720
101	÷ 5120	÷ 20480	÷ 15360	÷ 61440
110	÷ 10240	÷ 40960	÷ 30720	÷ 122880
111	External clock (frequency < φ ÷ 40)			

Register	Mnemonic	Address	Remarks
ASCI Status Register Channel 0	(STAT0)	0 4	

bit	RDRF	OVRN	PE	FE	RIE	$\overline{\text{DCD0}}$	TDRE	TIE
During reset	0	0	0	0	0	•	••	0
R/W	R	R	R	R	R/W	R	R	R/W

TIE — Transmit Interrupt Enable
TDRE — Transmit Data Register Empty
$\overline{\text{DCD0}}$ — Data Carrier Detect
RIE — Receive Interrupt Enable
FE — Framing Error
PE — Parity Error
OVRN — Over Run Error
RDRF — Receive Data Register Full

•• $\overline{\text{CTS}}_0$ Pin	TDRE
L	1
H	0

• $\overline{\text{DCD}}_0$: Depends on the condition of $\overline{\text{DCD}}_0$ Pin.

Register	Mnemonic	Address	Remarks
ASCI Status Register Channel 1	(STAT1)	0 5	

bit	RDRF	OVRN	PE	FE	RIE	CTS1E	TDRE	TIE
During reset	0	0	0	0	0	0	1	0
R/W	R	R	R	R	R/W	R/W	R	R/W

TIE — Transmit Interrupt Enable
TDRE — Transmit Data Register Empty
CTS1E — $\overline{\text{CTS1}}$ Enable
RIE — Receive Interrupt Enable
FE — Framing Error
PE — Parity Error
OVRN — Over Run Error
RDRF — Receive Data Register Full

(continued)

<table>
<tr><th>Register</th><th>Mnemonic</th><th>Address</th><th>Remarks</th></tr>
<tr><td>ASCI Transmit Data Register Channel 0</td><td>(TDR0)</td><td>0 6</td><td></td></tr>
<tr><td>ASCI Transmit Data Register Channel 1</td><td>(TDR1)</td><td>0 7</td><td></td></tr>
<tr><td>ASCI Receive Data Register Channel 0</td><td>(TSR0)</td><td>0 8</td><td></td></tr>
<tr><td>ASCI Receive Data Register Channel 1</td><td>(TSR1)</td><td>0 9</td><td></td></tr>
<tr><td>CSI/O Control Register</td><td>(CNTR)</td><td>0 A</td><td>
<table>
<tr><td>bit</td><td>EF</td><td>EIE</td><td>RE</td><td>TE</td><td>–</td><td>SS2</td><td>SS1</td><td>SS0</td></tr>
<tr><td>During reset</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td></tr>
<tr><td>R/W</td><td>R</td><td>R/W</td><td>R/W</td><td>R/W</td><td></td><td>R/W</td><td>R/W</td><td>R/W</td></tr>
</table>
Speed Select
Transmit Enable
Receive Enable
End Interrupt Enable
End Flag
<table>
<tr><th>SS2,1,0</th><th>Baud Rate</th><th>SS2,1,0</th><th>Baud Rate</th></tr>
<tr><td>000</td><td>φ÷ 20</td><td>100</td><td>φ÷ 320</td></tr>
<tr><td>001</td><td>÷ 40</td><td>101</td><td>÷ 640</td></tr>
<tr><td>010</td><td>÷ 80</td><td>110</td><td>÷1280</td></tr>
<tr><td>011</td><td>÷160</td><td>111</td><td>External
(frequency < ÷ 20)</td></tr>
</table>
</td></tr>
<tr><td>CSI/O Transmit/Receive Data Register</td><td>(TRDR)</td><td>0 B</td><td></td></tr>
<tr><td>Timer Data Register Channel 0L</td><td>(TMDR0L)</td><td>0 C</td><td></td></tr>
<tr><td>Timer Data Register Channel 0H</td><td>(TMDR0H)</td><td>0 D</td><td></td></tr>
<tr><td>Timer Reload Register Channel 0L</td><td>(RLDR0L)</td><td>0 E</td><td></td></tr>
<tr><td>Timer Reload Register Channel 0H</td><td>(RLDR0H)</td><td>0 F</td><td></td></tr>
<tr><td>Timer Control Register</td><td>(TCR)</td><td>1 0</td><td>
<table>
<tr><td>bit</td><td>TIF1</td><td>TIF0</td><td>TIE1</td><td>TIE0</td><td>TOC1</td><td>TOC0</td><td>TDE1</td><td>TDE0</td></tr>
<tr><td>During reset</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>R/W</td><td>R</td><td>R</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td></tr>
</table>
Timer Down Count Enable 1,0
Timer Output Control 1,0
Timer Interrupt Enable 1,0
Timer Interrupt Flag 1,0
<table>
<tr><th>TOC1,0</th><th>TOUT1</th></tr>
<tr><td>00</td><td>1</td></tr>
<tr><td>01</td><td>Toggle</td></tr>
<tr><td>10</td><td>0</td></tr>
<tr><td>11</td><td>1</td></tr>
</table>
</td></tr>
</table>

(continued)

Register	Mnemonic	Address	Remarks
Timer Data Register Channel 1L	(TMDR1L)	1 4	
Timer Data Register Channel 1H	(TMDR1H)	1 5	
Timer Reload Register Channel 1L	(RLDR1L)	1 6	
Timer Reload Register Channel 1H	(RLDR1H)	1 7	
Free Running Counter	(FRC)	1 8	Read only
DMA Source Address Register Channel 0L	(SAR0L)	2 0	
DMA Source Address Register Channel 0H	(SAR0H)	2 1	
DMA Source Address Register Channel 0B	(SAR0B)	2 2	Bits 0-3 are used for SAR0B. A_{19}, A_{18}, A_{17}, A_{16} | DMA Transfer Request X X 0 0 | $\overline{DREQ_0}$ (external) X X 0 1 | RDR0 (ASCI0) X X 1 0 | RDR1 (ASCI1) X X 1 1 | Not Used
DMA Destination Address Register Channel 0L	(DAR0L)	2 3	
DMA Destination Address Register Channel 0H	(DAR0H)	2 4	
DMA Destination Address Register Channel 0B	(DAR0B)	2 5	Bits 0-3 are used for DAR0B. A_{19}, A_{18}, A_{17}, A_{16} | DMA Transfer Request X X 0 0 | $\overline{DREQ_0}$ (external) X X 0 1 | TDR0 (ASCI0) X X 1 0 | TDR1 (ASCI1) X X 1 1 | Not Used
DMA Byte Count Register Channel 0L	(BCR0L)	2 6	
DMA Byte Count Register Channel 0H	(BCR0H)	2 7	
DMA Memory Address Register Channel 1L	(MAR1L)	2 8	
DMA Memory Address Register Channel 1H	(MAR1H)	2 9	
DMA Memory Address Register Channel 1B	(MAR1B)	2 A	Bits 0-3 are used for MAR1B.
DMA I/O Address Register Channel 1L	(IAR1L)	2 B	
DMA I/O Address Register Channel 1H	(IAR1H)	2 C	

(continued)

Register	Mnemonic	Address	Remarks
DMA Byte Count Register Channel 1L	(BCR1L)	2 E	
DMA Byte Count Register Channel 1H	(BCR1H)	2 F	
DMA Status Register	(DSTAT)	3 0	see below
DMA Mode Register	(DMODE)	3 1	see below

DMA Status Register (DSTAT):

bit	DE1	DE0	$\overline{DWE1}$	$\overline{DWE0}$	DIE1	DIE0	–	DME
During reset	0	0	1	1	0	0	1	0
R/W	R/W	R/W	W	W	R/W	R/W		R

DME: DMA Master Enable
DIE1, DIE0: DMA Interrupt Enable 1,0
$\overline{DWE1}$, $\overline{DWE0}$: DMA Enable Bit Write Enable 1,0
DE1, DE0: DMA Enable ch 1,0

DMA Mode Register (DMODE):

bit	–	–	DM1	DM0	SM1	SM0	MMOD	–
During reset	1	1	0	0	0	0	0	1
R/W			R/W	R/W	R/W	R/W	R/W	

MMOD: Memory Mode Select
SM1, SM0: Ch 0 Source Mode 1,0
DM1, DM0: Ch 0 Destination Mode 1, 0

DM1, 0	Destination	Address
0 0	M	DAR0+ 1
0 1	M	DAR0− 1
1 0	M	DAR0 fixed
1 1	I/O	DAR0 fixed

SM1, 0	Source	Address
0 0	M	SAR0+ 1
0 1	M	SAR0− 1
1 0	M	SAR0 fixed
1 1	I/O	SAR0 fixed

MMOD	Mode
0	Cycle Steal Mode
1	Burst Mode

(continued)

Register	Mnemonic	Address	Remarks
DMA/Wait Control Resiter	(DCNTL)	3 2	

bit	MWI1	MWI0	IWI1	IWI0	DMS1	DMS0	DIM1	DIM0
During reset	1	1	1	1	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

DIM1, DIM0: DMA Ch 1 I/O Memory Mode Select
DMS1, DMS0: $\overline{\text{DREQi}}$ Select, i = 1,0
IWI1, IWI0: I/O Wait Insertion
MWI1, MWI0: Memory Wait Insertion

MWI1,0	Number of wait states
00	0
01	1
10	2
11	3

IWI1,0	Number of wait states
00	1
01	2
10	3
11	4

DMSi	Sense
1	Edge sense
0	Level sense

DIM1,0	Transfer Mode	Address Increment/Decrement	
00	M→I/O	MAR1+1	IAR1 fixed
01	M→I/O	MAR1−1	IAR1 fixed
10	I/O→M	IAR1 fixed	MAR1+1
11	I/O→M	IAR1 fixed	MAR1−1

Register	Mnemonic	Address
Interrupt Vector Low Register	(IL)	3 3

bit	IL7	IL6	IL5	—	—	—	—	—
During reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W					

IL7, IL6, IL5: Interrupt Vector Low

Register	Mnemonic	Address
INT/TRAP Control Register	(ITC)	3 4

bit	TRAP	UFO	—	—	—	ITE2	ITE1	ITE0
During reset	0	0	1	1	1	0	0	1
R/W	R/W	R				R/W	R/W	R/W

ITE2, ITE1, ITE0: $\overline{\text{INT}}$ Enable 2,1,0
UFO: Undefined Fetch Object
TRAP: TRAP

Register	Mnemonic	Address
Refresh Control Register	(RCR)	3 6

bit	REFE	REFW	—	—	—	—	CYC1	CYC0
During reset	1	1	1	1	1	1	0	0
R/W	R/W	R/W					R/W	R/W

CYC1, CYC0: Cycle Select
REFW: Refresh Wait State
REFE: Refresh Enable

CYC1,0	Interval of Refresh Cycle
00	10 States
01	20
10	40
11	80

(continued)

<table>
<tr><th>Register</th><th>Mnemonic</th><th>Address</th><th>Remarks</th></tr>
<tr><td>MMU Common Base Register</td><td>(CBR)</td><td>3 8</td><td><table>
<tr><td>bit</td><td>CB7</td><td>CB6</td><td>CB5</td><td>CB4</td><td>CB3</td><td>CB2</td><td>CB1</td><td>CB0</td></tr>
<tr><td>During reset</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td></tr>
</table>
CB7–CB0: MMU Common Base Register</td></tr>
<tr><td>MMU Bank Base Register</td><td>(BBR)</td><td>3 9</td><td><table>
<tr><td>bit</td><td>BB7</td><td>BB6</td><td>BB5</td><td>BB4</td><td>BB3</td><td>BB2</td><td>BB1</td><td>BB0</td></tr>
<tr><td>During reset</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td></tr>
</table>
BB7–BB0: MMU Bank Base Register</td></tr>
<tr><td>MMU Common/Bank Area Register</td><td>(CBAR)</td><td>3 A</td><td><table>
<tr><td>bit</td><td>CA3</td><td>CA2</td><td>CA1</td><td>CA0</td><td>BA3</td><td>BA2</td><td>BA1</td><td>BA0</td></tr>
<tr><td>During reset</td><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td></tr>
</table>
CA3–CA0: MMU Common Area Register
BA3–BA0: MMU Bank Area Register</td></tr>
<tr><td>Operation Mode Control Register</td><td>(OMCR)</td><td>3 E</td><td><table>
<tr><td>bit</td><td>LIRE</td><td>$\overline{\text{LIRTE}}$</td><td>$\overline{\text{IOC}}$</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td></tr>
<tr><td>During reset</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td></tr>
<tr><td>R/W</td><td>R/W</td><td>W</td><td>R/W</td><td></td><td></td><td></td><td></td><td></td></tr>
</table>
$\overline{\text{IOC}}$: I/O Compatibility
$\overline{\text{LIRTE}}$: $\overline{\text{LIR}}$ Temporary Enable
LIRE: $\overline{\text{LIR}}$ Enable</td></tr>
<tr><td>I/O Control Register</td><td>(ICR)</td><td>3 F</td><td><table>
<tr><td>bit</td><td>IOA7</td><td>—</td><td>IOSTP</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td></tr>
<tr><td>During reset</td><td>0</td><td>1</td><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td></tr>
<tr><td>R/W</td><td>R/W</td><td></td><td>R/W</td><td></td><td></td><td></td><td></td><td></td></tr>
</table>
IOSTP: I/O Stop
IOA7: I/O Address</td></tr>
<tr><td>Timer 2 Free-Running Counter L</td><td>(T2FRCL)</td><td>4 0</td><td><table>
<tr><td>bit</td><td>T2FRCL7</td><td>T2FRCL6</td><td>T2FRCL5</td><td>T2FRCL4</td><td>T2FRCL3</td><td>T2FRCL2</td><td>T2FRCL1</td><td>T2FRCL0</td></tr>
<tr><td>During reset</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td></tr>
</table></td></tr>
<tr><td>Timer 2 Free-Running Counter H</td><td>(T2FRCH)</td><td>4 1</td><td><table>
<tr><td>bit</td><td>T2FRCH7</td><td>T2FRCH6</td><td>T2FRCH5</td><td>T2FRCH4</td><td>T2FRCH3</td><td>T2FRCH2</td><td>T2FRCH1</td><td>T2FRCH0</td></tr>
<tr><td>During reset</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td></tr>
</table></td></tr>
<tr><td>Timer 2 Output Compare Register 1L</td><td>(T2OCR1L)</td><td>4 2</td><td><table>
<tr><td>bit</td><td>T2OCR1L7</td><td>T2OCR1L6</td><td>T2OCR1L5</td><td>T2OCR1L4</td><td>T2OCR1L3</td><td>T2OCR1L2</td><td>T2OCR1L1</td><td>T2OCR1L0</td></tr>
<tr><td>During reset</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td></tr>
<tr><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td></tr>
</table></td></tr>
<tr><td>Timer 2 Output Compare Register 1H</td><td>(T2OCR1H)</td><td>4 3</td><td><table>
<tr><td>bit</td><td>T2OCR1H7</td><td>T2OCR1H6</td><td>T2OCR1H5</td><td>T2OCR1H4</td><td>T2OCR1H3</td><td>T2OCR1H2</td><td>T2OCR1H1</td><td>T2OCR1H0</td></tr>
<tr><td>During reset</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td></tr>
<tr><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td></tr>
</table></td></tr>
</table>

Register	Mnemonic	Address	Remarks								
Timer 2 Output Compare Register 2L	(T2OCR2L)	4 4	bit	T2OCR2L7	T2OCR2L6	T2OCR2L5	T2OCR2L4	T2OCR2L3	T2OCR2L2	T2OCR2L1	T2OCR2L0
			During reset	1	1	1	1	1	1	1	1
			R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Timer 2 Output Compare Register 2H	(T2OCR2H)	4 5	bit	T2OCR2H7	T2OCR2H6	T2OCR2H5	T2OCR2H4	T2OCR2H3	T2OCR2H2	T2OCR2H1	T2OCR2H0
			During reset	1	1	1	1	1	1	1	1
			R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Timer 2 Input Capture Register L	(T2ICRL)	4 6	bit	T2ICRL7	T2ICRL6	T2ICRL5	T2ICRL4	T2ICRL3	T2ICRL2	T2ICRL1	T2ICRL0
			During reset	0	0	0	0	0	0	0	0
			R/W	R	R	R	R	R	R	R	R
Timer 2 Input Capture Register H	(T2ICRH)	4 7	bit	T2ICRH7	T2ICRH6	T2ICRH5	T2ICRH4	T2ICRH3	T2ICRH2	T2ICRH1	T2ICRH0
			During reset	0	0	0	0	0	0	0	0
			R/W	R	R	R	R	R	R	R	R
Timer 2 Control/status Register 1	(T2CSR1)	4 8	bit	ICF	OCF1	TOF	EICI	EOCI1	ETOI	IEDG	OLVL1
			During reset	0	0	0	0	0	0	0	0
			R/W	R	R	R	R/W	R/W	R/W	R/W	R/W
Timer 2 Control/status Register 2	(T2CSR2)	4 9	bit	ICF	OCF1	OCF2	–	EOCI2	OLVL2	–	–
			During reset	0	0	0	1	0	0	0	0
			R/W	R	R	R	–	R/W	R/W	R/W	R/W
Comparator Control/status Register	(CCSR)	5 0	bit	RBIT	–	AIN2	AIN1	AIN0	REF2	REF1	REF0
			During reset	Note	1	1	0	1	1	0	0
			R/W	R		R/W	R/W	R/W	R/W	R/W	R/W
			Note: Undefined until the first comparison result is stored								
RAM Control Register	(RMCR)	5 1	bit	RMCR7	RMCR6	RMCR5	RMCR4	–	–	–	–
			During reset	0	0	0	0	1	1	1	1
			R/W	R/W	R/W	R/W	R/W				

Register	Mnemonic	Address	Remarks								
Port A Disable Register	(DERA)	53	bit	TEND1E	DREQ1E	CKSE	RXSE	TXSE	CKA1E	RXA1E	TXA1E
			During reset	0	0	0	0	0	0	0	0
			R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Port A Input Data Register	(IDRA)	60	bit	IDRA7	IDRA6	IDRA5	IDRA4	IDRA3	IDRA2	IDRA1	IDRA0
			During reset	(Note 1)							
			R/W	R	R	R	R	R	R	R	R
Port A Output Data Register	(ODRA)	60	bit	ODRA7	ODRA6	ODRA5	ODRA4	ODRA3	ODRA2	ODRA1	ODRA0
			During reset	(Note 2)							
			R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Port B Input Data Register	(IDRB)	61	bit	IDRB7	IDRB6	IDRB5	IDRB4	IDRB3	IDRB2	IDRB1	IDRB0
			During reset	(Note 1)							
			R/W	R	R	R	R	R	R	R	R
Port B Output Data Register	(ODRB)	61	bit	ODRB7	ODRB6	ODRB5	ODRB4	ODRB3	ODRB2	ODRB1	ODRB0
			During reset	(Note 2)							
			R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Port C Input Data Register	(IDRC)	62	bit	IDRC7	IDRC6	IDRC5	IDRC4	IDRC3	IDRC2	IDRC1	IDRC0
			During reset	(Note 1)							
			R/W	R	R	R	R	R	R	R	R
Port C Output Data Register	(ODRC)	62	bit	ODRC7	ODRC6	ODRC5	ODRC4	ODRC3	ODRC2	ODRC1	ODRC0
			During reset	(Note 2)							
			R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Port D Input Data Register	(IDRD)	63	bit	IDRD7	IDRD6	IDRD5	IDRD4	IDRD3	IDRD2	IDRD1	IDRD0
			During reset	(Note 1)							
			R/W	R	R	R	R	R	R	R	R
Port D Output Data Register	(ODRD)	63	bit	ODRD7	ODRD6	ODRD5	ODRD4	ODRD3	ODRD2	ODRD1	ODRD0
			During reset	(Note 2)							
			R/W	W	W	W	W	W	W	W	W
Port E Input Data Register	(IDRE)	64	bit	IDRE7	IDRE6	IDRE5	IDRE4	IDRE3	IDRE2	IDRE1	IDRE0
			During reset	(Note 1)							
			R/W	R	R	R	R	R	R	R	R
Port E Output Data Register	(ODRE)	64	bit	ODRE7	ODRE6	ODRE5	ODRE4	ODRE3	ODRE2	ODRE1	ODRE0
			During reset	(Note 2)							
			R/W	R/W	R/W	R/W	R/W	W	W	W	W
Port F Input Data Register	(IDRF)	65	bit	IDRF7	IDRF6	IDRF5	IDRF4	IDRF3	IDRF2	IDRF1	IDRF0
			During reset	(Note 1)							
			R/W	R	R	R	R	R	R	R	R
Port F Output Data Register	(ODRF)	65	bit	ODRF7	ODRF6	ODRF5	ODRF4	ODRF3	ODRF2	ODRF1	ODRF0
			During reset	(Note 2)							
			R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Note 1. Fetches terminal status
2. Undefined until data is written

Register	Mnemonic	Address	Remarks							
Port G Input Data Register	(IDRG)	66	bit	–	–	IDRG5	IDRG4	IDRG3	IDRG2	IDRG1	IDRG0
			During reset	1	1	(Note 1)					
			R/W			R	R	R	R	R	R
			Note 1 Fetches terminal status								
Port A Data Direction Register	(DDRA)	70	bit	DDRA7	DDRA6	DDRA5	DDRA4	DDRA3	DDRA2	DDRA1	DDRA0
			During reset	0	0	0	0	0	0	0	0
			R/W	W	W	W	W	W	W	W	W
Port B Data Direction Register	(DDRB)	71	bit	DDRB7	DDRB6	DDRB5	DDRB4	DDRB3	DDRB2	DDRB1	DDRB0
			During reset	0	0	0	0	0	0	0	0
			R/W	W	W	W	W	W	W	W	W
Port C Data Direction Register	(DDRC)	72	bit	DDRC7	DDRC6	DDRC5	DDRC4	DDRC3	DDRC2	DDRC1	DDRC0
			During reset	0	0	0	0	0	0	0	0
			R/W	W	W	W	W	W	W	W	W
Port D Data Direction Register	(DDRD)	73	bit	DDRD7	DDRD6	DDRD5	DDRD4	DDRD3	DDRD2	DDRD1	DDRD0
			During reset	0	0	0	0	0	0	0	0
			R/W	W	W	W	W	W	W	W	W
Port E Data Direction Register	(DDRE)	74	bit	DDRE7	DDRE6	DDRE5	DDRE4	DDRE3	DDRE2	DDRE1	DDRE0
			During reset	0	0	0	0	0	0	0	0
			R/W	W	W	W	W	W	W	W	W
Port F Data Direction Register	(DDRF)	75	bit	DDRF7	DDRF6	DDRF5	DDRF4	DDRF3	DDRF2	DDRF1	DDRF0
			During reset	0	0	0	0	0	0	0	0
			R/W	W	W	W	W	W	W	W	W

HD648180W
MCU (Micro Controller Unit)

■ DESCRIPTION

The HD648180W is an 8-bit CMOS microcontroller in the HD64180 family. Its instruction set is upward-compatible with the HD64180Z, hence with the Z-80. Twelve instructions (seven types) have been added, bringing the total number of instructions to 165.

Compared with the HD64180Z, the HD648180W also has more peripheral functions. In addition to a refresh controller and wait-state controller for memory access support, and a memory management unit (MMU) and DMA controller (DMAC) for processing large quantities of data, the HD64180W has on-chip RAM, EEPROM, and an A/D converter. Timer and I/O-port functions have also been enhanced.

Figure 1 shows the HD64180 family. Table 1 lists the features of the HD64180W.

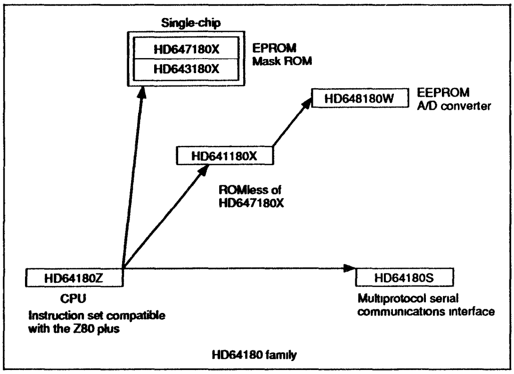
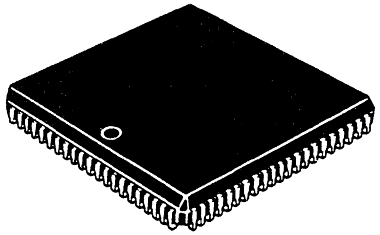
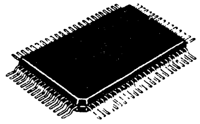


Figure 1 HD64180 Family



(CP-84)



(FP-80A)

■ ORDERING INFORMATION

Type No.	Speed and Package
HD648180W0CP4	4 MHz 84-pin PLCC
HD648180W0CP6	6 MHz 84-pin PLCC
HD648180W0F4	4 MHz 80-pin QFP
HD648180W0F6	6 MHz 80-pin QFP

Table 1 HD648180W Features

Item	Description
CPU	• Instruction set upward-compatible with HD64180Z • Maximum operating speed: 6.144 MHz
MMU	• 1-Mbyte physical address space
Memory	• 256-byte EEPROM on-chip —Byte or page write • 1-kbyte static RAM on-chip —Provides work area, stack area for interrupts, etc.
Timers (4 channels)	• 16-bit free-running timer (1 channel) on-chip —Input capture function for measuring input pulse width —Output compare function for generating waveforms • 16-bit reload timers (3 channels) on-chip —Toggle function for square-wave output with 50% duty cycle
Serial communication interface (2 channels)	• Asynchronous or clocked synchronous mode • Simultaneous transmit and receive (full duplex communication) • On-chip baud rate generator
DMA controller (2 channels)	• Directly addresses 1-Mbyte address space (without using MMU) • Directly addresses 64-kbyte I/O address space • Can transfer 64-kbyte data continuously • Data transfer speed (memory-to-memory): six system clocks/byte
A/D converter (8-bit resolution)	• Four analog input channels • Absolute precision: 2.0 LSB • Conversion time: 17 µs (at 6 MHz)
I/O ports	• 30 input/output lines (including 10 with direct LED driving capability: I_{OL} = 10 mA) • 4 input lines • 1 output line
Wait-state controller	• Wait states can be inserted by external signal input or by software
Refresh controller	• Outputs 8-bit refresh addresses • Programmable refresh interval

Table 1 HD648180W Features (cont)

Item	Description
Interrupts	• Four external interrupt lines: $\overline{NMI}$, $\overline{INT_0}$, $\overline{INT_1}$, $\overline{INT_2}$ • Ten on-chip interrupt sources
Special CPU modes	• Low-power modes (standby modes) • Sleep mode • Halt mode • Bus-release mode
Other features	• On-chip clock oscillator • Interfaces directly with Z-80 peripheral chips

■ Block Diagram

Figure 2 is a block diagram of the HD648180W.

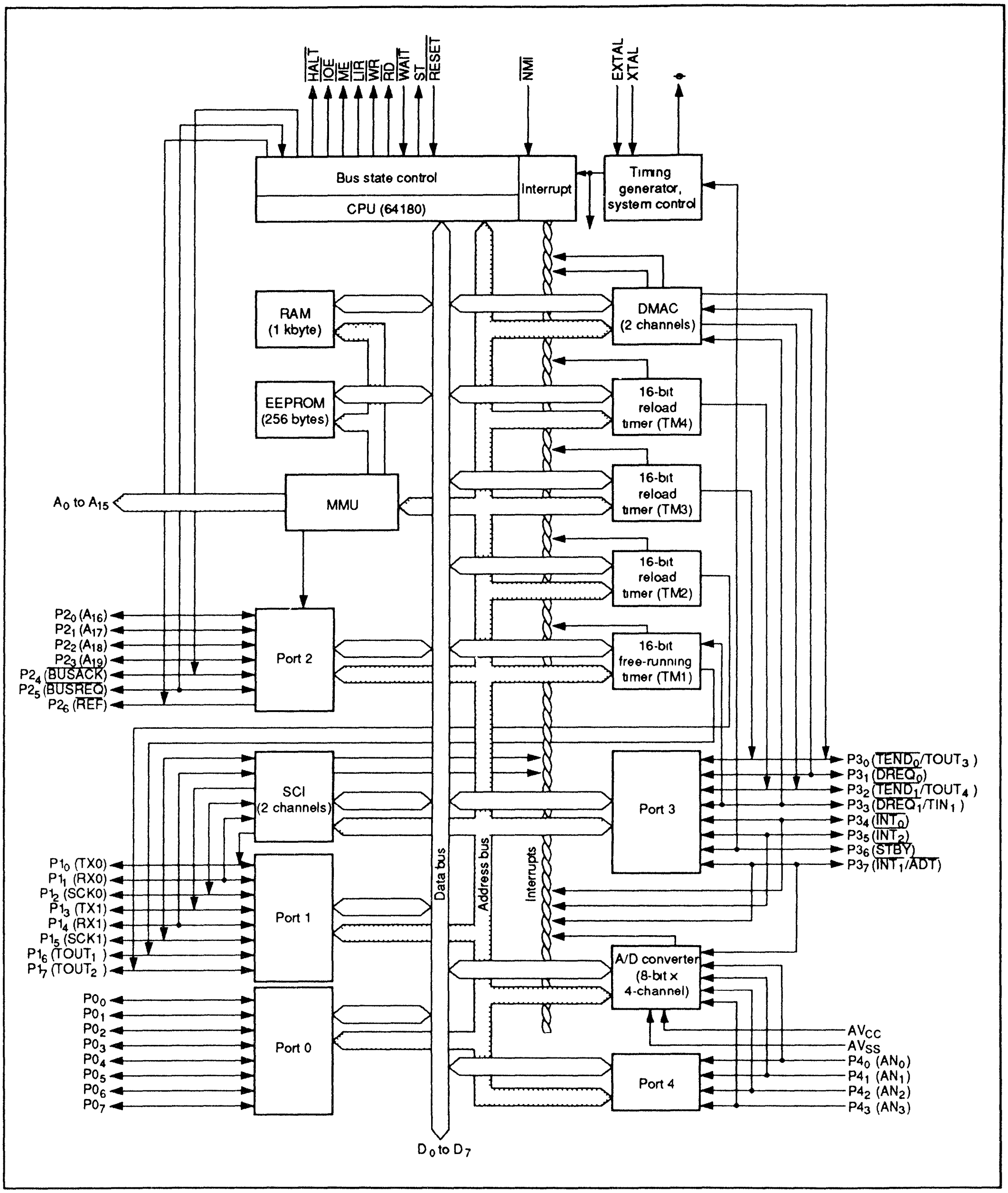


Figure 2 HD648180W Block Diagram

■ Pin Descriptions

• Pin Arrangement

Figure 3 shows the pin arrangement of the HD648180W in the FP-80A package. Figure 4 shows the pin arrangement in the CP-84 package.

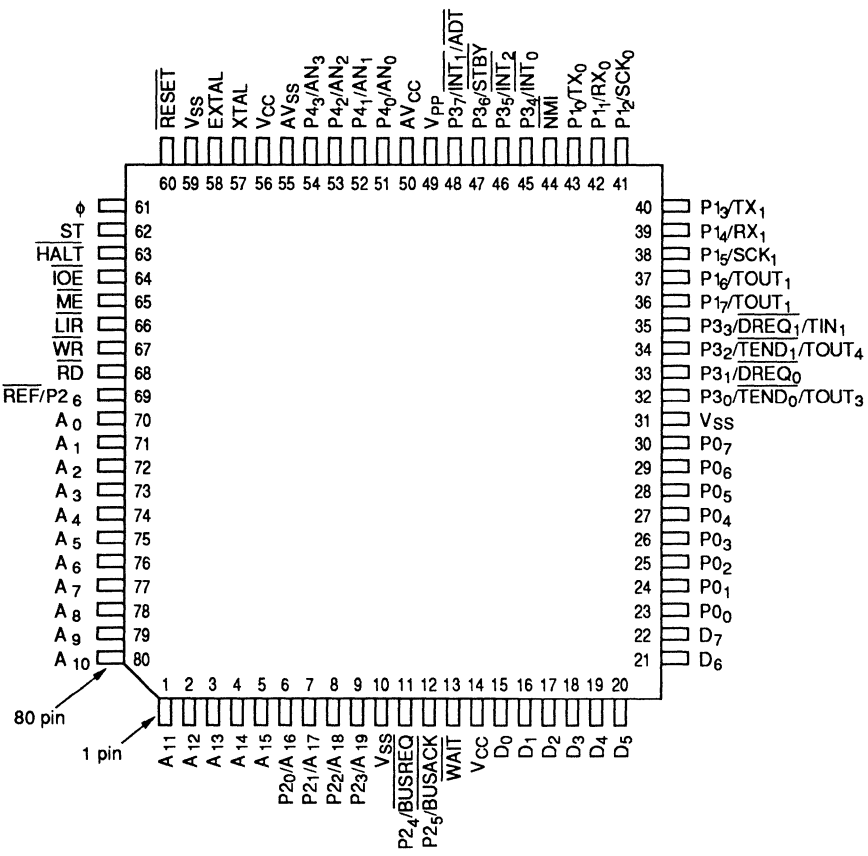
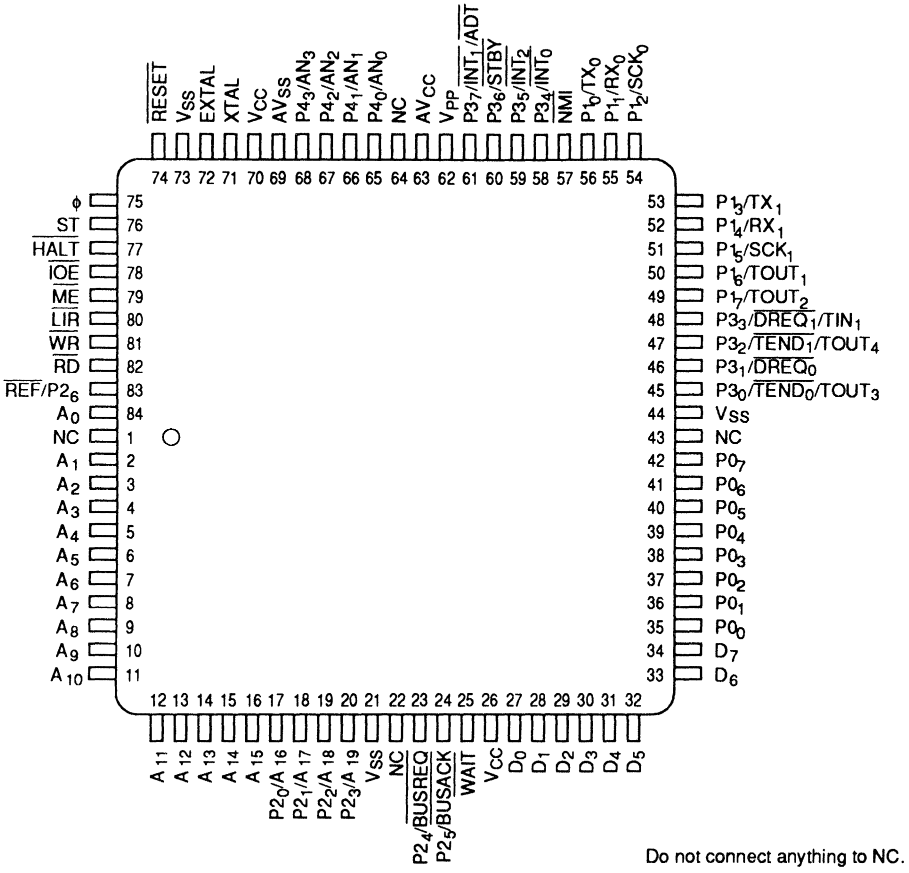


Figure 3 Pin Arrangement (FP-80A)



Do not connect anything to NC.

Figure 4 Pin Arrangement (CP-84)

- **Pin Functions**

Table 2 Pin Functions

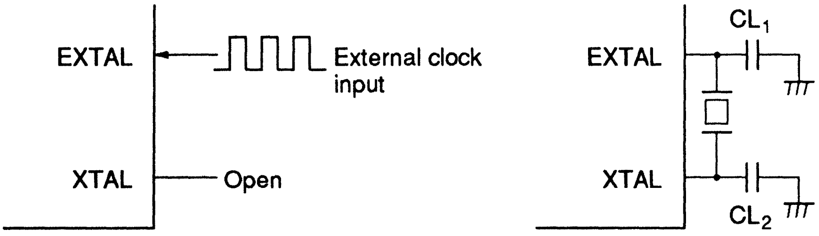
Type	Signal	Pin Number FP-80A	 PC-84	I/O	Name and Pin Function
Power supply GND	V_{CC}	14, 56	26, 70	Input	Power Supply: Connect all V_{CC} pins to system power supply (+5 V).
	V_{SS}	10, 31, 59	21, 44 73	Input	Ground: Connect all V_{SS} pins to system power supply (0 V).
XTAL clock	XTAL	57	71	Input	Connect to crystal oscillator with twice system clock (ϕ) frequency. When inputting external clock at EXTAL, leave XTAL open.
	EXTAL	58	72	Input	Connect to crystal oscillator or external clock. External clock frequency should be 2 times system clock.  Oscillator circuit examples
	ϕ	61	75	Output	System Clock: Supplies system clock to peripheral devices.
Reset	$\overline{\text{RESET}}$	60	74	Input	Reset: Chip is reset when this pin is dropped low.
Address bus	A_0 to A_{19}	1 to 9, 70 to 80	1 to 20, 84	Output (Tri-state)	Address Bus: For memory access. These lines go to high impedance only in the following cases. a. During reset b. When bus control is granted to another device (when $\overline{\text{BUSACK}}$ drops low because of low input at $\overline{\text{BUSREQ}}$).

Table 2 Pin Functions (cont)

Type	Signal	Pin Number		I/O	Name and Pin Function
		FP-80A	PC-84		
Data bus	D_0 to D_7	15 to 22	27 to 34	Input/ Output	Data Bus: 8-bit, bidirectional data bus
Memory I/O interface signal	$\overline{RD}$	68	82	Output (Tri-state)	Read: Indicates chip is in read cycle. At this time, data bus is in input mode.
	$\overline{WR}$	67	81	Output (Tri-state)	Write: Indicates chip is in write cycle. At this time, data bus is in output mode.
	$\overline{ME}$	65	79	Output (Tri-state)	Memory Enable: Indicates memory read/write is being performed. Goes low in the following cases: a. Instruction fetch and operand read b. Data read/write by memory reference instruction c. Memory access during DMA cycle d. Refresh cycle
	$\overline{IOE}$	64	78	Output (Tri-state)	I/O Enable: Indicates I/O read/write is being performed. Goes low in the following cases: a. Data read/write by I/O instruction execution b. I/O access during DMA cycle c. $\overline{INT_0}$ acknowledge cycle
	$\overline{WAIT}$	13	25	Input	Wait: Used to extend memory and I/O read/write cycles. If low at the falling clock edge in a T_2 or T_W state, a T_W state is inserted next.
System control signals	$\overline{BUSREQ}$	11	23	Input	Bus Request: Used by another device to issue a bus request. When this pin goes low, CPU terminates instruction execution and sets address bus, data bus, and some memory interface signals ($\overline{RD}$, $\overline{WR}$, $\overline{ME}$, $\overline{IOE}$) to high impedance.

Table 2 Pin Functions (cont)

Type	Signal	Pin Number FP-80A	 PC-84	I/O	Name and Pin Function
System control signals	$\overline{\text{BUSACK}}$	12	24	Output	Bus Acknowledge: Indicates receipt of $\overline{\text{BUSREQ}}$ signal by CPU and release of bus. Notifies device that output $\overline{\text{BUSREQ}}$ signal that bus is under its control.
	$\overline{\text{HALT}}$	63	77	Output	HALT: Goes low when CPU executes HALT or SLP instruction to indicate that CPU is in halt or sleep mode. Used together with ST and LIR signals (described below) to indicate operational status of internal DMA and CPU mode.
	$\overline{\text{LIR}}$	66	80	Output	Load Instruction Register: Indicates ongoing cycle is op code fetch cycle.
	ST	62	76	Output	Status: Indicates operational status. Do not connect pull-down resistance to this pin.

ST	$\overline{\text{HALT}}$	$\overline{\text{LIR}}$	Operation Mode
0	1	0	CPU operation (1st op-code fetch cycle)
1	1	0	CPU operation (2nd, 3rd op-code fetch cycle)
1	1	1	CPU operation (machine cycles other than op-code fetch cycles)
0	Undefined	1	DMA
0	0	0	Halt mode
1	0	1	Sleep mode

Table 2 Pin Functions (cont)

		Pin Number			
Type	Signal	FP-80A	PC-84	I/O	Name and Pin Function
System control signals	$\overline{\text{REF}}$	69	83	Output	Refresh: Low level indicates CPU is in DRAM refresh cycle. Refresh address is output on lower 8 bits of address bus (A_0 through A_7). Refresh cycle interval can be programmed to 10, 20, 40, or 80 states.
Interrupt signals	$\overline{\text{NMI}}$	44	57	Input	Non-Maskable Interrupt: Non-maskable interrupt request pin.
	$\overline{\text{INT}_0}$	45	58	Input	Interrupt 0: Maskable interrupt level 0 request pin. Level 0 has 3 modes. **Mode** **Meaning** 0 Execute instruction on data bus 1 Execute instructions starting from address 0038H 2 Vectored
	$\overline{\text{INT}_1}$	48	61	Input	Interrupt 1, 2: Maskable interrupt Level 1 and 2 request pins. Vectored.
	$\overline{\text{INT}_2}$	46	59	Input	
DMA signals	$\overline{\text{DREQ}_0}$	33	46	Input	DMA Request for Channel 0: Asks internal DMAC to perform transfer on channel 0. Enables internal DMAC to synchronize with external I/O device. Internal DMAC channel 0 supports the following transfers. a. Memory ↔ Memory b. Memory ↔ I/O c. Memory ↔ Memory-mapped I/O

HITACHI

Table 2 Pin Functions (cont)

Type	Signal	Pin Number		I/O	Name and Pin Function
		FP-80A	PC-84		
DMA signals	$\overline{TEND_0}$	32	45	Output	Transfer End for Channel 0: Internal DMAC channel 0 transfer end signal. This signal goes low at write cycle of final data transfer.
	$\overline{DREQ_1}$	35	48	Input	DMA Request for Channel 1: Asks internal DMAC to perform transfer on channel 1. Channel 1 supports Memory ↔ I/O transfers only.
	$\overline{TEND_1}$	34	47	Output	Transfer End for Channel 1: Internal DMAC channel 1 transfer end signal. This signal goes low at write cycle of final data transfer.
Serial communications	TX_0	43	56	Output	Transmit Data for SCI Channel 0: SCI channel 0 transmit data pin.
	RX_0	42	55	Input	Receive Data for SCI Channel 0: SCI channel 0 receive data pin.
	SCK_0	41	54	Input/ Output	Serial Clock for SCI Channel 0: SCI channel 0 clock input/output pin.
	TX_1	40	53	Output	Transmit Data for SCI Channel 1: SCI channel 1 transmit data pin.
	RX_1	39	52	Input	Receive Data for SCI Channel 1: SCI channel 1 receive data pin.
	SCK_1	38	51	Input/ Output	Serial Clock for SCI Channel 1: SCI channel 1 clock input/output pin.

Table 2 Pin Functions (cont)

Type	Signal	Pin Number		I/O	Name and Pin Function
		FP-80A	PC-84		
Timers	TIN_1	35	48	Input	Timer Input for Channel 1: Input signal pin for timer 1 input capture. Signal transitions at this pin transfer free-running counter value to input capture register.
	$TOUT_1$	37	50	Output	Timer Output for Channel 1: Timer 1 waveform output pin. When output compare register and free-running counter match, value of OLVL bit of timer control/status register 1 is output from this pin.
	$TOUT_2$	36	49	Output	Timer Output for Channel 2: Timer 2 waveform output pin. When timer 2 time constant register and timer 2 up-counter match, value selected by bit 2 and bit 3 of timer control/status register 2 is output from this pin.
	$TOUT_3$	32	45	Output	Timer Output for Channel 3: Timer 3 waveform output pin. When timer 3 time constant register and timer 3 up-counter match, value selected by bit 2 and bit 3 of timer control/status register 2 is output from this pin.
	$TOUT_4$	34	47	Output	Timer Output for Channel 4: Timer 4 waveform output pin. When timer 4 time constant register and timer 4 up-counter match, value selected by bit 2 and bit 3 of timer control/status register 4 is output from this pin.

Table 2 Pin Functions (cont)

Type	Signal	Pin Number FP-80A	 PC-84	I/O	Name and Pin Function
A/D converter	AV_{SS}	55	69	Input	Analog Ground: A/D converter power supply (ground)
	AV_{CC}	50	63	Input	Analog Power Supply: A/D converter power supply (+5 V). Connect to system power supply (V_{CC}).
	AN_0 to AN_3	51 to 54	65 to 68	Input	Analog Input: A/D converter input signal pins
	$\overline{ADT}$	48	61	Input	A/D Trigger: External trigger input pin to start A/D converter
EEPROM	V_{PP}	49	62	Output	Test pin. Do not connect.
Parallel I/O	$P0_0$ to $P0_7$	23 to 30	35 to 42	Input/ Output	Port 0: 8-bit parallel I/O port. Switched between input and output by data direction register 0 (DDR0).
	$P1_0$ to $P1_7$	36 to 43	49 to 56	Input/ Output	Port 1: 8-bit parallel I/O port. Switched between input and output by data direction register 1 (DDR1).
	$P2_0$ to $P2_6$	6 to 9 11, 12, 69	17 to 20 23, 24, 83	Input/ Output	Port 2: 7-bit parallel I/O port. Switched between input and output by data direction register 2 (DDR2). P26 is output only.
	$P3_0$ to $P3_7$	32 to 35 45 to 48	45 to 48 58 to 61	Input/ Output	Port 3: 8-bit parallel I/O port. Switched between input and output by data direction register 3 (DDR3).
	$P4_0$ to $P4_3$	51 to 54	65 to 68	Input	Port 4: 4-bit input-only port.

■ Basic CPU Architecture

The HD648180W has an advanced, high-speed, eight-bit CPU. Its instruction set is upward-compatible with the HD64180Z.

• Features

The HD648180W CPU has the following features.

- CPU architecture based on the Z-80
- Expanded instruction set with instructions for:

  — Input from on-chip I/O
  — Output to on-chip I/O
  — Block transfer between memory and on-chip I/O
  — 8-bit × 8-bit multiplication
  — AND operations on on-chip I/O
  — AND operations on accumulator A
  — Transition to sleep mode

- Eight addressing modes

  — Implied
  — Register direct
  — Register indirect
  — Indexed
  — Extended
  — Immediate
  — Relative
  — I/O

- Maximum operating speed: 6.144 MHz
- Special CPU operating modes

  — Low-power modes (standby modes)
  Software standby mode
  Hardware standby mode
  — Sleep mode
  — Halt mode
  — Bus-release mode

- **Address Space**

The HD648180W CPU has a 64-kbyte logical address space and 64-kbyte I/O address space.

The 64-kbyte logical address space is mapped into a 1-Mbyte physical address space by the memory management unit (MMU). For details, see Memory Management Unit (MMU).

The 64-kbyte I/O address space is assigned to on-chip and off-chip I/O. For details, see On-Chip I/O Address Space Map.

- **Register Configuration**

The CPU includes two general register sets (GR and GR'), and a single special-purpose register set. Register sets GR and GR' each include an accumulator, a flag register, and six general-purpose registers. The special-purpose register set consists of an interrupt vector register, an R counter, two 16-bit index registers, a stack pointer, and a program counter.

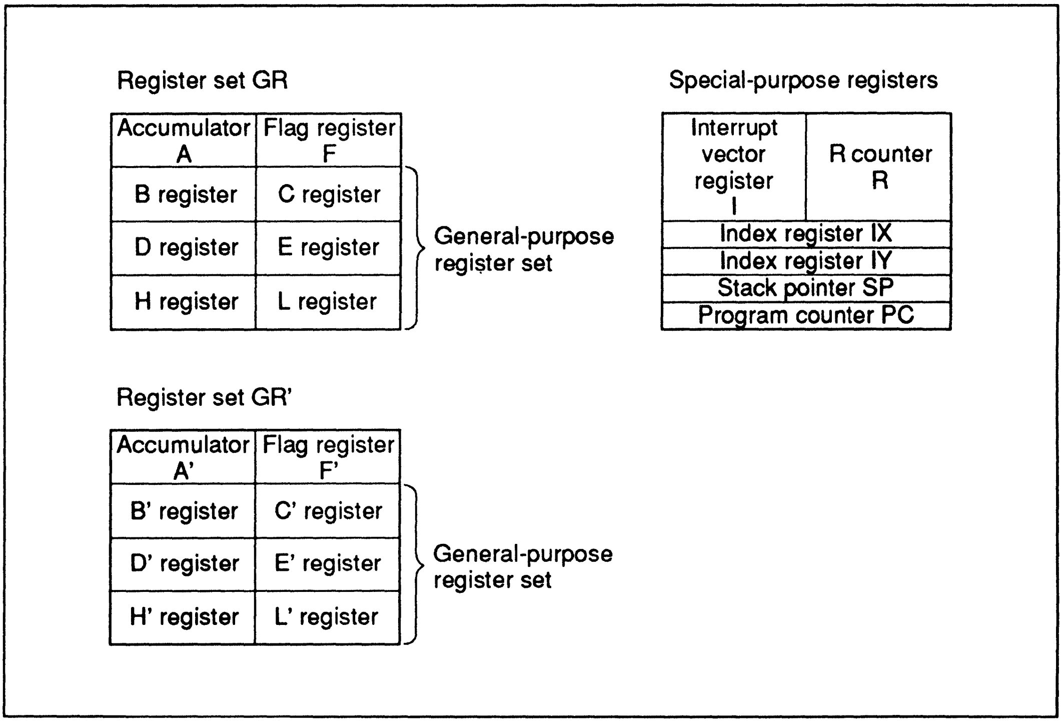


Figure 5 CPU Register Configuration

• Register Descriptions

The following table describes the function of each register. Unless otherwise specified, register content is undefined following a reset.

Register Name	Symbol	Function
Accumulators	A, A'	Accumulators are registers in which 8-bit arithmetic operations, logical operations, and shifts are performed. Accumulator A' is used in place of accumulator A following execution of the instruction EX AF, AF'.
Flag registers	F, F'	Flag registers store the status of operations. Flag register F' is used in place of flag register F following execution of the instruction EX AF, AF'.
General-purpose registers	B, C D, E H, L	The registers in register set GR can be used as 8-bit registers (B, C, D, E, H, L) or 16-bit registers (BC, DE, HL). They can be used to execute operations or store addresses.
	B', C' D', E' H', L'	The registers in register set GR' supplement the GR registers. The two register sets can be swapped using the EXX instruction.
Interrupt vector register	I	This 8-bit register stores the upper 8 bits of the 16-bit vector produced by an interrupt. Vectors are used for $\overline{INT_0}$ mode 2, $\overline{INT_1}$, $\overline{INT_2}$, and internal interrupts (7 sources). A reset initializes this register to 00H.
R counter	R	This 8-bit register counts the number of op-code fetch cycles performed. The content of this counter is unrelated to the refresh address, which is generated by another on-chip counter that cannot be accessed by the user. A reset initializes this register to 00H.
Index registers	IX, IY	These 16-bit registers are used for indexed addressing and 16-bit operations. In indexed addressing, the base address is stored in the index registers. A signed 8-bit displacement is added to the base address to generate the operand's effective address. In 16-bit operations that involve the index registers, a general-purpose register (except HL), the index registers, or the stack pointer can be used for the other operand.
Stack pointer	SP	This 16-bit register stores the top address of the stack area. A reset initializes this register to 00H.
Program counter	PC	This 16-bit register stores the logical address of the next instruction to be executed. This register is normally incremented by 1 each time a 1-byte instruction code is accessed, but execution of a jump instruction causes the address of the jump destination to replace the current content of the program counter. A reset initializes this register to 0000H.

- Flag Register

The flag register (F) contains individual flags that are set and reset to represent the status of the result of an 8-bit or 16-bit operation. This register is referenced by extended arithmetic instructions and conditional jump instructions.

	7	6	5	4	3	2	1	0
Flag register F	S	Z	—	H	—	P/V	N	C

Flag Name	Symbol	Function
Sign	S	Bit 7 is set when an operation produces a negative result (MSB = 1) and reset by a positive result (MSB = 0).
Zero	Z	Bit 6 is set when an instruction execution produces a zero result, and reset by any other result.
Half-carry	H	Bit 5 is set when an operation results in a carry or borrow from the 4th bit. The half-carry flag is reset when neither a carry nor borrow is generated. This flag is referenced for adjustment of decimal operations (DAA instruction).
Parity/overflow	P/V	Bit 2 can be used as either a parity or an overflow flag. The parity flag indicates the parity of the data stored in the accumulator following execution of a logical operation. An even number of 1s in the accumulator value sets the parity flag, while an odd number of 1s resets it. The overflow flag is set when the result of a signed arithmetic operation exceeds the range of +127 through –128 for an 8-bit operation, or the range of +32767 through –32768 for a 16-bit operation. Results within these ranges reset the overflow flag.
Negate	N	Bit 1 is set by execution of a subtraction instruction (examples: SUB, DEC, CP) and reset by an addition instruction (examples: ADD, INC).
Carry	C	Bit 0 is set when an operation results in a carry or borrow from the most significant bit. Any other result resets this flag. The following lists the various types of carries and borrows. • Carry produced by an addition result • Borrow produced by a subtraction result • Carry produced by a shift or rotate

Addressing Modes

The CPU instruction set has eight addressing modes.

1. Implied Addressing (IMP)

   This addressing mode is based on data included within op codes. It is used by instructions that manipulate bit positions specified by the accumulator (A), index registers (IX, IY), stack pointer (SP), HL general-purpose register, or op codes.

2. Register Direct Addressing (REG)

   In this addressing mode, 8-bit and 16-bit registers are specified by op code fields g, g', ww, xx, yy, and zz. The following tables show the relationships between field codes and registers.

8-Bit Register Specification

g or g' Field	Specified Register
000	B
001	C
010	D
011	E
100	H
101	L
110	—
111	A

16-bit Register Specification

ww Field	Specified Register
00	BC
01	DE
10	HL
11	SP

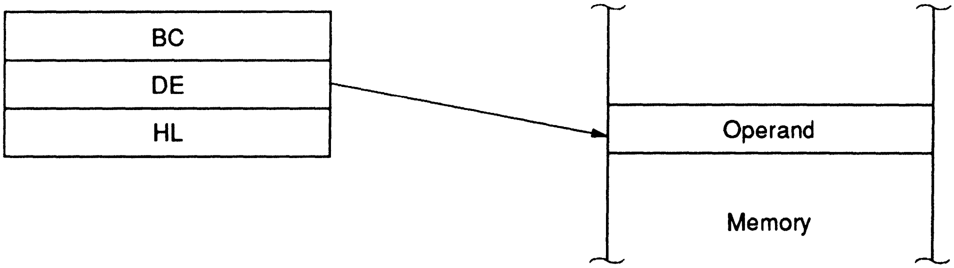
xx Field	Specified Register
00	BC
01	DE
10	IX
11	SP

yy Field	Specified Register
00	BC
01	DE
10	IY
11	SP

zz Field	Specified Register
00	BC
01	DE
10	HL
11	AF

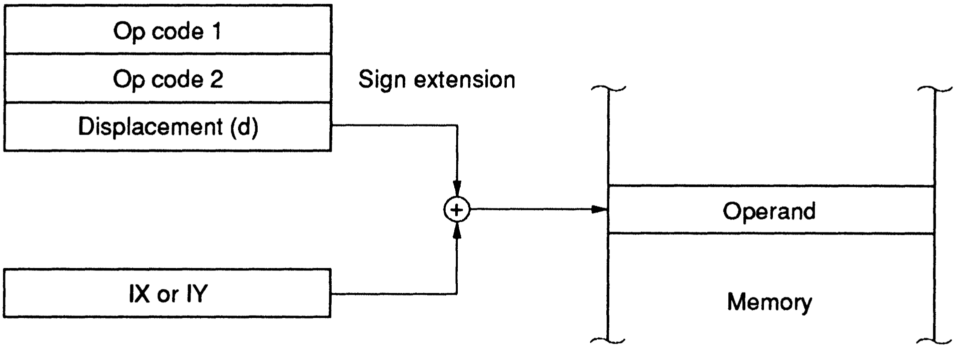
3. Register Indirect Addressing (REGI)

In this addressing mode, general-purpose registers are used as 16-bit address registers for specification of operands in memory.



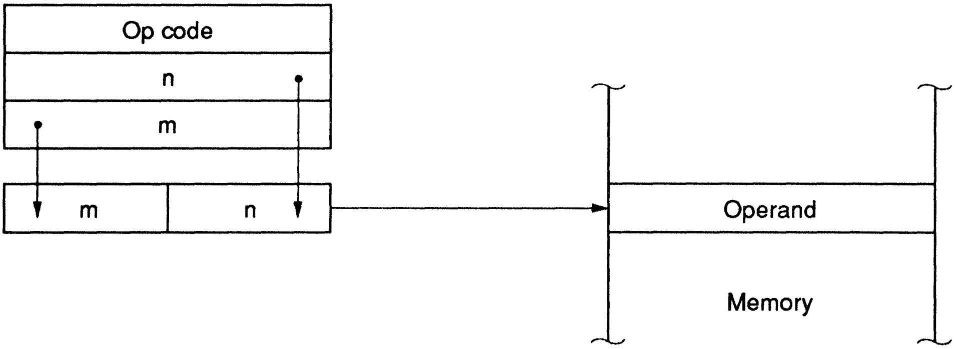
4. Indexed Addressing (INDX)

In this addressing mode, effective addresses are generated by adding a displacement (d: signed 8-bit value) to the index registers (IX, IY).



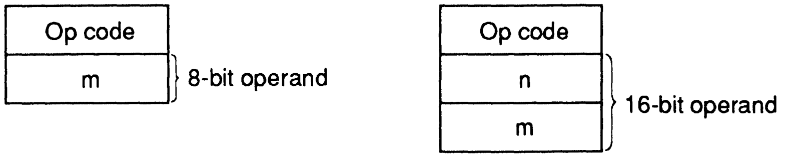
5. Extended Addressing (EXT)

With this addressing mode, 2 bytes (m, n) following the op code are used as a 16-bit address for direct specification of an operand in memory.



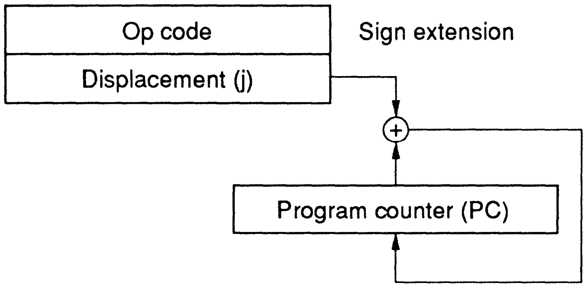
6. Immediate Addressing (IMMED)

With this addressing mode, 1 byte (m) or 2 bytes (m, n) following the op code are used as a direct operand.



7. Relative Addressing (REL)

This addressing mode can be used with branching instructions only. A displacement (j: signed 8-bit value) is added to the program counter (PC) to generate a branch address. In the case of a conditional branch, the branch is taken only when a condition is satisfied.



8. I/O Addressing

This addressing mode can be used with I/O instructions only. The specified address is an I/O address ($\overline{\text{IOE}}$ = 0). One of the following addresses is output by this addressing mode.

- An address formed by the content of the operand, which is output to A_0 through A_7, and the content of accumulator A, which is output to A_8 through A_{15}.
- An address formed by the content of register C, which is output to A_0 through A_7, and the content of register B, which is output to A_8 through A_{15}.
- An address formed by the content of the operand, which is output to A_0 through A_7, and 00H, which is output to A_8 through A_{15}. This is convenient for accessing on-chip I/O registers.
- An address formed by the content of register C, which is output to A_0 through A_7, and 00H, which is output to A_8 through A_{15}. This is convenient for accessing on-chip I/O registers.

Instruction Set Overview

The CPU instruction set for this chip can be broken down as follows.

- Operation instructions
- Load instructions
- Program control instructions
- I/O instructions
- Special control instructions

These instructions consist of 1 to 4 bytes. Typical examples are shown below.

	7	6	5	4	3	2	1	0	
1-byte instruction	0	1	g			g'			LD g, g'

	7	6	5	4	3	2	1	0	
2-byte instruction	0	0	g			1	1	0	LD g, m
	m								Immediate data

	7	6	5	4	3	2	1	0	
3-byte instruction	1	1	0	1	1	1	0	1	LD g, (IX + d)
	0	1	g			1	1	0	
	d								Displacement

	7	6	5	4	3	2	1	0	
4-byte instruction	1	1	0	1	1	1	0	1	LD (IX + d), m
	0	0	1	1	0	1	1	0	
	d								Displacement
	m								Immediate data

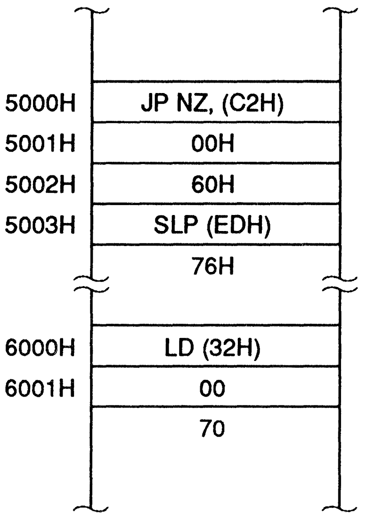
See Instruction Set Lists for further details.

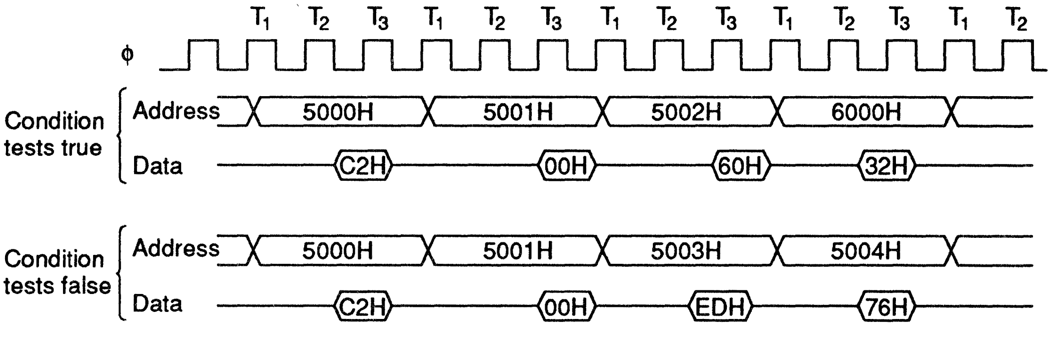
Conditional Jump and Conditional Call Instruction Precautions

The following illustrates operation when the conditional jump instruction JP f, mn is executed. Note the difference in operation when the condition tests true and when it tests false.

Example

When the instruction JP NZ, 6000H is at address 5000H (MMU sets base register to 00H).




T1 T2 T3 T1 T2 T3 T1 T2 T3 T1 T2 T3 T1 T2
ϕ
Condition tests true
Address 5000H 5001H 5002H 6000H
Data C2H 00H 60H 32H
Condition tests false
Address 5000H 5001H 5003H 5004H
Data C2H 00H EDH 76H

Note that when the condition tests false, execution proceeds to the next instruction without reading the 2nd operand of JP f, mm. This is also true in the case of the conditional call: CALL f, mn.

■ Absolute Maximum Ratings

Item	Symbol	Rating	Unit
Analog supply voltage	AV_{CC}	–0.3 to +7.0	V*1
Input voltage (port 4)	$V_{in}1$	–0.3 to AV_{CC} + 0.3	V
Analog input voltage	A_{in}	–0.3 to AV_{CC} + 0.3	V
Total current inflow	ΣI_{OL}	80	mA*2
Total current outflow	$\Sigma \lvert I_{OH} \rvert$	30	mA*3

Notes: 1. AV_{CC} must equal V_{CC}.

2. Total current inflow is the total current sunk simultaneously from all input/output lines via output buffers to the HD648180W's V_{SS}.

3. Total current outflow is the total current sourced simultaneously from the HD648180W's V_{CC} via output buffers to all input/output pins.

■ Electrical Characteristics

DC Characteristics (V_{CC} = 5 V ±10%, V_{SS} = 0 V, T_a = –20 to +75°C unless otherwise noted)

Item	Symbol	Test Conditions	Min	Typ	Max	Unit
Input high voltage level for $\overline{\text{RESET}}$, EXTAL, $\overline{\text{NMI}}$, $\overline{\text{STBY}}$	V_{IH1}		V_{CC} – 0.6	—	V_{CC} + 0.3	V
Input high voltage level for other pins	V_{IH2}		2.0	—	V_{CC} + 0.3	V
Input low voltage level for $\overline{\text{RESET}}$, EXTAL, $\overline{\text{NMI}}$, $\overline{\text{STBY}}$	V_{IL1}		–0.3	—	0.6	V
Input low voltage level for other pins	V_{IL2}		–0.3	—	0.8	V
Output high voltage level for all output pins	V_{OH}	I_{OH} = –200 µA	2.4	—	—	V
		I_{OH} = –20 µA	V_{CC} – 1.2	—	—	
Output low voltage level for all output pins	V_{OL}	I_{OL} = 2.2 mA	—	—	0.45	V
Input leakage current for all input pins except XTAL and EXTAL	I_{IL}	V_{in} = 0.5 to V_{CC} – 0.5	—	—	1.0	µA
Tri-state leakage current	I_{TL}	V_{in} = 0.5 to V_{CC} – 0.5	—	—	1.0	µA
Current consumption (normal operation)Note	I_{CC}	f = 4 MHz	—	15	30	mA
		f = 6 MHz	—	20	40	mA
Current consumption (standby mode)Note	I_{CCSTBY}	—	—	10	20	µA
Pin capacitance	C_P	V_{in} = 0 V, f = 1 MHz T_a = 25°C	—	—	15	pF

Note: Signal levels:
$\overline{\text{RESET}}$, EXTAL, $\overline{\text{NMI}}$: V_{IH}min = V_{CC} – 0.6 V, V_{IL}max = 0.6 V
Other pins: V_{IH}min = V_{CC} – 1.0 V, V_{IL}max = 0.8 V
All output pins are unloaded.

- **DC Characteristics (I/O ports) (V_{cc}=5V ± 10%, V_{SS}=0 V, T_a=–20 to +75°C unless otherwise noted)**

Item		Symbol	Test Conditions	Min	Typ	Max	Unit
Input high voltage	Ports 0 to 4	V_{IHP}		2.0	—	V_{CC} + 0.3	V
Input low voltage	Ports 0 to 4	V_{ILP}		–0.3	—	0.8	V
Output high voltage	Ports 0 to 3	V_{OHP}	I_{OH} = –200 µA	2.4	—	—	V
			I_{OH} = –20 µA	V_{CC} – 1.2	—	—	
Output low voltage	Ports 0 to 3	V_{OLP}	I_{OL} = 2.2 mA	—	—	0.45	V
	Port 0		I_{OL} = 10 mA	—	—	1.0	
Input leakage current		$\|I_{ILP}\|$	V_{in} = 0.5 to V_{CC} – 0.5	—	—	1.0	µA

- **A/D Conversion Characteristics (AV_{CC}=V_{CC}, V_{SS}=0 V, T_a=–20 to +75°C unless otherwise noted)**

Item	Symbol	Test Conditions	Min	Typ	Max	Unit
Analog input voltage	AV_{in}		AV_{SS}	—	AV_{CC}	V
Analog input leakage current	$\|I_{AL}\|$		—	—	1.0	µA
Resolution			—	8	—	Bit
Input hold time (conversion time)		A/D conversion clock = φ/3 = 2.048 MHz when φ = 6.144 MHz	16.6	—	—	µs
Number of inputs			0	—	4	Channel
Absolute precision		T_a = 25°C, V_{CC} = 5.0 V	—	—	2.0	LSB

Note: AV_{CC} must equal V_{CC}.

AC Characteristics (V_{CC} = 5 V ±10%, V_{SS} = 0 V, T_a = –20 to +75°C unless otherwise noted)

			HD648180W-4			HD648180W-6			
Item	**Symbol**	**Test Conditions**	**Min**	**Typ**	**Max**	**Min**	**Typ**	**Max**	**Unit**
Clock cycle time	t_{cyc}	Figure 6	250	—	333	162	—	333	ns
Clock high pulse duration	t_{CHW}		110	—	—	65	—	—	ns
Clock low pulse duration	t_{CLW}		110	—	—	65	—	—	ns
Clock fall time	t_{cf}		—	—	15	—	—	15	ns
Clock rise time	t_{cr}		—	—	15	—	—	15	ns
External clock cycle time	t_{ECYC}	Figure 18	125	—	—	81	—	—	ns
External clock high pulse duration	t_{EXHW}		50	—	—	32	—	—	ns
External clock low pulse duration	t_{EXLW}		50	—	—	32	—	—	ns
External clock rise time	t_{EXr}*1	Figure 18	—	—	25	—	—	25	ns
External clock fall time	t_{EXf}*1		—	—	25	—	—	25	ns
Address delay time	t_{AD}	Figure 6	—	—	110	—	—	90	ns
Address setup time (measured from $\overline{ME}$ or $\overline{IOE}$ falling edge)	t_{AS}		50	—	—	30	—	—	ns
$\overline{ME}$ delay time 1	t_{MED1}	Figure 6, 7,	—	—	85	—	—	60	ns
$\overline{RD}$ delay time 1 (when $\overline{IOC}$ = 1)	t_{RDD1}	Figure 6	—	—	85	—	—	60	ns
(when $\overline{IOC}$ = 0)		Figure 8	—	—	85	—	—	65	ns
$\overline{LIR}$ delay time 1	t_{LD1}	Figure 6	—	—	100	—	—	80*2	ns
Address hold time (measured from $\overline{ME}$, $\overline{IOE}$, $\overline{RD}$, or $\overline{WR}$ rising edge)	t_{AH}		80	—	—	35	—	—	ns
$\overline{ME}$ delay time 2	t_{MED2}	Figure 6, 7	—	—	85	—	—	60	ns
$\overline{RD}$ delay time 2	t_{RDD2}	Figure 6	—	—	85	—	—	60	ns
$\overline{LIR}$ delay time 2	t_{LD2}		—	—	100	—	—	80*2	ns

Notes: 1. If external clock pulse duration specifications (t_{EXHW}, t_{EXLW}) are not satisfied, adjust t_{EXr} and t_{EXf}.

2. t_{LD1} (t_{LD2}) = 60 ns when bus timing test load capacitance C = 40 pF.

AC Characteristics (V_{CC} = 5 V ±10%, V_{SS} = 0 V, T_a = –20 to +75°C unless otherwise noted) (cont)

			HD648180W-4			HD648180W-6			
Item	**Symbol**	**Test Conditions**	**Min**	**Typ**	**Max**	**Min**	**Typ**	**Max**	**Unit**
Data read setup time	t_{DRS}	Figure 6, 7	50	—	—	40	—	—	ns
Data read hold time	t_{DRH}		0	—	—	0	—	—	ns
ST delay time 1	t_{STD1}	Figure 6 Figure 10	—	—	110	—	—	90	ns
ST delay time 2	t_{STD2}		—	—	110	—	—	90	ns
$\overline{\text{WAIT}}$ setup time	t_{WS}	Figure 6	80	—	—	40	—	—	ns
$\overline{\text{WAIT}}$ hold time	t_{WH}		70	—	—	40	—	—	ns
Write data floating delay time	t_{WDZ}	Figure 6	—	—	100	—	—	95	ns
$\overline{\text{WR}}$ delay time 1	t_{WRD1}		—	—	90	—	—	65	ns
Write data delay time	t_{WDD}		—	—	110	—	—	90	ns
Write data setup time (measured from WR falling edge)	t_{WDS}		60	—	—	40	—	—	ns
$\overline{\text{WR}}$ delay time 2	t_{WRD2}		—	—	90	—	—	80	ns
$\overline{\text{WR}}$ pulse duration	t_{WRP}		280	—	—	170	—	—	ns
Write data hold time (measured from $\overline{\text{WR}}$ rising edge)	t_{WDH}		60	—	—	40	—	—	ns
$\overline{\text{IOE}}$ delay time 1 (when $\overline{\text{IOC}}$ = 1)	t_{IOD1}	Figure 6	—	—	85	—	—	60	ns
(when $\overline{\text{IOC}}$ = 0)		Figure 8	—	—	85	—	—	65	ns
$\overline{\text{IOE}}$ delay time 2	t_{IOD2}	Figure 6	—	—	85	—	—	60	ns
$\overline{\text{IOE}}$ delay time 3 (measured from $\overline{\text{LIR}}$ falling edge)	t_{IOD3}	Figure 7	540	—	—	340	—	—	ns

AC Characteristics (V_{CC} = 5 V ±10%, V_{SS} = 0 V, T_a = –20 to +75°C unless otherwise noted) (cont)

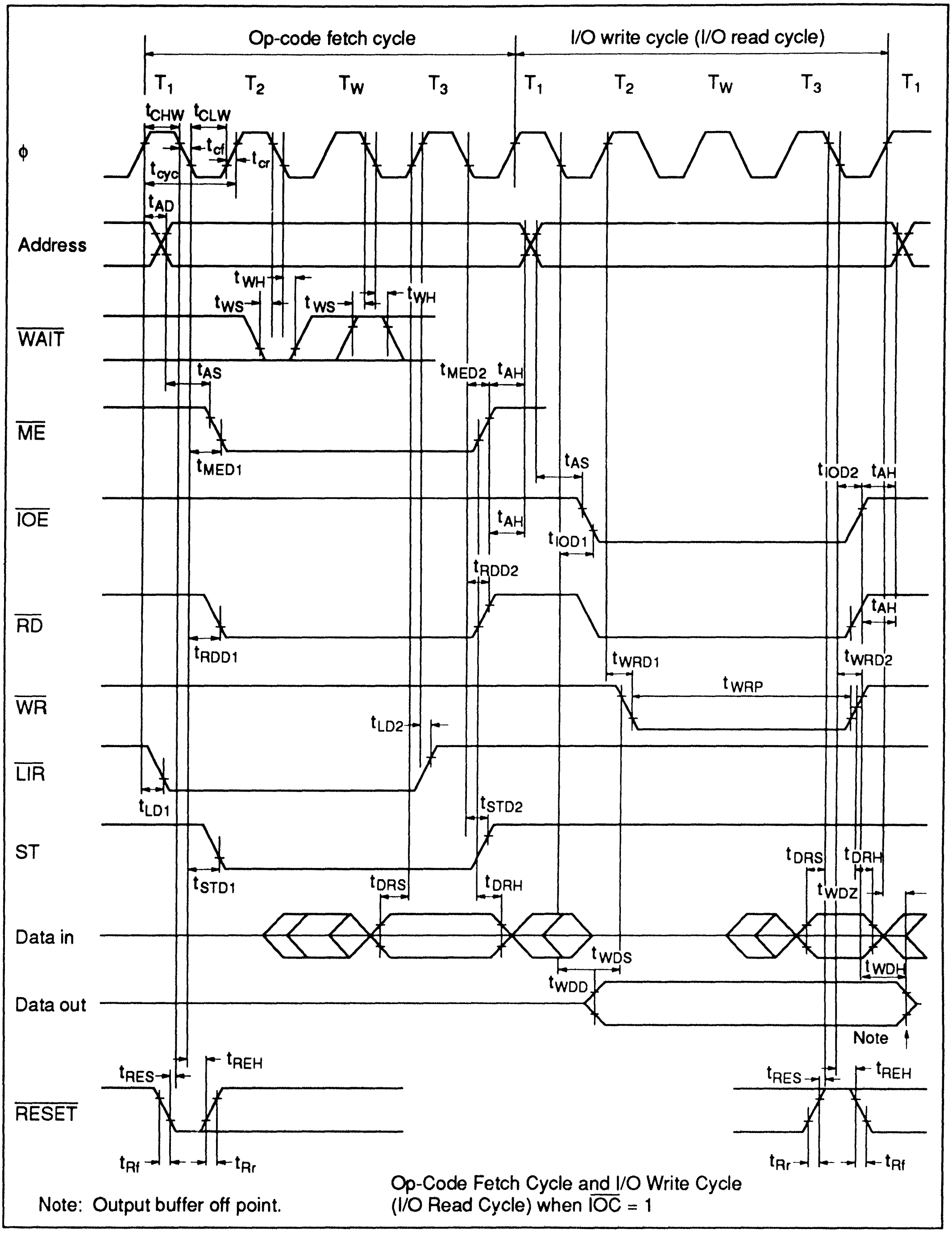
			HD648180W-4			HD648180W-6			
Item	**Symbol**	**Test Conditions**	**Min**	**Typ**	**Max**	**Min**	**Typ**	**Max**	**Unit**
$\overline{\text{INT}}$ setup time (measured from φ falling edge)	t_{INTS}	Figure 7 Figure 13	80	—	—	40	—	—	ns
$\overline{\text{INT}}$ hold time (measured from φ falling edge)	t_{INTH}		70	—	—	40	—	—	ns
$\overline{\text{NMI}}$ pulse duration	t_{NMIW}		120	—	—	120	—	—	ns
$\overline{\text{BUSREQ}}$ setup time (measured from φ falling edge)	t_{BRS}	Figure 7	80	—	—	40	—	—	ns
$\overline{\text{BUSREQ}}$ hold time (measured from φ falling edge)	t_{BRH}		70	—	—	40	—	—	ns
$\overline{\text{BUSACK}}$ delay time 1	t_{BAD1}		—	—	100	—	—	95	ns
$\overline{\text{BUSACK}}$ delay time 2	t_{BAD2}		—	—	100	—	—	95	ns
Bus floating delay time	t_{BZD}		—	—	130	—	—	125	ns
$\overline{\text{ME}}$ pulse duration (high)	t_{MEWH}		200	—	—	110	—	—	ns
$\overline{\text{ME}}$ pulse duration (low)	t_{MEWL}		210	—	—	125	—	—	ns
Port data output delay time	t_{PWD}	Figure 13	—	—	110	—	—	90	ns
Port data input setup time	t_{PDS}	Figure 13	180	—	—	150	—	—	ns
Port data input hold time	t_{PDH}	Figure 13	60	—	—	40	—	—	ns
$\overline{\text{REF}}$ delay time 1	t_{RFD1}	Figure 7	—	—	110	—	—	90	ns
$\overline{\text{REF}}$ delay time 2	t_{RFD2}		—	—	110	—	—	90	ns
$\overline{\text{HALT}}$ delay time 1	t_{HAD1}	Figure 7	—	—	110	—	—	90	ns
$\overline{\text{HALT}}$ delay time 2	t_{HAD2}	Figure 12	—	—	110	—	—	90	ns
$\overline{\text{DREQi}}$ setup time 1	t_{DRQS}	Figure 10	80	—	—	40	—	—	ns
$\overline{\text{DREQi}}$ hold time 1	t_{DRQH}		70	—	—	40	—	—	ns
$\overline{\text{TENDi}}$ delay time 1	t_{TED1}		—	—	85	—	—	70	ns
$\overline{\text{TENDi}}$ delay time 2	t_{TED2}		—	—	85	—	—	70	ns

AC Characteristics (V_{CC} = 5 V ±10%, V_{SS} = 0 V, T_a = –20 to +75°C unless otherwise noted) (cont)

			HD648180W-4			HD648180W-6			
Item	**Symbol**	**Test Conditions**	**Min**	**Typ**	**Max**	**Min**	**Typ**	**Max**	**Unit**
Timer output delay time	t_{TOD}	Figure 11	—	—	300	—	—	300	ns
SCI input clock cycle (clocked synchronous mode)	t_{scyc}	Figure 17	40	—	—	40	—	—	t_{cyc}
SCI transmit data delay time (clocked synchronous mode)	t_{TXD}		—	—	200	—	—	200	ns
SCI receive data setup time (clocked synchronous mode)	t_{SRX}	Figure 17	260	—	—	260	—	—	ns
SCI receive data hold time (clocked synchronous mode)	t_{HRX}		100	—	—	100	—	—	ns
SCI input clock pulse duration	t_{PWSCXH} t_{PWSCXL}	Figure 14	0.4*1	—	0.6*1	0.4*1	—	0.6*1	t_{scyc}
Timer 1 input clock pulse duration	t_{PWTH} t_{PWTL}	Figure 15	8	—	—	8	—	—	t_{cyc}
$\overline{\text{RESET}}$ setup time	t_{RES}	Figure 15	120	—	—	120	—	—	ns
$\overline{\text{RESET}}$ hold time	t_{REH}		80	—	—	80	—	—	ns
$\overline{\text{RESET}}$ rise time	t_{Rr}		—	—	50*2	—	—	50*2	ms
$\overline{\text{RESET}}$ fall time	t_{Rf}		—	—	50*2	—	—	50*2	ms
Input pin rise time (other than $\overline{\text{RESET}}$)	t_{Ir}	Figure 19	—	—	100*2	—	—	100*2	ns
Input pin fall time (other than $\overline{\text{RESET}}$)	t_{If}		—	—	100*2	—	—	100*2	ns
$\overline{\text{ADT}}$ input pulse duration	t_{PWADH} t_{PWADL}	Figure 15	8	—	—	8	—	—	t_{cyc}
$\overline{\text{STBY}}$ input delay duration	t_{STBYD}	Figure 6	0	—	—	0	—	—	ns
Oscillation stabilization time	t_{OSC}	Figure 9	—	—	20	—	—	20	ms

Notes: 1. Set so: (t_{PWSCXH}) + (t_{PWSCXL}) + (t_{IR}) + (t_{IF}) = t_{SCYC}.

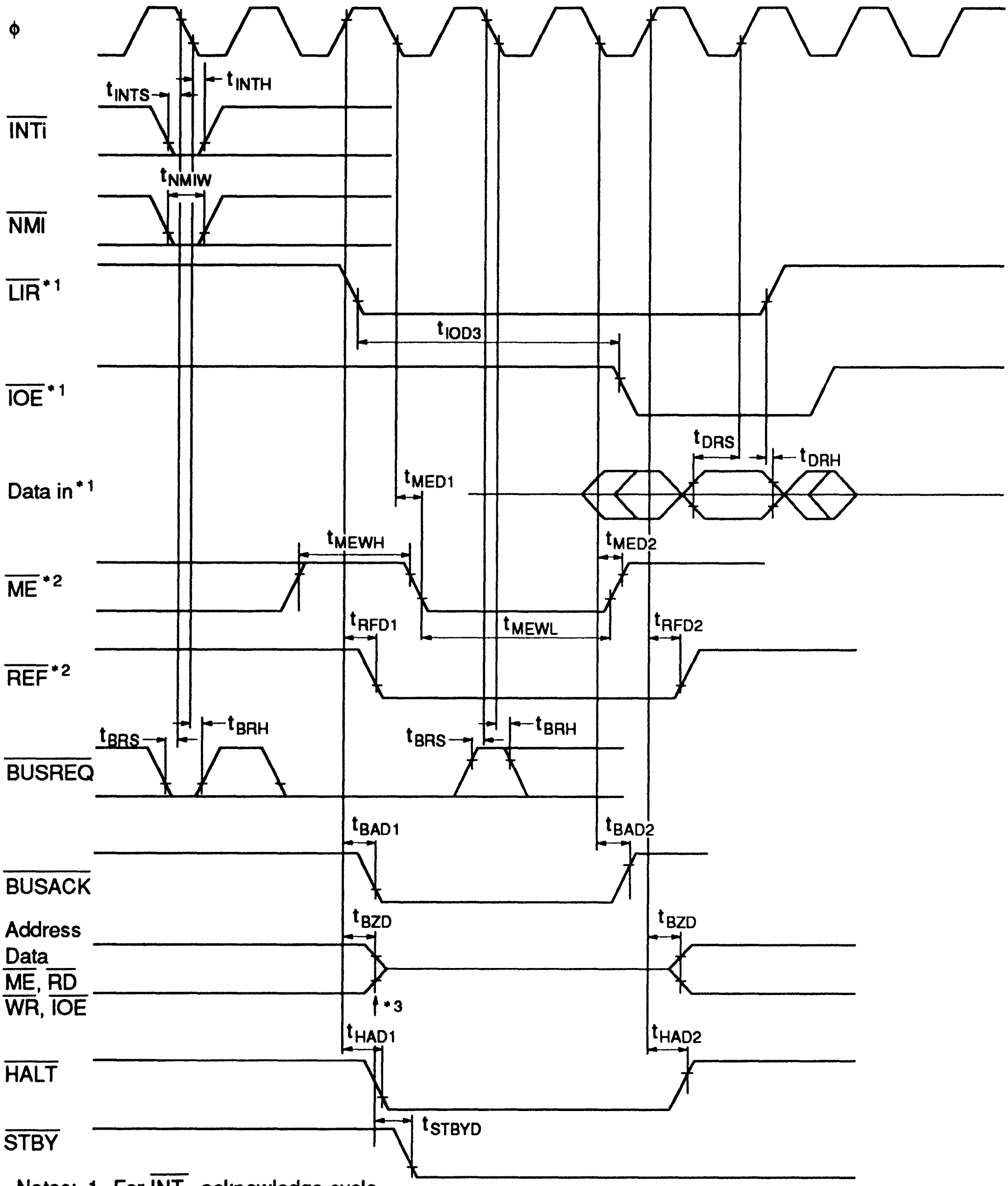
2. If these rise and fall times are satisfied, but other specifications are not satisfied, adjust these rise and fall times to satisfy the other specifications.



Note: Output buffer off point.

Op-Code Fetch Cycle and I/O Write Cycle (I/O Read Cycle) when $\overline{IOC}$ = 1

Figure 6 CPU Timing (1)



Notes: 1. For $\overline{INT_0}$ acknowledge cycle
2. For refresh cycle
3. Output buffer off point

$\overline{INT_0}$ Acknowledge Cycle, Refresh Cycle, Bus Release Mode, Halt Mode, sleep Mode, and System Stop Mode

Figure 7 CPU Timing (2)

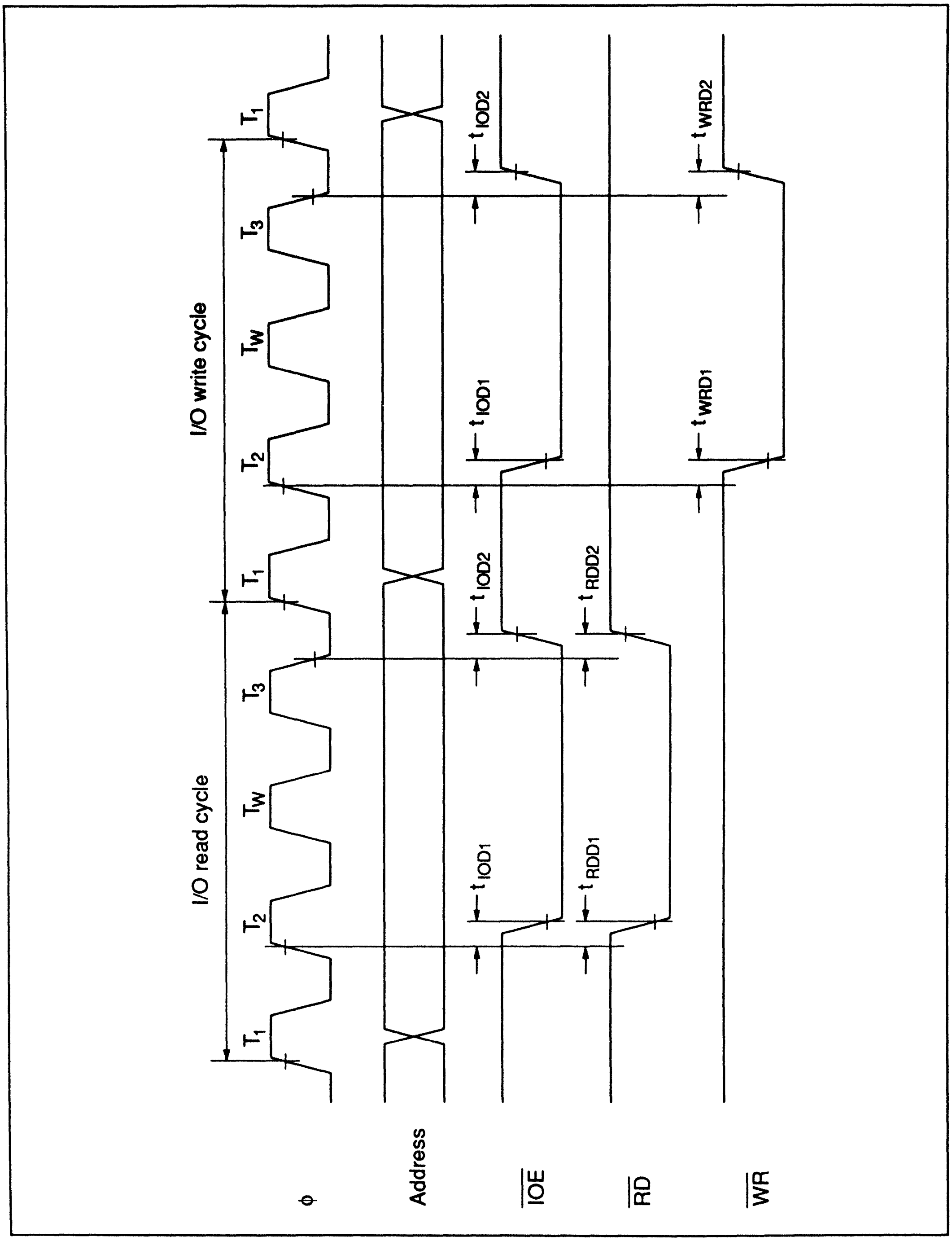


Figure 8 CPU Timing (3)

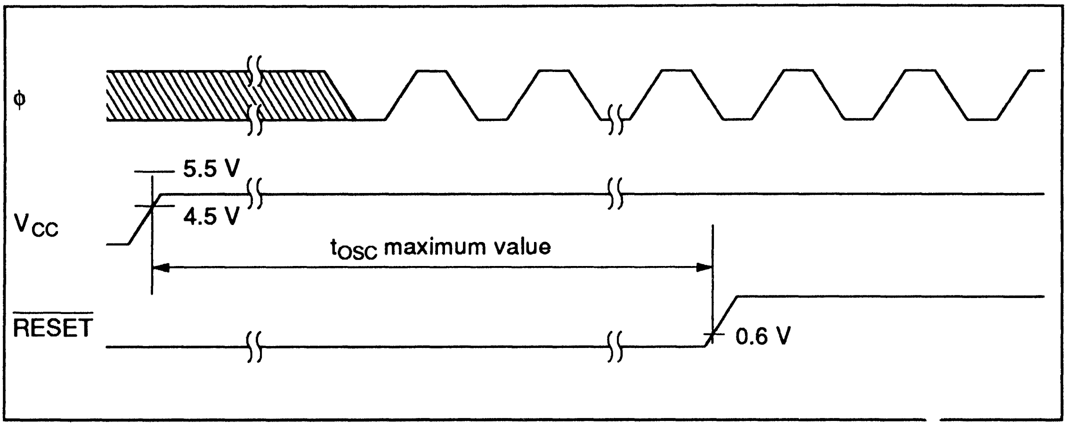
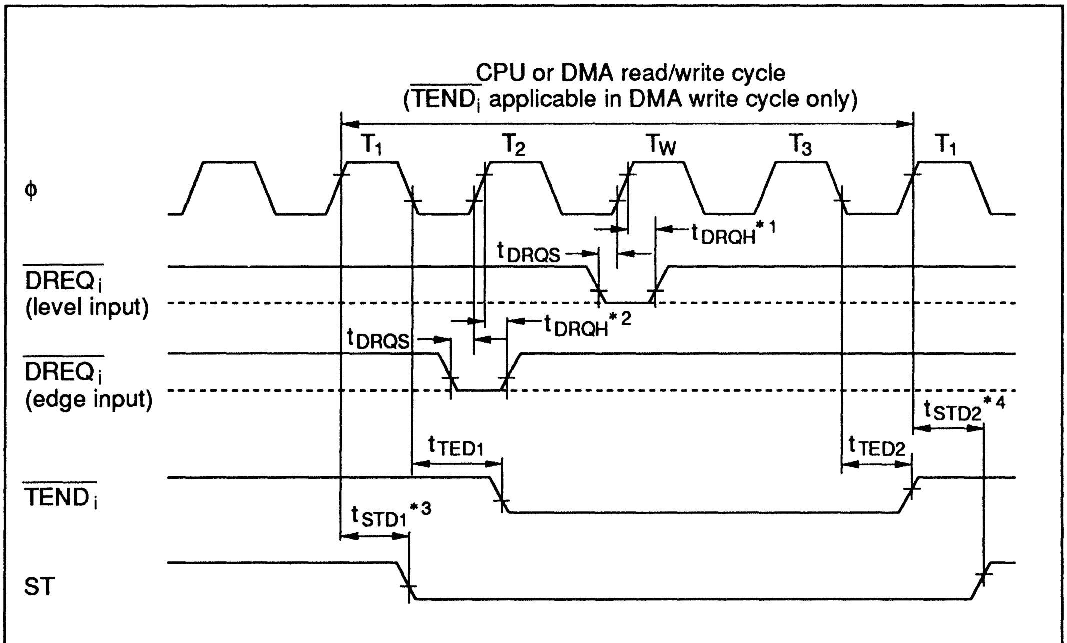


Figure 9 CPU Timing (4)



Notes: 1. Prescribed for clock rising edge immediately before T_3
2. Prescribed for each clock rising edge
3. T_1 of DMA cycle start
4. T_1 of CPU cycle start

Figure 10 DMA Control Signals

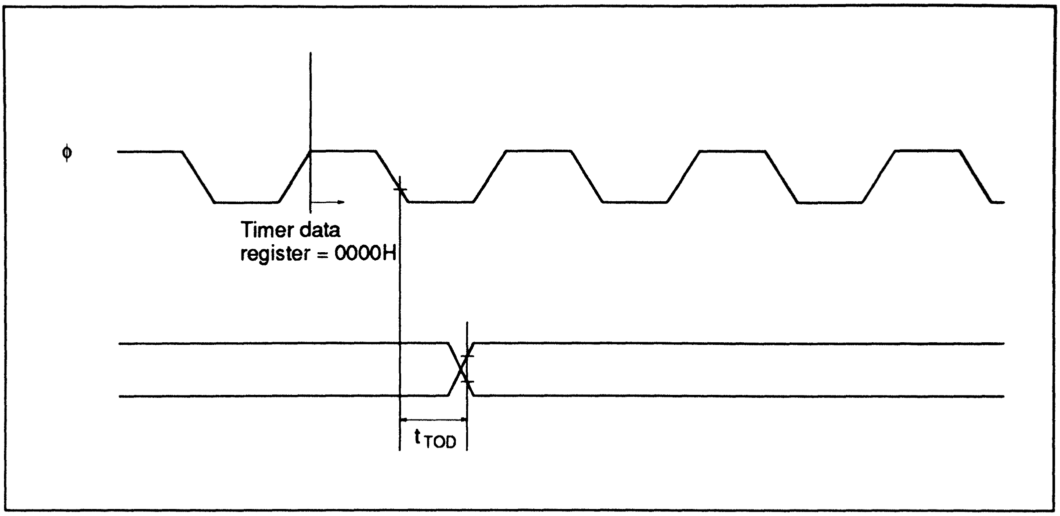


Figure 11 Timer Output

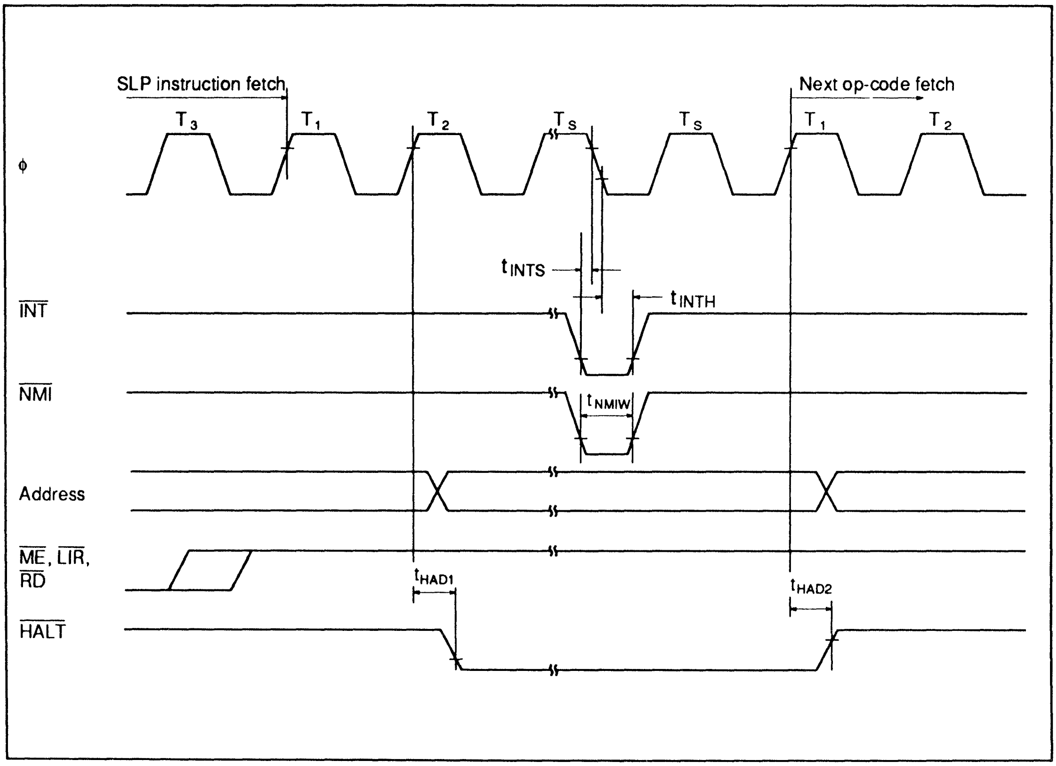


Figure 12 SLP Execution Cycle

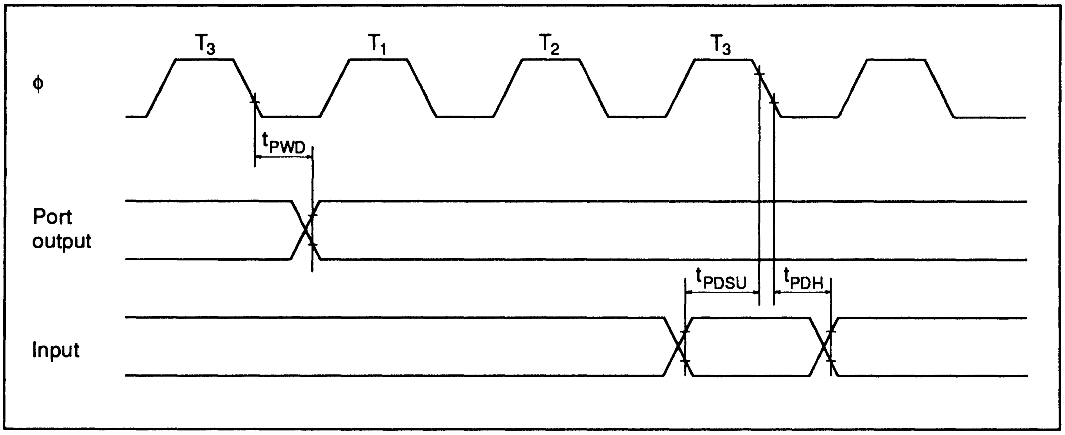


Figure 13 Port I/O Timing

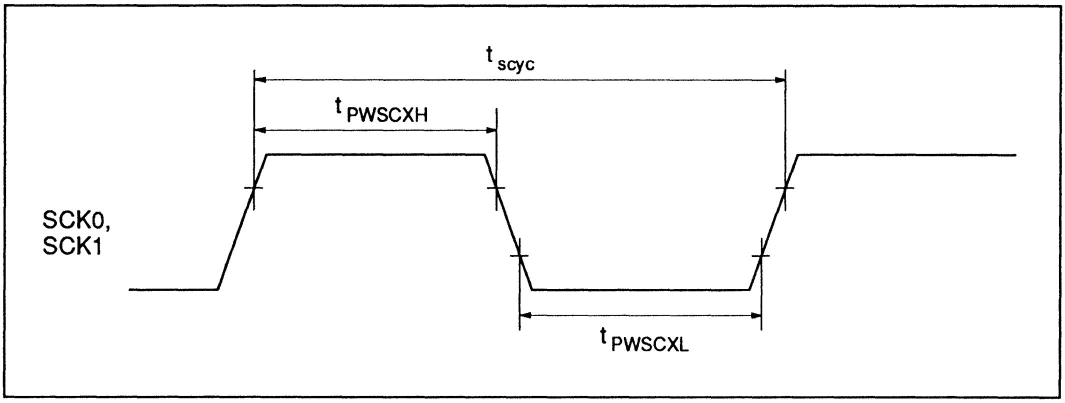


Figure 14 SCI Clock Input Timing

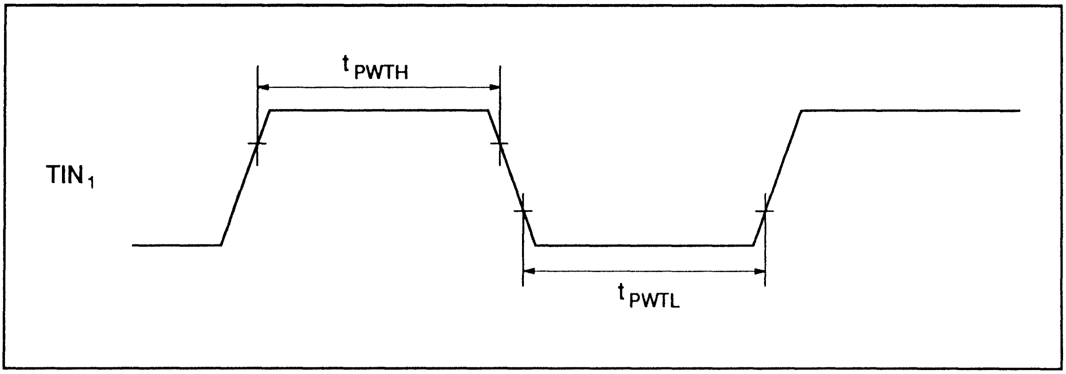


Figure 15 Timer 1 Input Pulse Duration

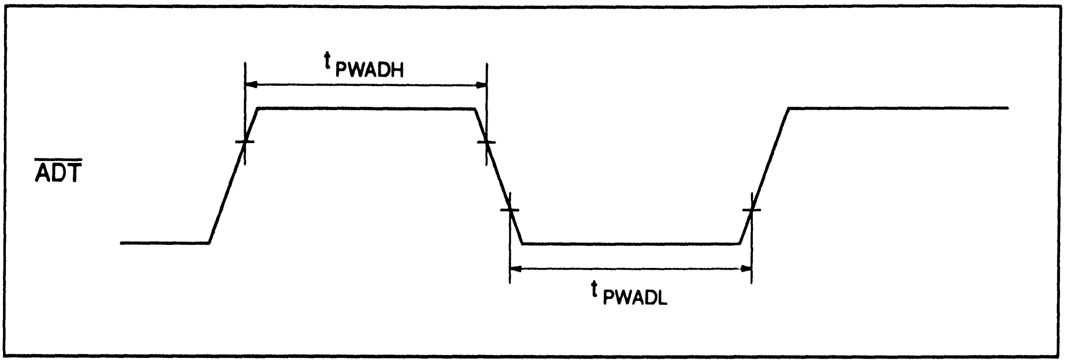


Figure 16 $\overline{ADT}$ Input Pulse Duration

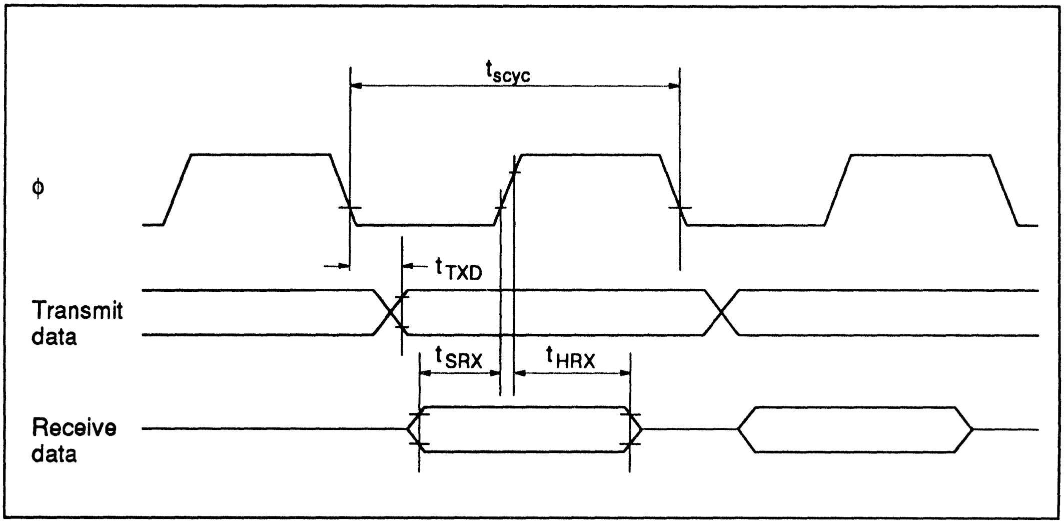


Figure 17 SCI Clocked Synchronous Timing (direct phase)

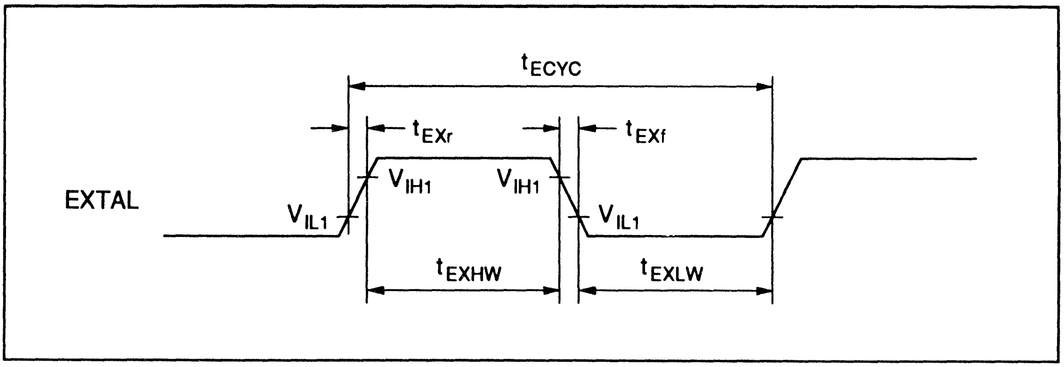


Figure 18 Rise and Fall Times for External Clock Input Signal

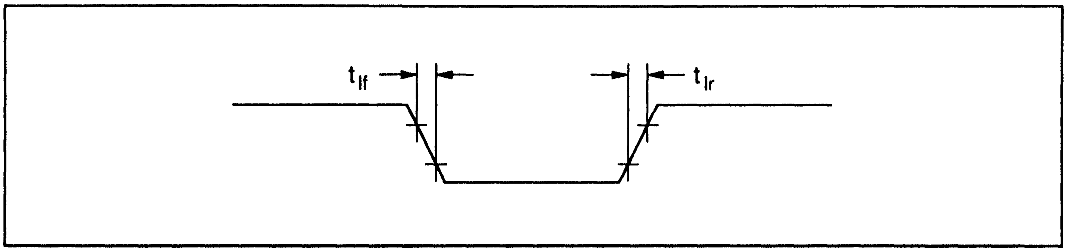


Figure 19 Rise and Fall Times for Input Signals (except EXTAL AND $\overline{\text{RESET}}$)

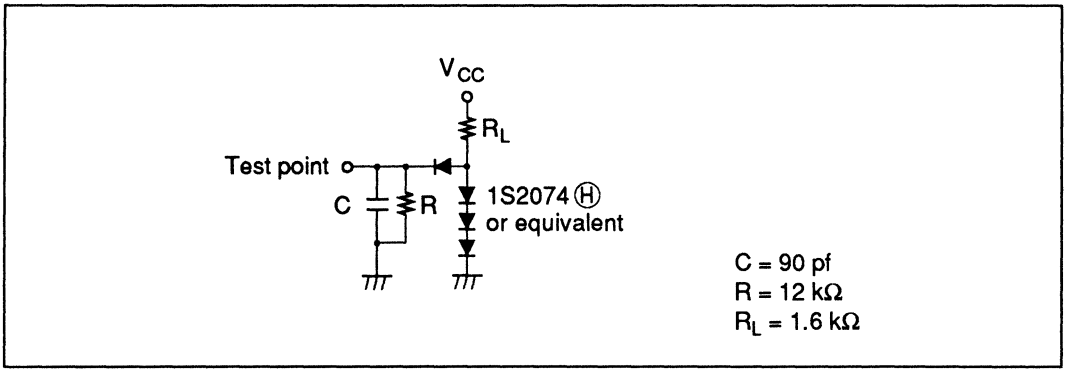


Figure 20 Bus Timing Test Load (TTL load)

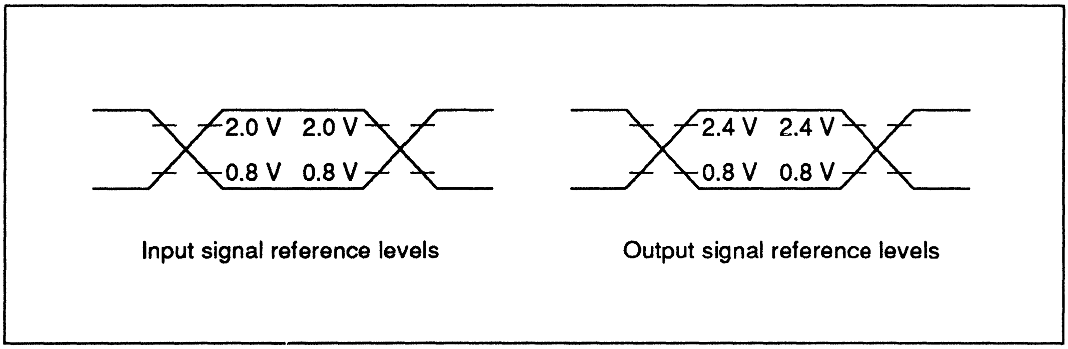


Figure 21 Reference Levels

Instruction Set Lists

The following explains the symbols used throughout the instruction set lists contained in this appendix.

1. Register Specification

The register specification symbols are: g, g', ww, xx, yy, and zz. The symbols g and g' represent 8-bit registers, while ww, xx, yy, and zz represent 16-bit registers, as shown below.

g, g'	Reg.	ww	Reg.	xx	Reg.	yy	Reg.	zz	Reg.
0 0 0	B	0 0	BC	0 0	BC	0 0	BC	0 0	BC
0 0 1	C	0 1	DE	0 1	DE	0 1	DE	0 1	DE
0 1 0	D	1 0	HL	1 0	IX	1 0	IY	1 0	HL
0 1 1	E	1 1	SP	1 1	SP	1 1	SP	1 1	AF
1 0 0	H								
1 0 1	L								
1 1 1	A								

Note: The letters "H" (high) and "L" (low) are appended to the 16-bit register symbols to indicate the upper 8 bits and lower 8 bits. For example: wwH, IXL.

2. Bit Specification

The symbol b in the bit operand of a bit manipulation instruction, specifies the bit location as shown below.

b	Bit
0 0 0	0
0 0 1	1
0 1 0	2
0 1 1	3
1 0 0	4
1 0 1	5
1 1 0	6
1 1 1	7

3. Condition Specification

The symbol f specifies the condition against which the result of an operation is compared to determine the next instruction to be executed, as shown below.

f	Condition	
0 0 0	NZ	non zero
0 0 1	Z	zero
0 1 0	NC	non carry
0 1 1	C	carry
1 0 0	PO	parity odd
1 0 1	PE	parity even
1 1 0	P	sign plus
1 1 1	M	sign minus

4. Restart Address

The symbol v specifies the restart address for a restart instruction, as shown below.

v	Address
0 0 0	00H
0 0 1	08H
0 1 0	10H
0 1 1	18H
1 0 0	20H
1 0 1	28H
1 1 0	30H
1 1 1	38H

5. Flags

The following symbols are used to denote changes in flags.

–: Flag not affected
×: Flag change undefined
↕: Flag affected according to operation result
S: Flag set to 1
R: Flag reset to 0
P: Flag changes as a parity flag
V: Flag changes as an overflow flag

6. Other Symbols

$(\ )_M$: Parentheses contain memory address
$(\ )_I$: Parentheses contain I/O address
m or n: 8-bit value
mn: 16-bit value
r: r suffix indicates 8-bit register
R: R suffix indicates 16-bit register
$b \bullet (\ )_M$: b-th bit of memory address in parentheses
b • gr: b-th bit of general register gr
d or j: Signed 8-bit displacement
S: Source addressing mode
D: Destination addressing mode
•: AND operation
+: OR operation
⊕: XOR operation

• Data Manipulation Instructions

1. Arithmetic and Logical Instructions (8-bit)

Operation Name	Mnemonics	Op Code	Addressing							Bytes	States	Operation	Flag					
			IMMED	EXT	INDX	REG	REGI	IMP	REL				7	6	4	2	1	0
													S	Z	H	P/V	N	C
ADD	ADD A, g	10 000 g				S		D		1	4	Ar + gr → Ar	↕	↕	↕	V	R	↕
	ADD A, (HL)	10 000 110					S	D		1	6	Ar + $(HL)_M$ → Ar	↕	↕	↕	V	R	↕
	ADD A, m	11 000 110 <m>	S					D		2	6	Ar + m → Ar						
	ADD A, (IX + d)	11 011 101 10 000 110 <d>			S			D		3	14	Ar + $(IX + d)_M$ → Ar	↕	↕	↕	V	R	↕
	ADD A, (IY + d)	11 111 101 10 000 110 <d>			S			D		3	14	Ar + $(IY + d)_M$ → Ar	↕	↕	↕	V	R	↕
ADC	ADC A, g	10 001 g				S		D		1	4	Ar + gr + c → Ar	↕	↕	↕	V	R	↕
	ADC A, (HL)	10 001 110					S	D		1	6	Ar + $(HL)_M$ + c → Ar	↕	↕	↕	V	R	↕
	ADC A, m	11 001 110 <m>	S					D		2	6	Ar + m + c → Ar	↕	↕	↕	V	R	↕
	ADC A, (IX + d)	11 011 101 10 001 110 <d>			S			D		3	14	Ar + $(IX + d)_M$ + c → Ar	↕	↕	↕	V	R	↕
	ADC A, (IY + d)	11 111 101 10 001 110 <d>			S			D		3	14	Ar + $(IY + d)_M$ + c → Ar	↕	↕	↕	V	R	↕
AND	AND g	10 100 g				S		D		1	4	Ar·gr → Ar	↕	↕	S	P	R	R
	AND (HL)	10 100 110					S	D		1	6	Ar·$(HL)_M$ → Ar	↕	↕	S	P	R	R
	AND m	11 100 110 <m>	S					D		2	6	Ar·m → Ar	↕	↕	S	P	R	R
	AND (IX + d)	11 011 101 10 100 110 <d>			S			D		3	14	Ar·$(IX + d)_M$ → Ar	↕	↕	S	P	R	R
	AND (IY + d)	11 111 101 10 100 110 <d>			S			D		3	14	Ar·$(IY + d)_M$ → Ar	↕	↕	S	P	R	R
Compare	CP g	10 111 g				S		D		1	4	Ar − gr	↕	↕	↕	V	S	↕
	CP (HL)	10 111 110					S	D		1	6	Ar − $(HL)_M$	↕	↕	↕	V	S	↕
	CP m	11 111 110 <m>	S					D		2	6	Ar − m	↕	↕	↕	V	S	↕

1. Arithmetic and Logical Instructions (8-bit) (cont)

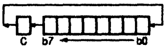
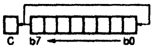
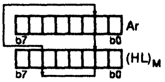
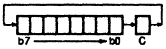
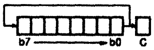
			Addressing										Flag					
													7	6	4	2	1	0
Operation Name	Mnemonics	Op Code	IMMED	EXT	INDX	REG	REGI	IMP	REL	Bytes	States	Operation	S	Z	H	P/V	N	C
Compare	CP (IX + d)	11 011 101 10 111 110 <d>			S			D		3	14	Ar – (IX + d)$_M$	↕	↕	↕	V	S	↕
	CP (IY + d)	11 111 101 10 111 110 <d>			S			D		3	14	Ar – (IY + d)$_M$	↕	↕	↕	V	S	↕
Comple-ment	CPL	00 101 111						S/D		1	3	$\overline{\mathrm{Ar}}$ → Ar	—	—	S	—	S	—
DEC	DEC g	00 g 101				S/D				1	4	gr – 1 → gr	↕	↕	↕	V	S	—
	DEC (HL)	00 110 101					S/D			1	10	(HL)$_M$ – 1 → (HL)$_M$	↕	↕	↕	V	S	—
	DEC (IX + d)	11 011 101 00 110 101 <d>			S/D					3	18	(IX + d)$_M$ – 1 → (IX + d)$_M$	↕	↕	↕	V	S	—
	DEC (IY + d)	11 111 101 00 110 101 <d>			S/D					3	18	(IY + d)$_M$ – 1 → (IY + d)$_M$	↕	↕	↕	V	S	↕
INC	INC g	00 g 100				S/D				1	4	gr + 1 → gr	↕	↕	↕	V	R	—
	INC (HL)	00 110 100					S/D			1	10	(HL)$_M$ + 1 → (HL)$_M$	↕	↕	↕	V	R	—
	INC (IX + d)	11 011 101 00 110 100 <d>			S/D					3	18	(IX + d)$_M$ + 1 → (IX + d)$_M$	↕	↕	↕	V	R	—
	INC (IY + d)	11 111 101 00 110 100 <d>			S/D					3	18	(IY + d)$_M$ + 1 → (IY + d)$_M$	↕	↕	↕	V	R	—
MULT	MLT ww	11 101 101 01 ww1 100				S/D				2	17	wwHr × wwLr → ww	—	—	—	—	—	—
NEGATE	NEG	11 101 101 01 000 100						S/D		2	6	0 – Ar → Ar	↕	↕	↕	V	S	↕
OR	OR g	10 110 g				S		D		1	4	Ar + gr → Ar	↕	↕	R	P	R	R
	OR (HL)	10 110 110					S	D		1	6	Ar + (HL)$_M$ → Ar	↕	↕	R	P	R	R
	OR m	11 110 110 <m>	S					D		2	6	Ar + m → Ar	↕	↕	R	P	R	R
	OR (IX + d)	11 011 101 10 110 110 <d>			S			D		3	14	Ar + (IX + d)$_M$ → Ar	↕	↕	R	P	R	R
	OR (IY + d)	11 111 101 10 110 110 <d>			S			D		3	14	Ar + (IY + d)$_M$ → Ar	↕	↕	R	P	R	R

1. Arithmetic and Logical Instructions (8-bit) (cont)

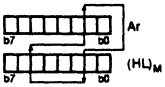
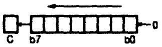
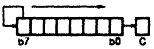
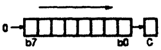
			Addressing										Flag					
													7	6	4	2	1	0
Operation Name	Mnemonics	Op Code	IMMED	EXT	INDX	REG	REGI	IMP	REL	Bytes	States	Operation	S	Z	H	P/V	N	C
SUB	SUB g	10 010 g				S		D		1	4	$Ar - gr \rightarrow Ar$	↕	↕	↕	V	S	↕
	SUB (HL)	10 010 110					S	D		1	6	$Ar - (HL)_M \rightarrow Ar$	↕	↕	↕	V	S	↕
	SUB m	11 010 110 <m>	S					D		2	6	$Ar - m \rightarrow Ar$	↕	↕	↕	V	S	↕
	SUB (IX + d)	11 011 101 10 010 110 <d>			S			D		3	14	$Ar - (IX + d)_M \rightarrow Ar$	↕	↕	↕	V	S	↕
	SUB (IY + d)	11 111 101 10 010 110 <d>			S			D		3	14	$Ar - (IY + d)_M \rightarrow Ar$	↕	↕	↕	V	S	↕
SUBC	SBC A, g	10 011 g				S		D		1	4	$Ar - gr - c \rightarrow Ar$	↕	↕	↕	V	S	↕
	SBC A, (HL)	10 011 110					S	D		1	6	$Ar - (HL)_M - c \rightarrow Ar$	↕	↕	↕	V	S	↕
	SBC A, m	11 011 110 <m>	S					D		2	6	$Ar - m - c \rightarrow Ar$	↕	↕	↕	V	S	↕
	SBC A, (IX + d)	11 011 101 10 011 110 <d>			S			D		3	14	$Ar - (IX + d)_M - c \rightarrow Ar$	↕	↕	↕	V	S	↕
	SBC A, (IY + d)	11 111 101 10 011 110 <d>			S			D		3	14	$Ar - (IY + d)_M - c \rightarrow Ar$	↕	↕	↕	V	S	↕
TEST	TST g	11 101 101 00 g 100				S				2	7	$Ar \cdot gr$	↕	↕	S	P	R	R
	TST (HL)	11 101 101 00 110 100					S			2	10	$Ar \cdot (HL)_M$	↕	↕	S	P	R	R
	TST m	11 101 101 01 100 100 <m>	S							3	9	$Ar \cdot m$	↕	↕	S	P	R	R
XOR	XOR g	10 101 g				S		D		1	4	$Ar \oplus gr \rightarrow Ar$	↕	↕	R	P	R	R
	XOR (HL)	10 101 110					S	D		1	6	$Ar \oplus (HL)_M \rightarrow Ar$	↕	↕	R	P	R	R
	XOR m	11 101 110 <m>	S					D		2	6	$Ar \oplus m \rightarrow Ar$	↕	↕	R	P	R	R
	XOR (IX + d)	11 011 101 10 101 110 <d>			S			D		3	14	$Ar \oplus (IX + d)_M \rightarrow Ar$	↕	↕	R	P	R	R
	XOR (IY + d)	11 111 101 10 101 110 <d>			S			D		3	14	$Ar \oplus (IY + d)_M \rightarrow Ar$	↕	↕	R	P	R	R

3

2. Rotate and Shift Instructions

Operation Name	Mnemonics	Op Code	Addressing							Bytes	States	Operation	Flag					
													7	6	4	2	1	0
			IMMED	EXT	INDX	REG	REGI	IMP	REL				S	Z	H	P/V	N	C
Rotate and shift data	RLA	00 010 111						S/D		1	3		—	—	R	—	R	↕
	RL g	11 001 011 00 010 g				S/D				2	7		↕	↕	R	P	R	↕
	RL (HL)	11 001 011 00 010 110					S/D			2	13		↕	↕	R	P	R	↕
	RL (IX + d)	11 011 101 11 001 011 <d> 00 010 110			S/D					4	19		↕	↕	R	P	R	↕
	RL (IY + d)	11 111 101 11 001 011 <d> 00 010 110			S/D					4	19		↕	↕	R	P	R	↕
	RLCA	00 000 111						S/D		1	3		—	—	R	—	R	↕
	RLC g	11 001 011 00 000 g				S/D				2	7		↕	↕	R	P	R	↕
	RLC (HL)	11 001 011 00 000 110					S/D			2	13		↕	↕	R	P	R	↕
	RLC (IX + d)	11 011 101 11 001 011 <d> 00 000 110			S/D					4	19		↕	↕	R	P	R	↕
	RLC (IY + d)	11 111 101 11 001 011 <d> 00 000 110			S/D					4	19		↕	↕	R	P	R	↕
	RLD	11 101 101 01 101 111						S/D		2	16		↕	↕	R	P	R	—
	RRA	00 011 111						S/D		1	3		—	—	R	—	R	↕
	RR g	11 001 011 00 011 g				S/D				2	7		↕	↕	R	P	R	↕
	RR (HL)	11 001 011 00 011 110					S/D			2	13		↕	↕	R	P	R	↕
	RR (IX + d)	11 011 101 11 001 011 <d> 00 011 110			S/D					4	19		↕	↕	R	P	R	↕
	RR (IY + d)	11 111 101 11 001 011 <d> 00 011 110			S/D					4	19		↕	↕	R	P	R	↕
	RRCA	00 001 111						S/D		1	3		—	—	R	—	R	↕
	RRC g	11 001 011 00 001 g				S/D				2	7		↕	↕	R	P	R	↕

2. Rotate and Shift Instructions (cont)

Operation Name	Mnemonics	Op Code	Addressing							Bytes	States	Operation	Flag					
			IMMED	EXT	INDX	REG	REGI	IMP	REL				7	6	4	2	1	0
													S	Z	H	P/V	N	C
Rotate and shift data	RRC (HL)	11 001 011 00 001 110					S/D			2	13		↕	↕	R	P	R	↕
	RRC (IX + d)	11 011 101 11 001 011 <d> 00 001 110			S/D					4	19		↕	↕	R	P	R	↕
	RRC (IY + d)	11 111 101 11 001 011 <d> 00 001 110			S/D					4	19		↕	↕	R	P	R	↕
	RRD	11 101 101 01 100 111						S/D		2	16		↕	↕	R	P	R	—
	SLA g	11 001 011 00 100 g				S/D				2	7		↕	↕	R	P	R	↕
	SLA (HL)	11 001 011 00 100 110					S/D			2	13		↕	↕	R	P	R	↕
	SLA (IX + d)	11 011 101 11 001 011 <d> 00 100 110			S/D					4	19		↕	↕	R	P	R	↕
	SLA (IY + d)	11 111 101 11 001 011 <d> 00 100 110			S/D					4	19		↕	↕	R	P	R	↕
	SRA g	11 001 011 001 101 g				S/D				2	7		↕	↕	R	P	R	↕
	SRA (HL)	11 001 011 00 101 110					S/D			2	13		↕	↕	R	P	R	↕
	SRA (IX + d)	11 011 101 11 001 011 <d> 00 101 110			S/D					4	19		↕	↕	R	P	R	↕
	SRA (IY + d)	11 111 101 11 001 011 <d> 00 101 110			S/D					4	19		↕	↕	R	P	R	↕
	SRL g	11 001 011 00 111 g				S/D				2	7		↕	↕	R	P	R	↕
	SRL (HL)	11 001 011 00 111 110					S/D			2	3		↕	↕	R	P	R	↕
	SRL (IX + d)	11 011 101 11 001 011 <d> 00 111 110			S/D					4	19		↕	↕	R	P	R	↕
	SRL (IY + d)	11 111 101 11 001 011 <d> 00 111 110			S/D					4	19		↕	↕	R	P	R	↕

3. Bit Manipulation Instructions

Operation Name	Mnemonics	Op Code	Addressing							Bytes	States	Operation	Flag					
													7	6	4	2	1	0
			IMMED	EXT	INDX	REG	REGI	IMP	REL				S	Z	H	P/V	N	C
Bit set	SET b, g	11 001 011 11 b g				S/D				2	7	$1 \to b \cdot gr$	—	—	—	—	—	—
	SET b, (HL)	11 001 011 11 b 110					S/D			2	13	$1 \to b \cdot (HL)_M$	—	—	—	—	—	—
	SET b, (IX + d)	11 011 101 11 001 011 <d> 11 b 110			S/D					4	19	$1 \to b \cdot (IX + d)_M$	—	—	—	—	—	—
	SET b, (IY + d)	11 111 101 11 001 011 <d> 11 b 110			S/D					4	19	$1 \to b \cdot (IY + d)_M$	—	—	—	—	—	—
Bit reset	RES b, g	11 001 011 10 b g				S/D				2	7	$0 \to b \cdot gr$	—	—	—	—	—	—
	RES b, (HL)	11 001 011 10 b 110					S/D			2	13	$0 \to b \cdot (HL)_M$	—	—	—	—	—	—
	RES b, (IX + d)	11 011 101 11 001 011 <d> 10 b 110			S/D					4	19	$0 \to b \cdot (IX + d)_M$	—	—	—	—	—	—
	RES b, (IY + d)	11 111 101 11 001 011 <d> 10 b 110			S/D					4	19	$0 \to b \cdot (IY + d)_M$	—	—	—	—	—	—
Bit test	BIT b, g	11 001 011 01 b g				S				2	6	$\overline{b \cdot gr} \to z$	X	↕	S	X	R	—
	BIT b, (HL)	11 001 011 01 b 110					S			2	9	$\overline{b \cdot (HL)_M} \to z$	X	↕	S	X	R	—
	BIT b, (IX + d)	11 011 101 11 001 011 <d> 01 b 110			S					4	15	$\overline{b \cdot (IX + d)_M} \to z$	X	↕	S	X	R	—
	BIT b, (IY + d)	11 111 101 11 001 011 <d> 01 b 110			S					4	15	$\overline{b \cdot (IY + d)_M} \to z$	X	↕	S	X	R	—

4. Arithmetic Instructions (16-bit)

Operation Name	Mnemonics	Op Code	Addressing							Bytes	States	Operation	Flag					
			IMMED	EXT	INDX	REG	REGI	IMP	REL				7	6	4	2	1	0
													S	Z	H	P/V	N	C
ADD	ADD HL, ww	00 ww1 001				S		D		1	7	$HL_R + ww_R \rightarrow HL_R$	—	—	X	—	R	↕
	ADD IX, xx	11 011 101 00 xx1 001				S		D		2	10	$IX_R + xx_R \rightarrow IX_R$	—	—	X	—	R	↕
	ADD IY, yy	11 111 101 00 yy1 001				S		D		2	10	$IY_R + yy_R \rightarrow IY_R$	—	—	X	—	R	↕
ADC	ADC HL, ww	11 101 101 01 ww1 010				S		D		2	10	$HL_R + ww_R + c \rightarrow HL_R$	↕	↕	X	V	R	↕
DEC	DEC ww	00 ww1 011				S/D				1	4	$ww_R - 1 \rightarrow ww_R$	—	—	—	—	—	—
	DEC IX	11 011 101 00 101 011						S/D		2	7	$IX_R - 1 \rightarrow IX_R$	—	—	—	—	—	—
	DEC IY	11 111 101 00 101 011						S/D		2	7	$IY_R - 1 \rightarrow IY_R$	—	—	—	—	—	—
INC	INC ww	00 ww0 011				S/D				1	4	$ww_R + 1 \rightarrow ww_R$	—	—	—	—	—	—
	INC IX	11 011 101 00 100 011						S/D		2	7	$IX_R + 1 \rightarrow IX_R$	—	—	—	—	—	—
	INC IY	11 111 101 00 100 011						S/D		2	7	$IY_R + 1 \rightarrow IY_R$	—	—	—	—	—	—
SBC	SBC HL, ww	11 101 101 01 ww0 010				S		D		2	10	$HL_R - ww_R - c \rightarrow HL_R$	↕	↕	X	V	S	↕

3

• Data Transfer Instructions

1. 8-Bit Load

Operation Name	Mnemonics	Op Code	Addressing							Bytes	States	Operation	Flag					
			IMMED	EXT	INDX	REG	REGI	IMP	REL				7	6	4	2	1	0
													S	Z	H	P/V	N	C
Load 8-bit data	LD A, I	11 101 101 01 010 111						S/D		2	6	Ir → Ar	↕	↕	R	IEF_2	R	—
	LD A, R	11 101 101 01 011 111						S/D		2	6	Rr → Ar	↕	↕	R	IEF_2	R	—
	LD A, (BC)	00 001 010					S	D		1	6	$(BC)_M$ → Ar(Note)	—	—	—	—	—	—
	LD A, (DE)	00 011 010					S	D		1	6	$(DE)_M$ → Ar	—	—	—	—	—	—
	LD A, (mn)	00 111 010 <n> <m>		S				D		3	12	$(mn)_M$ → Ar	—	—	—	—	—	—
	LD I, A	11 101 101 01 000 111						S/D		2	6	Ar → Ir	—	—	—	—	—	—
	LD R, A	11 101 101 01 001 111						S/D		2	6	Ar → Rr	—	—	—	—	—	—
	LD (BC), A	00 000 010					D	S		1	7	Ar → $(BC)_M$	—	—	—	—	—	—
	LD (DE), A	00 010 010					D	S		1	7	Ar → $(DE)_M$	—	—	—	—	—	—
	LD (mn), A	00 110 010 <n> <m>		D				S		3	13	Ar → $(mn)_M$	—	—	—	—	—	—
	LD g, g'	01 g g'				S/D				1	4	gr' → gr	—	—	—	—	—	—
	LD g, (HL)	01 g 110				D	S			1	6	$(HL)_M$ → gr	—	—	—	—	—	—
	LD g, m	00 g 110 <m>	S			D				2	6	m → gr	—	—	—	—	—	—
	LD g, (IX + d)	11 011 101 01 g 110 <d>			S	D				3	14	$(IX + d)_M$ → gr	—	—	—	—	—	—
	LD g, (IY + d)	11 111 101 01 g 110 <d>			S	D				3	14	$(IY + d)_M$ → gr	—	—	—	—	—	—
	LD (HL), m	00 110 110 <m>	S				D			2	9	m → $(HL)_M$	—	—	—	—	—	—
	LD (IX + d), m	11 011 101 00 110 110 <d> <m>	S		D					4	15	m → $(IX + d)_M$	—	—	—	—	—	—
	LD (IY + d), m	11 111 101 00 110 110 <d> <m>	S		D					4	15	m → $(IY + d)_M$	—	—	—	—	—	—

Note: Interrupts are not detected at the end of the LD A, I and LD A, R instructions.

1. 8-Bit Load (cont)

Operation Name	Mnemonics	Op Code	Addressing							Bytes	States	Operation	Flag					
			IMMED	EXT	INDX	REG	REGI	IMP	REL				7	6	4	2	1	0
													S	Z	H	P/V	N	C
Load 8-bit data	LD (HL), g	01 110 g				S	D			1	7	$gr \rightarrow (HL)_M$	—	—	—	—	—	—
	LD (IX + d), g	11 011 101 01 110 g <d>			D	S				3	15	$gr \rightarrow (IX + d)_M$	—	—	—	—	—	—
	LD (IY + d), g	11 111 101 01 110 g <d>			D	S				3	15	$gr \rightarrow (IY + d)_M$	—	—	—	—	—	—

2. 16-Bit Load

Operation Name	Mnemonics	Op Code	Addressing							Bytes	States	Operation	Flag					
			IMMED	EXT	INDX	REG	REGI	IMP	REL				7	6	4	2	1	0
													S	Z	H	P/V	N	C
Load 16-bit data	LD ww, mn	00 ww0 001 <n> <m>	S			D				3	9	$mn \rightarrow ww_R$	—	—	—	—	—	—
	LD IX, mn	11 011 101 00 100 001 <n> <m>	S					D		4	12	$mn \rightarrow IX_R$	—	—	—	—	—	—
	LD IY, mn	11 111 101 00 100 001 <n> <m>	S					D		4	12	$mn \rightarrow IY_R$	—	—	—	—	—	—
	LD SP, HL	11 111 001						S/D		1	4	$HL_R \rightarrow SP_R$	—	—	—	—	—	—
	LD SP, IX	11 011 101 11 111 001						S/D		2	7	$IX_R \rightarrow SP_R$	—	—	—	—	—	—
	LD SP, IY	11 111 101 11 111 001						S/D		2	7	$IY_R \rightarrow SP_R$	—	—	—	—	—	—
	LD ww, (mn)	11 101 101 01 ww1 011 <n> <m>		S		D				4	18	$(mn + 1)_M \rightarrow wwHr$	—	—	—	—	—	—
	LD HL, (mn)	00 101 010 <n> <m>		S				D		3	15	$(mn + 1)_M \rightarrow Hr$ $(mn)_M \rightarrow Lr$	—	—	—	—	—	—

2. 16-Bit Load (cont)

			Addressing										Flag					
													7	6	4	2	1	0
Operation Name	Mnemonics	Op Code	IMMED	EXT	INDX	REG	REGI	IMP	REL	Bytes	States	Operation	S	Z	H	P/V	N	C
Load 16-bit data	LD IX, (mn)	11 011 101 00 101 010 <n> <m>		S				D		4	18	$(mn+1)_M$ → IXHr $(mn)_M$ → IXLr	—	—	—	—	—	—
	LD IY, (mn)	11 111 101 00 101 010 <n> <m>		S				D		4	18	$(mn+1)_M$ → IYHr $(mn)_M$ → IYLr	—	—	—	—	—	—
	LD (mn), ww	11 101 101 01 ww0 011 <n> <m>		D		S				4	19	wwHr → $(mn+1)_M$ wwLr → $(mn)_M$	—	—	—	—	—	—
	LD (mn), HL	00 100 010 <n> <m>		D				S		3	16	Hr → $(mn+1)_M$ Lr → $(mn)_M$	—	—	—	—	—	—
	LD (mn), IX	11 011 101 00 100 010 <n> <m>		D				S		4	19	IXHr → $(mn+1)_M$ IXLr → $(mn)_M$	—	—	—	—	—	—
	LD (mn), IY	11 111 101 00 100 010 <n> <m>		D				S		4	19	IYHr → $(mn+1)_M$ IYLr → $(mn)_M$	—	—	—	—	—	—

3. Block Transfer

Operation Name	Mnemonics	Op Code	Addressing							Bytes	States	Operation	Flag					
			IMMED	EXT	INDX	REG	REGI	IMP	REL				7	6	4	2	1	0
													S	Z	H	P/V	N	C
														*2		*1		
Block transfer search data	CPD	11 101 101 10 101 001					S	S		2	12	Ar – $(HL)_M$ $BC_R - 1 \rightarrow BC_R$ $HL_R - 1 \rightarrow HL_R$	↕	↕	↕	↕	S	—
														*2		*1		
	CPDR	11 101 101 10 111 001					S	S		2	14 12	$BC_R \neq 0$ Ar ≠ $(HL)_M$ $BC_R = 0$ or Ar = $(HL)_M$	↕	↕	↕	↕	S	—
												$Q\begin{cases}\text{Ar} - (HL)_R\\ BC_R - 1 \rightarrow BC_R\\ HL_R - 1 \rightarrow HL_R\end{cases}$ Repeat Q until		*2		*1		
												Ar = $(HL)_M$ or $BC_R = 0$	↕	↕	↕	↕	S	—
	CPI	11 101 101 10 100 001					S	S		2	12	Ar – $(HL)_M$ $BC_R - 1 \rightarrow BC_R$ $HL_R + 1 \rightarrow HL_R$	↕	↕	↕	↕	S	—
														*2		*1		
	CPIR	11 101 101 10 110 001					S	S		2	14 12	$BC_R \neq 0$ Ar ≠ $(HL)_M$ $BC_R = 0$ or Ar = $(HL)_M$ $Q\begin{cases}\text{Ar} - (HL)_M\\ BC_R - 1 \rightarrow BC_R\\ HL_R + 1 \rightarrow HL_R\end{cases}$ Repeat Q until Ar = $(HL)_M$ or $BC_R = 0$	↕	↕	↕	↕	S	—
																*1		
	LDD	11 101 101 10 101 000					S/D			2	12	$(HL)_M \rightarrow (DE)_M$ $BC_R - 1 \rightarrow BC_R$ $DE_R - 1 \rightarrow DE_R$ $HL_R - 1 \rightarrow HL_R$	—	—	R	↕	R	—
	LDDR	11 101 101 10 111 000					S/D			2	14 ($BC_R \neq 0$) 12 ($BC_R = 0$)	$Q\begin{cases}(HL)_M \rightarrow (DE)_M\\ BC_R - 1 \rightarrow BC_R\\ DE_R - 1 \rightarrow DE_R\\ HL_R - 1 \rightarrow HL_R\end{cases}$ Repeat Q until $BC_R = 0$	—	—	R	R	R	—
																*1		
	LDI	11 101 101 10 100 000					S/D			2	12	$(HL)_M \rightarrow (DE)_M$ $BC_R - 1 \rightarrow BC_R$ $DE_R + 1 \rightarrow DE_R$ $HL_R + 1 \rightarrow HL_R$	—	—	R	↕	R	—
	LDIR	11 101 101 10 110 000					S/D			2	14 ($BC_R \neq 0$) 12 ($BC_R = 0$)	$Q\begin{cases}(HL)_M \rightarrow (DE)_M\\ BC_R - 1 \rightarrow BC_R\\ DE_R + 1 \rightarrow DE_R\\ HL_R + 1 \rightarrow HL_R\end{cases}$ Repeat Q until $BC_R = 0$	—	—	R	R	R	—

Notes 1 P/V = 0: $BC_R - 1 = 0$
P/V = 1· $BC_R - 1 \neq 0$
2. Z = 1· Ar = $(HL)_M$
Z = 0: Ar ≠ $(HL)_M$

4. Stack and Exchange

			Addressing										Flag					
													7	6	4	2	1	0
Operation Name	Mnemonics	Op Code	IMMED	EXT	INDX	REG	REGI	IMP	REL	Bytes	States	Operation	S	Z	H	P/V	N	C
PUSH	PUSH zz	11 zz0 101				S		D		1	11	$zzLr \rightarrow (SP-2)_M$ $zzHr \rightarrow (SP-1)_M$ $SP_R - 2 \rightarrow SP_R$	—	—	—	—	—	—
	PUSH IX	11 011 101 11 100 101						S/D		2	14	$IXLr \rightarrow (SP-2)_M$ $IXHr \rightarrow (SP-1)_M$ $SP_R - 2 \rightarrow SP_R$	—	—	—	—	—	—
	PUSH IY	11 111 101 11 100 101						S/D		2	14	$IYLr \rightarrow (SP-2)_M$ $IYHr \rightarrow (SP-1)_M$ $SP_R - 2 \rightarrow SP_R$	—	—	—	—	—	—
POP	POP zz	11 zz0 001				D		S		1	9	$(SP+1)_M \rightarrow zzHr$(Note) $(SP)_M \rightarrow zzLr$ $SP_R + 2 \rightarrow SP_R$	—	—	—	—	—	—
	POP IX	11 011 101 11 100 001						S/D		2	12	$(SP+1)_M \rightarrow IXHr$ $(SP)_M \rightarrow IXr$ $SP_R + 2 \rightarrow SP_R$	—	—	—	—	—	—
	POP IY	11 111 101 11 100 001						S/D		2	12	$(SP+1)_M \rightarrow IYHr$ $(SP)_M \rightarrow IYLr$ $SP_R + 2 \rightarrow SP_R$	—	—	—	—	—	—
Exchange	EX AF, AF'	00 001 000						S/D		1	4	$AF_R \rightarrow AF_R'$	—	—	—	—	—	—
	EX DE, HL	11 101 011						S/D		1	3	$DE_R - HL_R$	—	—	—	—	—	—
	EXX	11 011 001						S/D		1	3	$BC_R - BC_R'$ $DE_R - DE_R'$ $HL_R - HL_R'$	— —	— —	— —	— —	— —	— —
	EX (SP), HL	11 100 011						S/D		1	16	$Hr - (SP+1)_M$ $Lr - (SP)_M$	—	—	—	—	—	—
	EX (SP), IX	11 011 101 11 100 011						S/D		2	19	$IXHr - (SP+1)_M$ $IXLr - (SP)_M$	—	—	—	—	—	—
	EX (SP), IY	11 111 101 11 100 011						S/D		2	19	$IYHr - (SP+1)_M$ $IYLr - (SP)_M$	—	—	—	—	—	—

Note: POP AF writes stack contents to flag.

• Program Control Instructions

Operation Name	Mnemonics	Op Code	Addressing							Bytes	States	Operation	Flag					
			IMMED	EXT	INDX	REG	REGI	IMP	REL				7	6	4	2	1	0
													S	Z	H	P/V	N	C
Call	CALL mn	11 001 101 <n> <m>		D						3	16	$PCHr \rightarrow (SP-1)_M$ $PCLr \rightarrow (SP-2)_M$ $mn \rightarrow PC_R$ $SP_R - 2 \rightarrow SP_R$	—	—	—	—	—	—
	CALL f, mn	11 f 100 <n> <m>		D						3	6 (f: false) 16 (f: true)	continue: f is false CALL mn: f is true	—	—	—	—	—	—
Jump	DJNZ j	00 010 000 <j - 2>							D	2 2	9 (Br ≠ 0) 7 (Br = 0)	$Br - 1 \rightarrow Br$ continue: Br = 0 $PC_R + j \rightarrow PC_R$: Br ≠ 0	—	—	—	—	—	—
	JP f, mn	11 f 010 <n> <m>		D						3 3	6 (f: false) 9 (f: true)	$mn \rightarrow PC_R$: f is true continue. f is false	—	—	—	—	—	—
	JP mn	11 000 011 <n> <m>		D						3	9	$mn \rightarrow PC_R$	—	—	—	—	—	—
	JP (HL)	11 101 001					D			1	3	$HL_R \rightarrow PC_R$	—	—	—	—	—	—
	JP (IX)	11 011 101 11 101 001					D			2	6	$IX_R \rightarrow PC_R$	—	—	—	—	—	—
	JP (IY)	11 111 101 11 101 001					D			2	6	$IY_R \rightarrow PC_R$	—	—	—	—	—	—
	JR j	00 011 000 <j- 2>							D	2	8	$PC_R + j \rightarrow PCv$	—	—	—	—	—	—
	JR C, j	00 111 000 <j- 2>							D	2 2	6 8	continue· C = 0 $PC_R + j \rightarrow PC_R$: C = 1	—	—	—	—	—	—
	JR NC, j	00 110 000 <j-2>							D	2 2	6 8	continue· C = 1 $PC_R + j \rightarrow PC_R$: C = 0	—	—	—	—	—	—
	JR Z, j	00 101 000 <j-2>							D	2 2	6 8	continue· Z = 0 $PC_R + j \rightarrow PC_R$: Z = 1	—	—	—	—	—	—
	JR NZ, j	00 100 000 <j-2>							D	2 2	6 8	continue: Z = 1 $PC_R + j \rightarrow PC_R$: Z = 0	—	—	—	—	—	—
Return	RET	11 001 001						D		1	9	$(SP)_M \rightarrow PCLr$ $(SP+1)_M \rightarrow PCHr$ $SP_R + 2 \rightarrow SP_R$	—	—	—	—	—	—
	RET f	11 f 000							D	1 1	5 (f: false) 10 (f: true)	continue: f is false RET: f is true	—	—	—	—	—	—
	RETI	11 101 101 01 001 101						D		2	22 (Z) 12 (R1)	$(SP)_M \rightarrow PCLr$ $(SP+1)_M \rightarrow PCHr$ $SP_R + 2 \rightarrow SP_R$	—	—	—	—	—	—

3

• Program Control Instructions (cont)

Operation Name	Mnemonics	Op Code	Addressing							Bytes	States	Operation	Flag					
			IMMED	EXT	INDX	REG	REGI	IMP	REL				7	6	4	2	1	0
													S	Z	H	P/V	N	C
Return	RETN	11 101 101 01 000 101						D		2	12	$(SP)_M \rightarrow PCLr$ $(SP+1)_M \rightarrow PCHr$ $SP_R + 2 \rightarrow SP_R$ IEF2 → IEF1	—	—	—	—	—	—
Restart	RST v	11 v 111						D		1	11	$PCHr \rightarrow (SP-1)_M$ $PCLr \rightarrow (SP-2)_M$ 0 → PCHr v → PCLr $SP_R - 2 \rightarrow SP_R$	—	—	—	—	—	—

• I/O Instructions

Operation Name	Mnemonics	Op Code	Addressing							Bytes	States	Operation	Flag					
			IMMED	EXT	INDX	REG	REGI	IMP	REL				7	6	4	2	1	0
													S	Z	H	P/V	N	C
Input	IN A, (m)	11 011 011 <m>						D	S	2	9	$(Am)_I \rightarrow Ar$ $m \rightarrow A_0$ to A_7 $Ar \rightarrow A_8$ to A_{15}	—	—	—	—	—	—
	IN g, (C)	11 101 101 01 g 000				D			S	2	9	$(BC)_I \rightarrow gr$ g = 110: only the flags will change. $Cr \rightarrow A_0$ to A_7 $Br \rightarrow A_8$ to A_{15}	↕	↕	R	P	R	—
	IN0 g, (m)	11 101 101 00 g 000 <m>				D			S	3	12	$(00m)_I \rightarrow gr$ g = 110: only the flags will change. $m \rightarrow A_0$ to A_7 $00 \rightarrow A_8$ to A_{15}	↕	↕ *1	R	P	R *2	—
	IND	11 101 101 10 101 010					D		S	2	12	$(BC)_I \rightarrow (HL)M$ $HL_R - 1 \rightarrow HL_R$ Br − 1 → Br $Cr \rightarrow A_0$ to A_7 $Br \rightarrow A_8$ to A_{15}	X	↕	X	X	↕ *2	X

Notes: 1. Z = 1: Br − 1 = 0
Z = 0: Br − 1 ≠ 0
2. N = 1: MSB of data = 1
N = 0: MSB of data = 0

• I/O Instructions (cont)

Operation Name	Mnemonics	Op Code	Addressing							Bytes	States	Operation	Flag					
			IMMED	EXT	INDX	REG	REGI	IMP	REL				7	6	4	2	1	0
													S	Z	H	P/V	N	C
Input	INDR	11 101 101 10 111 010					D		S	2	14 (Br ≠ 0) 12 (Br = 0)	Q { $(BC)_I \to (HL)_M$ $HL_R - 1 \to HL_R$ $Br - 1 \to Br$ Repeat Q until Br = 0 $Cr \to A_0$ to A_7 $Br \to A_6$ to A_{15}	X	S *1	X	X	↕ *2	X
	INI	11 101 101 10 100 010					D		S	2	12	$(BC)_I \to (HL)_M$ $HL_R + 1 \to HL_R$ $Br - 1 \to Br$ $Cr \to A_0$ to A_7 $Br \to A_6$ to A_{15}	X	↕	X	X	↕ *2	X
	INIR	11 101 101 10 110 010					D		S	2	14 (Br ≠ 0) 12 (Br = 0)	Q { $(BC)_I \to (HL)_M$ $HL_R + 1 \to HL_R$ $Br - 1 \to Br$ Repeat Q until Br = 0 $Cr \to A_0$ to A_7 $Br \to A_6$ to A_{15}	X	S	X	X	↕	X
Output	OUT (m), A	11 010 011 <m>						S	D	2	10	$Ar \to (Am)_I$ $m \to A_0$ to A_7 $Ar \to A_6$ to A_{15}	—	—	—	—	—	—
	OUT (C), g	11 101 101 01 g 001				S			D	2	10	$gr \to (BC)_I$ $Cr \to A_0$ to A_7 $Br \to A_6$ to A_{15}	—	—	—	—	—	—
	OUT0 (m), g	11 101 101 00 g 001 <m>				S			D	3	13	$gr \to (00m)_I$ $m \to A_0$ to A_7 $00 \to A_6$ to A_{15}	—	— *1	—	—	— *2	—
	OTDM	11 101 101 10 001 011					S		D	2	14	$(HL)_M \to (00C)_I$ $HL_R - 1 \to HL_R$ $Cr - 1 \to Cr$ $Br - 1 \to Br$ $Cr \to A_0$ to A_7 $00 \to A_6$ to A_{15}	↕	↕	↕	P	↕ *2	↕
	OTDMR	11 101 101 10 011 011					S		D	2	16 (Br ≠ 0) 14 (Br = 0)	Q { $(HL)_M \to (00C)_I$ $HL_R - 1 \to HL_R$ $Cr - 1 \to Cr$ $Br - 1 \to Br$ Repeat Q until Br = 0 $Cr \to A_0$ to A_7 $00 \to A_6$ to A_{15}	R	S	R	S	↕ *2	R

Notes: 1. Z = 1: Br − 1 = 0
Z = 0: Br − 1 ≠ 0
2. N = 1: MSB of data = 1
N = 0: MSB of data = 0

3

• I/O Instructions (cont)

			Addressing										Flag					
													7	6	4	2	1	0
Operation Name	Mnemonics	Op Code	IMMED	EXT	INDX	REG	REGI	IMP	REL	Bytes	States	Operation	S	Z	H	P/V	N	C
Output	OTDR	11 101 101 10 111 011					S		D	2	14 (Br ≠ 0) 12 (Br = 0)	$Q\left\{\begin{array}{l}(HL)_M \to (BC)_I\\ HL_R - 1 \to HL_R\\ Br - 1 \to Br\end{array}\right.$ Repeat Q until Br = 0 $Cr \to A_0$ to A_7 $Br \to A_6$ to A_{15}	X	S *1	X	X	↕ *2	X
	OUTI	11 101 101 10 100 011					S		D	2	12	$(HL)_M \to (BC)_I$ $HL_R + 1 \to HL_R$ $Br - 1 \to Br$ $Cr \to A_0$ to A_7 $Br \to A_6$ to A_{15}	X	↕	X	X	↕ *2	X
	OTIR	11 101 101 10 110 011					S		D	2	14 (Br ≠ 0) 12 (Br = 0)	$Q\left\{\begin{array}{l}(HL)_M \to (BC)_I\\ HL_R + 1 \to HL_R\\ Br - 1 \to Br\end{array}\right.$ Repeat Q until Br = 0 $Cr \to A_0$ to A_7 $Br \to A_6$ to A_{15}	X	S	X	X	↕	X
	TSTIO m	11 101 101 01 110 100 <m>	S						S	3	12	$(00C)_I \cdot m$ $Cr \to A_0$ to A_7 $00 \to A_6$ to A_{15}	↕	↕ *1	S	P	R *2	R
	OTIM	11 101 101 10 000 011					S		D	2	14	$(HL)_M \to (00C)_I$ $HL_R + 1 \to HL_R$ $Cr + 1 \to Cr$ $Br - 1 \to Br$ $Cr \to A_0$ to A_7 $00 \to A_6$ to A_{15}	↕	↕	↕	P	↕ *2	↕
	OTIMR	11 101 101 10 010 011					S		D	2	16 (Br ≠ 0) 14 (Br = 0)	$Q\left\{\begin{array}{l}(HL)_M \to (00C)_I\\ HL_R + 1 \to HL_R\\ Cr + 1 \to Cr\\ Br - 1 \to Br\end{array}\right.$ Repeat Q until Br = 0 $Cr \to A_0$ to A_7 $00 \to A_6$ to A_{15}	R	S *1	R	S	↕ *2	R
	OUTD	11 101 101 10 101 011					S		D	2	12	$(HL)_M \to (BC)_I$ $HL_R - 1 \to HL_R$ $Br - 1 \to Br$ $Cr \to A_0$ to A_7 $Br \to A_6$ to A_{15}	X	↕	X	X	↕	X

Notes: 1. Z = 1: Br − 1 = 0
Z = 0: Br − 1 ≠ 0
2. N = 1: MSB of data = 1
N = 0: MSB of data = 0

• Special Control Instructions

Operation Name	Mnemonics	Op Code	Addressing							Bytes	States	Operation	Flag					
													7	6	4	2	1	0
			IMMED	EXT	INDX	REG	REGI	IMP	REL				S	Z	H	P/V	N	C
Special function	DAA	00 100 111						S/D		1	4	Decimal adjust accumulator	↕	↕	↕	P	—	↕
Carry control	CCF	00 111 111								1	3	$\overline{C} \rightarrow C$	—	—	R	—	R	↕
	SCF	00 110 111								1	3	$1 \rightarrow C$	—	—	R	—	R	S
CPU control	DI	11 110 011								1	3	$0 \rightarrow IEF_1, 0 \rightarrow IEF_2$(Note)	—	—	—	—	—	—
	EI	11 111 011								1	3	$1 \rightarrow IEF_1, 1 \rightarrow IEF_2$(Note)	—	—	—	—	—	—
	HALT	01 110 110								1	3	CPU halted	—	—	—	—	—	—
	IM 0	11 101 101 01 000 110								2	6	Interrupt mode 0	—	—	—	—	—	—
	IM 1	11 101 101 01 010 110								2	6	Interrupt mode 1	—	—	—	—	—	—
	IM 2	11 101 101 01 011 110								2	6	Interrupt mode 2	—	—	—	—	—	—
	NOP	00 000 000								1	3	No operation	—	—	—	—	—	—
	SLP	11 101 101 01 110 110								2	8	Sleep	—	—	—	—	—	—

Note: Interrupts are not detected at the end of the DI or EI instruction.

■ Alphabetical Instruction List

Mnemonics	Bytes	Machine Cycles	States
ADC A, m	2	2	6
ADC A, g	1	2	4
ADC A, (HL)	1	2	6
ADC A, (IX + d)	3	6	14
ADC A, (IY + d)	3	6	14
ADC HL, ww	2	6	10
ADD A, m	2	2	6
ADD A, g	1	2	4
ADD A, (HL)	1	2	6
ADD A, (IX + d)	3	6	14
ADD A, (IY + d)	3	6	14
ADD HL, ww	1	5	7
ADD IX, xx	2	6	10
ADD IY, yy	2	6	10
AND m	2	2	6
AND g	1	2	4
AND (HL)	1	2	6
AND (IX + d)	3	6	14
AND (IY + d)	3	6	14
BIT b, (HL)	2	3	9
BIT b, (IX + d)	4	5	15
BIT b, (IY + d)	4	5	15
BIT b, g	2	2	6
CALL f, mn	3 3	2 6	6 (If condition is false) 16 (If condition is true)
CALL mn	3	6	16
CCF	1	1	3
CPD	2	6	12
CPDR	2 2	8 6	14 (If $BC_R \neq 0$ and $Ar \neq (HL)_M$) 12 (If $BC_R = 0$ or $Ar = (HL)_M$)

Mnemonics	Bytes	Machine Cycles	States
CP (HL)	1	2	6
CPI	2	6	12
CPIR	2 2	8 6	14 (If $BC_R \neq 0$ and $Ar \neq (HL)_M$) 12 (If $BC_R = 0$ or $Ar = (HL)_M$)
CP (IX + d)	3	6	14
CP (IY + d)	3	6	14
CPL	1	1	3
CP m	2	2	6
CP g	1	2	4
DAA	1	2	4
DEC (HL)	1	4	10
DEC IX	2	3	7
DEC IY	2	3	7
DEC (IX + d)	3	8	18
DEC (IY + d)	3	8	18
DEC g	1	2	4
DEC ww	1	2	4
DI	1	1	3
DJNZ j	2 2	5 3	9 (If $Br \neq 0$) 7 (If $Br = 0$)
EI	1	1	3
EX AF, AF'	1	2	4
EX DE, HL	1	1	3
EX (SP), HL	1	6	16
EX (SP), IX	2	7	19
EX (SP), IY	2	7	19
EXX	1	1	3
HALT	1	1	3
IM 0	2	2	6
IM 1	2	2	6
IM 2	2	2	6

3

Mnemonics	Bytes	Machine Cycles	States
INC g	1	2	4
INC (HL)	1	4	10
INC (IX + d)	3	8	18
INC (IY + d)	3	8	18
INC ww	1	2	4
INC IX	2	3	7
INC IY	2	3	7
IN A, (m)	2	3	9
IN g, (C)	2	3	9
INI	2	4	12
INIR	2	6	14 (If Br ≠ 0)
INIR	2	4	12 (If Br = 0)
IND	2	4	12
INDR	2	6	14 (If Br ≠ 0)
	2	4	12 (If Br = 0)
IN0 g, (m)	3	4	12
JP f, mn	3	2	6 (If f is false)
	3	3	9 (If f is true)
JP (HL)	1	1	3
JP (IX)	2	2	6
JP (IY)	2	2	6
JP mm	3	3	9
JR j	2	4	8
JR C, j	2	2	6 (If condition is false)
	2	4	8 (If condition is true)
JR NC, j	2	2	6 (If condition is false)
	2	4	8 (If condition is true)
JR Z, j	2	2	6 (If condition is false)
	2	4	8 (If condition is true)
JR NZ, j	2	2	6 (If condition is false)
	2	4	8 (If condition is true)
LD A, (BC)	1	2	6
LD A, (DE)	1	2	6

Mnemonics	Bytes	Machine Cycles	States
LD A, I	2	2	6
LD A, (mm)	3	4	12
LD A, R	2	2	6
LD (BC), A	1	3	7
LDD	2	4	12
LD (DE), A	1	3	7
LD ww, mn	3	3	9
LD ww, (mn)	4	6	18
LDDR	2 2	6 4	14 (If $BC_R \neq 0$) 12 (If $BC_R = 0$)
LD (HL), m	2	3	9
LD HL, (mn)	3	5	15
LD (HL), g	1	3	7
LDI	2	4	12
LDI, A	2	2	6
LDIR	2 2	6 4	14 (If $BC_R \neq 0$) 12 (If $BC_R = 0$)
LD IX, mn	4	4	12
LD IX, (mn)	4	6	18
LD (IX + d), m	4	5	15
LD (IX + d), g	3	7	15
LD IY, mn	4	4	12
LD IY, (mn)	4	6	18
LD (IY + d), m	4	5	15
LD (IY + d), g	3	7	15
LD (mn), A	3	5	13
LD (mn), ww	4	7	19
LD (mn), HL	3	6	16
LD (mn), IX	4	7	19
LD (mn), IY	4	7	19
LD R, A	2	2	6
LD g, (HL)	1	2	6

Mnemonics	Bytes	Machine Cycles	States
LD g, (IX + d)	3	6	14
LD g, (IY + d)	3	6	14
LD g, m	2	2	6
LD g, g'	1	2	4
LD SP, HL	1	2	4
LD SP, IX	2	3	7
LD SP, IY	2	3	7
MLT ww	2	13	17
NEG	2	2	6
NOP	1	1	3
OR (HL)	1	2	6
OR (IX + d)	3	6	14
OR (IY + d)	3	6	14
OR m	2	2	6
OR g	1	2	4
OTDM	2	6	14
OTDMR	2	8	16 (If Br ≠ 0)
	2	6	14 (If Br = 0)
OTDR	2	6	14 (If Br ≠ 0)
	2	4	12 (If Br = 0)
OTIM	2	6	14
OTIMR	2	8	16 (If Br ≠ 0)
	2	6	14 (If Br = 0)
OTIR	2	6	14 (If Br ≠ 0)
	2	4	12 (If Br = 0)
OUTD	2	4	12
OUTI	2	4	12
OUT (m), A	2	4	10
OUT (C), g	2	4	10
OUT0 (m), g	3	5	13
POP IX	2	4	12
POP IY	2	4	12

Mnemonics	Bytes	Machine Cycles	States
POP zz	1	3	9
PUSH IX	2	6	14
PUSH IY	2	6	14
PUSH zz	1	5	11
RES b, (HL)	2	5	13
RES b, (IX + d)	4	7	19
RES b, (IY + d)	4	7	19
RES b, g	2	3	7
RET	1	3	9
RET f	1	3	5 (If condition is false)
	1	4	10 (If condition is true)
RETI	2	10 (Z) 4 (R1)	22 (Z) 12 (R1)
RETN	2	4	12
RLA	1	1	3
RLCA	1	1	3
RLC (HL)	2	5	13
RLC (IX + d)	4	7	19
RLC (IY + d)	4	7	19
RLC g	2	3	7
RLD	2	8	16
RL (HL)	2	5	13
RL (IX + d)	4	7	19
RL (IY + d)	4	7	19
RL g	2	3	7
RRA	1	1	3
RRCA	1	1	3
RRC (HL)	2	5	13
RRC (IX + d)	4	7	19
RRC (IY + d)	4	7	19
RRC g	2	3	7

Mnemonics	Bytes	Machine Cycles	States
RRD	2	8	16
RR (HL)	2	5	13
RR (IX + d)	4	7	19
RR (IY + d)	4	7	19
RR g	2	3	7
RST v	1	5	11
SBC A, (HL)	1	2	6
SBC A, (IX + d)	3	6	14
SBC A, (IY + d)	3	6	14
SBC A, m	2	2	6
SBC A, g	1	2	4
SBC HL, ww	2	6	10
SCF	1	1	3
SET b, (HL)	2	5	13
SET b, (IX + d)	4	7	19
SET b, (IY + d)	4	7	19
SET b, g	2	3	7
SLA (HL)	2	5	13
SLA (IX + d)	4	7	19
SLA (IY + d)	4	7	19
SLA g	2	3	7
SLP	2	2	8
SRA (HL)	2	5	13
SRA (IX + d)	4	7	19
SRA (IY + d)	4	7	19
SRA g	2	3	7
SRL (HL)	2	5	13
SRL (IX + d)	4	7	19
SRL (IY + d)	4	7	19
SRL g	2	3	7
SUB (HL)	1	2	6

Mnemonics	Bytes	Machine Cycles	States
SUB (IX + d)	3	6	14
SUB (IY + d)	3	6	14
SUB m	2	2	6
SUB g	1	2	4
TSTIO m	3	4	12
TST g	2	3	7
TST m	3	3	9
TST (HL)	2	4	10
XOR (HL)	1	2	6
XOR (IX + d)	3	6	14
XOR (IY + d)	3	6	14
XOR m	2	2	6
XOR g	1	2	4

■ Op Code Map

Table 3 Op Code Map (1)

First op code
Instruction format: XX

				ww (LO = ALL)												LO = 0 to 7				
				BC	DE	HL	SP									BC	DE	HL	AF	zz
				g (LO = 0 to 7)												NZ	NC	PO	P	f
				B	D	H	(HL)	B	D	H	(HL)					00H	10H	20H	30H	v
			HI	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111	
		LO		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
s (HI = ALL)	B	0000	0	NOP	DJNZ, j	JR NZ, j	JR NC, j	LD g, s				ADD A, s	SUB s	AND s	OR s	RET f				0
	C	0001	1	LD ww, mn							* 1					POP zz				1
	D	0010	2	LD (ww), A		LD (mn), HL	LD (mn), A									JP f, mn				2
	E	0011	3	INC ww												JP mn	OUT (m), A	EX (SP), HL	DI	3
	H	0100	4	INC g			* 1									CALL f, mn				4
	L	0101	5	DEC g			* 1									PUSH zz				5
	(HL)	0110	6	LD g, m			* 1	Note 2			HALT					ADD A, m	SUB m	AND m	OR m	6
	A	0111	7	RLCA	RLA	DAA	SCF					* 2	* 2	* 2	* 2	RST v				7
	B	1000	8	EXAF, AF	JR j	JR Z, j	JR C, j	LD g, s				ADC A, s	SBC A, s	XOR s	CP s	RET f				8
	C	1001	9	ADD HL, ww												RET	EXX	JP (HL)	LD SP, HL	9
	D	1010	A	LD A, (ww)		LD HL, (mn)	LD A, (mn)									JP f, mn				A
	E	1011	B	DEC ww												Table 2	IN A, (m)	EX DE, HL	EI	B
	H	1100	C	INC g												CALL f, mn				C
	L	1101	D	DEC g												CALL mn	* 3	Table 3	* 3	D
	(HL)	1110	E	LD g, m				* 2				* 2	* 2	* 2	* 2	ADC A, m	SBC A, m	XOR m	CP m	E
	A	1111	F	RRCA	RRA	CPL	CCF									RST v				F
				0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
				C	E	L	A	C	E	L	A					Z	C	PE	M	f
				g (LO = 8 to F)												08H	18H	28H	38H	v
																LO = 8 to F				

Notes: 1. g is replaced by (HL).

2. s is replaced by (HL).

3. Appending DD to the beginning of an op code (DD XX) in an instruction that has HL or (HL) in its operand produces the same operation as the original format with the following replacements:

HL replaced by IX

(HL) replaced by (IX + d)

Example:

22H ; LD (mn), HL → DDH 22H ; LD (mn), IX

Similarly, appending FD to the beginning of an op code (FD XX) in an instruction that has HL or (HL) in its operand produces the same operation as the original format with the following replacements:

HL replaced by IY

(HL) replaced by (IY + d)

Example:

34H ; INC (HL) → FDH 34H ; INC (IY + d)

An exception is the JP (HL) instruction (E9H). Appending DDH or FDH to the beginning replaces (HL) with (IX) or (IY).

If DDH or FDH is appended to the EX DE, HL instruction (EBH), an undefined instruction results, without replacement of HL.

Table 4 Op Code Map (2)

Second op code
Instruction format: CB XX

								b (LO = 0 to 7)												
								0	2	4	6	0	2	4	6	0	2	4	6	
			HI	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111	
		LO		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
g (HI = ALL)	B	0000	0																	0
	C	0001	1																	1
	D	0010	2																	2
	E	0011	3																	3
	H	0100	4	RLC g	RL g	SLA g		BIT b, g				RES b, g				SET b, g				4
	L	0101	5																	5
	(HL)	0110	6	* 1	* 1	* 1		* 1				* 1				* 1				6
	A	0111	7																	7
	B	1000	8																	8
	C	1001	9																	9
	D	1010	A																	A
	E	1011	B																	B
	H	1100	C	RRC g	RR g	SRA g	SRL g	BIT b, g				RES b, g				SET b, g				C
	L	1101	D																	D
	(HL)	1110	E	Note	Note	Note	Note	Note				Note				Note				E
	A	1111	F																	F
				0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
								1	3	5	7	1	3	5	7	1	3	5	7	
								b (LO = 8 to F)												

Note: If DDH is appended to the beginning of the op code, the instruction is executed by the op code DD CB d XX, replacing (HL) with (IX + d). Similarly, if FDH is appended to the beginning of the op code, the instruction is executed by the op code FD CB d XX, replacing (HL) with (IY + d).

Table 5 Op Code Map (3)

Second op code
Instruction format: ED XX

						ww (LO = ALL)												
						BC	DE	HL	SP									
		g (LO = 0 to 7)																
		B	D	H		B	D	H										
	HI	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111	
LO		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0000	0	IN0 g, (m)				IN g, (C)						LDI	LDIR					0
0001	1	OUT0 (m), g				OUT (C), g						CPI	CPIR					1
0010	2					SBC HL, ww						INI	INIR					2
0011	3					LD (mn), ww				OTIM	OTIMR	OUTI	OTIR					3
0100	4	TST g			TST (HL)	NEG		TST M	TSTIO m									4
0101	5					RETN												5
0110	6					IM 0	IM 1		SLP									6
0111	7					LD I, A	LD A, I	RRD										7
1000	8	IN0 g, (m)				IN g, (C)						LDD	LDDR					8
1001	9	OUT0 (m), g				OUT (C), g						CPD	CPDR					9
1010	A					ADC HL, ww						IND	INDR					A
1011	B					LD ww, (mn)				OTDM	OTDMR	OUTD	OTDR					B
1100	C	TST g				MLT ww												C
1101	D					RETI												D
1110	E						IM 2											E
1111	F					LD R, A	LD A, R	RLD										F
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
		C	E	L	A	C	E	L	A									
		g (LO = 8 to F)																

■ Bus Cycle Conditions

A hyphen in the Address column indicates that address output is undefined. A "Z" in the Data column indicates that the data pin is at high impedance.

Instruction	Machine Cycle	States	Address	Data	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{LIR}$	$\overline{HALT}$	ST
ADD HL, ww	MC$_1$	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC$_2$ to MC$_3$	TiTiTiTi	—	Z	1	1	1	1	1	1	1
ADD IX, ww ADD IY, yy	MC$_1$	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	$T_1T_2T_3$	2nd op-code address	2nd op-code	0	1	0	1	0	1	1
	MC$_3$ to MC$_6$	TiTiTiTi	—	Z	1	1	1	1	1	1	1
ADC HL, ww SBC HL, ww	MC$_1$	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	$T_1T_2T_3$	2nd op-code address	2nd op-code	0	1	0	1	0	1	1
	MC$_3$ to MC$_6$	TiTiTiTi	—	Z	1	1	1	1	1	1	1
ADD A, g ADC A, g SUB g SBC A, g AND g OR g XOR g CP g	MC$_1$	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	Ti	—	Z	1	1	1	1	1	1	1
ADD A, m ADC A, m SUB m SBC A, m AND m OR m XOR m CP m	MC$_1$	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	$T_1T_2T_3$	2nd operand address	2nd op-code	0	1	0	1	1	1	1

Instruction	Machine Cycle	States	Address	Data	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{LIR}$	$\overline{HALT}$	ST
ADD A, (HL) ADC A, (HL) SUB (HL) SBC A, (HL) AND (HL) OR (HL) XOR (HL) CP (HL)	MC$_1$	T$_1$T$_2$T$_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	T$_1$T$_2$T$_3$	HL	Data	0	1	0	1	1	1	1
ADD A, (IX + d) ADD A, (IY + d) ADC A, (IX + d) ADC A, (IY + d) SUB (IX + d) SUB (IY + d) SBC A, (IX + d) SBC A, (IY + d) AND (IX + d) AND (IY + d) OR (IX + d) OR (IY + d) XOR (IX + d) XOR (IY + d) CP (IX + d) CP (IY + d)	MC$_1$	T$_1$T$_2$T$_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	T$_1$T$_2$T$_3$	2nd op-code address	2nd op-code	0	1	0	1	0	1	1
	MC$_3$	T$_1$T$_2$T$_3$	1st operand address	d	0	1	0	1	1	1	1
	MC$_4$ to MC$_5$	TiTi	—	Z	1	1	1	1	1	1	1
	MC$_6$	T$_1$T$_2$T$_3$	IX + d IY + d	Data	0	1	0	1	1	1	1
BIT b, g	MC$_1$	T$_1$T$_2$T$_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	T$_1$T$_2$T$_3$	2nd op-code address	2nd op-code	0	1	0	1	0	1	1
BIT b, (HL)	MC$_1$	T$_1$T$_2$T$_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	T$_1$T$_2$T$_3$	2nd op-code address	2nd op-code	0	1	0	1	0	1	1
	MC$_3$	T$_1$T$_2$T$_3$	HL	Data	0	1	0	1	1	1	1

Instruction	Machine Cycle	States	Address	Data	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{LIR}$	$\overline{HALT}$	ST
BIT b, (IX + d) BIT b, (IY + d)	MC_1	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code address	2nd op-code	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	1st operand address	d	0	1	0	1	1	1	1
	MC_4	$T_1T_2T_3$	3rd op-code address	3rd op-code	0	1	0	1	0	1	1
	MC_5	$T_1T_2T_3$	IX + d IY + d	Data	0	1	0	1	1	1	1
CALL mn	MC_1	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	1st operand address	n	0	1	0	1	1	1	1
	MC_3	$T_1T_2T_3$	2nd operand address	m	0	1	0	1	1	1	1
	MC_4	Ti	—	Z	1	1	1	1	1	1	1
	MC_5	$T_1T_2T_3$	SP – 1	PCH	1	0	0	1	1	1	1
	MC_6	$T_1T_2T_3$	SP – 2	PCL	1	0	0	1	1	1	1
CALL f, mn (If condition is false)	MC_1	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	1st operand address	n	0	1	0	1	1	1	1
CALL f, mn (If condition is false)	MC_1	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	1st operand address	n	0	1	0	1	1	1	1
	MC_3	$T_1T_2T_3$	2nd operand address	m	0	1	0	1	1	1	1
	MC_4	Ti	—	Z	1	1	1	1	1	1	1
	MC_5	$T_1T_2T_3$	SP – 1	PCH	1	0	0	1	1	1	1
	MC_6	$T_1T_2T_3$	SP – 2	PCL	1	0	0	1	1	1	1

Instruction	Machine Cycle	States	Address	Data	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{LIR}$	$\overline{HALT}$	ST
CCF	MC$_1$	T$_1$T$_2$T$_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
CPI CPD	MC$_1$	T$_1$T$_2$T$_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	T$_1$T$_2$T$_3$	2nd op-code address	2nd op-code	0	1	0	1	0	1	1
	MC$_3$	T$_1$T$_2$T$_3$	HL	Data	0	1	0	1	1	1	1
	MC$_4$ to MC$_6$	TiTiTi	—	Z	1	1	1	1	1	1	1
CPIR CPDR (If BCR ≠ 0 and Ar ≠ (HL)$_M$)	MC$_1$	T$_1$T$_2$T$_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	T$_1$T$_2$T$_3$	2nd op-code address	2nd op-code	0	1	0	1	0	1	1
	MC$_3$	T$_1$T$_2$T$_3$	HL	Data	0	1	0	1	1	1	1
	MC$_4$ to MC$_8$	TiTiTiTiTi	—	Z	1	1	1	1	1	1	1
CPIR CPDR (If BCR = 0 or Ar = (HL)$_M$)	MC$_1$	T$_1$T$_2$T$_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	T$_1$T$_2$T$_3$	2nd op-code address	2nd op-code	0	1	0	1	0	1	1
	MC$_3$	T$_1$T$_2$T$_3$	HL	Data	0	1	0	1	1	1	1
	MC$_4$ to MC$_6$	TiTiTiTiTi	—	Z	1	1	1	1	1	1	1
CPL	MC$_1$	T$_1$T$_2$T$_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
DAA	MC$_1$	T$_1$T$_2$T$_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	Ti	—	Z	1	1	1	1	1	1	1
DI[Note]	MC$_1$	T$_1$T$_2$T$_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0

Note: Interrupts are not detected at the end of the DI instruction.

Instruction	Machine Cycle	States	Address	Data	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{LIR}$	$\overline{HALT}$	ST
DJNZ j (If Br ≠ 0)	MC_1	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC_2	Ti[*1]	—	Z	1	1	1	1	1	1	1
	MC_3	$T_1T_2T_3$	1st operand address	j – 2	0	1	0	1	1	1	1
	MC_4 to MC_5	TiTi	—	Z	1	1	1	1	1	1	1
DJNZ j (If Br = 0)	MC_1	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC_2	Ti[*1]	—	Z	1	1	1	1	1	1	1
	MC_3	$T_1T_2T_3$	1st operand address	j – 2	0	1	0	1	1	1	1
EI[*2]	MC_1	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
EX DE, HL EXX	MC_1	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
EX AF, AF'	MC_1	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC_2	Ti	—	Z	1	1	1	1	1	1	1
EX (SP), HL	MC_1	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	SP	Data	0	1	0	1	1	1	1
	MC_3	$T_1T_2T_3$	SP + 1	Data	0	1	0	1	1	1	1
	MC_4	Ti	—	Z	1	1	1	1	1	1	1
	MC_5	$T_1T_2T_3$	SP + 1	H	1	0	0	1	1	1	1
	MC_6	$T_1T_2T_3$	SP	L	1	0	0	1	1	1	1

Notes: 1. Immediately after this state, DMA, refresh, and bus release cannot be executed. Any such requests are ignored.

2. Interrupts are not detected at the end of the EI instruction.

Instruction	Machine Cycle	States	Address	Data	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{LIR}$	$\overline{HALT}$	ST
EX (SP), IX EX (SP), IY	MC$_1$	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	$T_1T_2T_3$	2nd op-code address	2nd op-code	0	1	0	1	0	1	1
	MC$_3$	$T_1T_2T_3$	SP	Data	0	1	0	1	1	1	1
	MC$_4$	$T_1T_2T_3$	SP + 1	Data	0	1	0	1	1	1	1
	MC$_5$	Ti	—	Z	1	1	1	1	1	1	1
	MC$_6$	$T_1T_2T_3$	SP + 1	IXH IYH	1	0	0	1	1	1	1
	MC$_7$	$T_1T_2T_3$	SP	IXL IYL	1	0	0	1	1	1	1
HALT	MC$_1$	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	—	—	Next op-code address	Next op-code	0	1	0	1	0	0	0
IM 0 IM 1 IM 2	MC$_1$	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	$T_1T_2T_3$	2nd op-code address	2nd op-code	0	1	0	1	0	1	1
INC g DEC g	MC$_1$	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	Ti	—	Z	1	1	1	1	1	1	1
INC (HL) DEC (HL)	MC$_1$	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	$T_1T_2T_3$	HL	Data	0	1	0	1	1	1	1
	MC$_3$	Ti	—	Z	1	1	1	1	1	1	1
	MC$_4$	$T_1T_2T_3$	HL	Data	1	1	1	1	1	1	1
INC (IX + d) INC (IY + d) DEC (IX + d) DEC (IY + d)	MC$_1$	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	$T_1T_2T_3$	2nd op-code address	2nd op-code	0	1	0	1	0	1	1
	MC$_3$	Ti	1st operand address	d	0	1	0	1	1	1	1

Instruction	Machine Cycle	States	Address	Data	$\overline{\text{RD}}$	$\overline{\text{WR}}$	$\overline{\text{ME}}$	$\overline{\text{IOE}}$	$\overline{\text{LIR}}$	$\overline{\text{HALT}}$	ST
INC (IX + d) INC (IY + d) DEC (IX + d) DEC (IY + d)	MC$_4$ to MC$_5$	TiTi	—	Z	1	1	1	1	1	1	1
	MC$_6$	T$_1$T$_2$T$_3$	IX + d IY + d	Data	0	1	0	1	1	1	1
	MC$_7$	Ti	—	Z	1	1	1	1	1	1	1
	MC$_8$	T$_1$T$_2$T$_3$	IX + d IY + d	Data	1	0	0	1	1	1	1
INC ww DEC ww	MC$_1$	T$_1$T$_2$T$_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	Ti	—	Z	1	1	1	1	1	1	1
INC IX INC IY DEC IX INC IY	MC$_1$	T$_1$T$_2$T$_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	T$_1$T$_2$T$_3$	2nd op-code address	2nd op-code	0	1	0	1	0	1	1
	MC$_3$	Ti	—	Z	1	1	1	1	1	1	1
IN A, (m)	MC$_1$	T$_1$T$_2$T$_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	T$_1$T$_2$T$_3$	1st operand address	m	0	1	0	1	1	1	1
	MC$_3$	T$_1$T$_2$T$_3$	m to A$_0$–A$_7$ A to A$_8$–A$_{15}$	Data	0	1	1	0	1	1	1
IN g, (C)	MC$_1$	T$_1$T$_2$T$_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	T$_1$T$_2$T$_3$	2nd op-code address	2nd op-code	0	1	0	1	0	1	1
	MC$_3$	T$_1$T$_2$T$_3$	BC	Data	0	1	1	0	1	1	1
IN0 g, (m)	MC$_1$	T$_1$T$_2$T$_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	T$_1$T$_2$T$_3$	2nd op-code address	2nd op-code	0	1	0	1	0	1	1
	MC$_3$	T$_1$T$_2$T$_3$	1st operand address	m	0	1	0	1	1	1	1
	MC$_4$	T$_1$T$_2$T$_3$	m to A$_0$–A$_7$ 00H to A$_8$–A$_{15}$	Data	0	1	1	0	1	1	1

Instruction	Machine Cycle	States	Address	Data	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{LIR}$	$\overline{HALT}$	ST
INI IND	MC_1	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code address	2nd op-code	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	BC	Data	0	1	1	0	1	1	1
	MC_4	$T_1T_2T_3$	HL	Data	1	0	0	1	1	1	1
INIR INDR (If Br ≠ 0)	MC_1	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code address	2nd op-code	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	BC	Data	0	1	1	0	1	1	1
	MC_4	$T_1T_2T_3$	HL	Data	1	0	0	1	1	1	1
	MC_5 to MC_6	TiTi	—	Z	1	1	1	1	1	1	1
INIR INDR (If Br = 0)	MC_1	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code address	2nd op-code	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	BC	Data	0	1	1	0	1	1	1
	MC_4	$T_1T_2T_3$	HL	Data	1	0	0	1	1	1	1
JP mn	MC_1	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	1st operand address	n	0	1	0	1	1	1	1
	MC_3	$T_1T_2T_3$	2nd operand address	m	0	1	0	1	1	1	1
JP f, mn (If f is false)	MC_1	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	1st operand address	n	0	1	0	1	1	1	1

Instruction	Machine Cycle	States	Address	Data	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{LIR}$	$\overline{HALT}$	ST
JP f, mn (If is true)	MC_1	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	1st operand address	n	0	1	0	1	1	1	1
	MC_3	$T_1T_2T_3$	2nd operand address	m	0	1	1	1	1	1	1
JP (HL)	MC_1	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
JP (IX) JP (IY)	MC_1	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code address	2nd op-code	0	1	0	1	0	1	1
JR j	MC_1	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	1st operand address	j – 2	0	1	0	1	1	1	1
	MC_3 to MC_4	TiTi	—	Z	1	1	1	1	1	1	1
JR C, j JR NC, j JR Z, j JR NZ, j (If condition is false)	MC_1	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	1st operand address	j – 2	0	1	0	1	1	1	1
JR C, j JR NC, j JR Z, j JR NZ, j (If condition is true)	MC_1	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	1st operand address	j – 2	0	1	0	1	1	1	1
	MC_3 to MC_4	TiTi	—	Z	1	1	1	1	1	1	1
LD g, g’	MC_1	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC_2	Ti	—	Z	1	1	1	1	1	1	1
LD g, m	MC_1	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	1st operand address	m	0	1	0	1	1	1	1
LD g, (HL)	MC_1	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	HL	Data	0	1	0	1	1	1	1

Instruction	Machine Cycle	States	Address	Data	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{LIR}$	$\overline{HALT}$	ST
LD g, (IX + d) LD g, (IY + d)	MC$_1$	T$_1$T$_2$T$_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	T$_1$T$_2$T$_3$	2nd op-code address	2nd op-code	0	1	0	1	0	1	1
	MC$_3$	T$_1$T$_2$T$_3$	1st operand address	d	0	1	0	1	1	1	1
	MC$_4$ to MC$_5$	TiTi	—	Z	1	1	1	1	1	1	1
	MC$_6$	T$_1$T$_2$T$_3$	IX + d IY + d	Data	0	1	0	1	1	1	1
LD (HL), g	MC$_1$	T$_1$T$_2$T$_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	Ti	—	Z	1	1	1	1	1	1	1
	MC$_3$	T$_1$T$_2$T$_3$	HL	g	1	0	0	1	1	1	1
LD (IX + d), g LD (IY + d), g	MC$_1$	T$_1$T$_2$T$_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	T$_1$T$_2$T$_3$	2nd op-code address	2nd op-code	0	1	0	1	0	1	1
	MC$_3$	T$_1$T$_2$T$_3$	1st operand address	d	0	1	0	1	1	1	1
	MC$_4$ to MC$_6$	TiTiTi	—	Z	1	1	1	1	1	1	1
	MC$_7$	T$_1$T$_2$T$_3$	IX + d IY + d	g	1	0	0	1	1	1	1
LD (HL), m	MC$_1$	T$_1$T$_2$T$_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	T$_1$T$_2$T$_3$	1st operand address	m	0	1	0	1	1	1	1
	MC$_3$	T$_1$T$_2$T$_3$	HL	Data	1	0	0	1	1	1	1
LD (IX + d), m LD (IY + d), m	MC$_1$	T$_1$T$_2$T$_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	T$_1$T$_2$T$_3$	2nd op-code address	2nd op-code	0	1	0	1	0	1	1
	MC$_3$	T$_1$T$_2$T$_3$	1st operand address	d	0	1	0	1	1	1	1
	MC$_4$	T$_1$T$_2$T$_3$	2nd operand address	m	0	1	0	1	1	1	1
	MC$_5$	T$_1$T$_2$T$_3$	IX + d IY + d	Data	1	0	0	1	1	1	1

Instruction	Machine Cycle	States	Address	Data	$\overline{\text{RD}}$	$\overline{\text{WR}}$	$\overline{\text{ME}}$	$\overline{\text{IOE}}$	$\overline{\text{LIR}}$	$\overline{\text{HALT}}$	ST
LD A, (BC) LD A, (DE)	MC$_1$	T$_1$T$_2$T$_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	T$_1$T$_2$T$_3$	BC DE	Data	0	1	0	1	1	1	1
LD A, (mn)	MC$_1$	T$_1$T$_2$T$_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	T$_1$T$_2$T$_3$	1st operand address	n	0	1	0	1	1	1	1
	MC$_3$	T$_1$T$_2$T$_3$	2nd operand address	m	0	1	0	1	1	1	1
	MC$_4$	T$_1$T$_2$T$_3$	mn	Data	1	1	1	1	1	1	1
LD (BC), A LD (DE), A	MC$_1$	T$_1$T$_2$T$_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	Ti	—	Z	1	1	1	1	1	1	1
	MC$_3$	T$_1$T$_2$T$_3$	BC DE	A	1	0	0	1	1	1	1
LD (mn), A	MC$_1$	T$_1$T$_2$T$_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	T$_1$T$_2$T$_3$	1st operand address	n	0	1	0	1	1	1	1
	MC$_3$	T$_1$T$_2$T$_3$	2nd op-code address	m	0	1	0	1	1	1	1
	MC$_4$	Ti	—	Z	1	1	1	1	1	1	1
	MC$_5$	T$_1$T$_2$T$_3$	mn	A	1	0	0	1	1	1	1
LD A, I[Note] LD A, R[Note] LD I, A LD R, A	MC$_1$	T$_1$T$_2$T$_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	T$_1$T$_2$T$_3$	2nd op-code address	2nd op-code	0	1	0	1	0	1	1
LD ww, mn	MC$_1$	T$_1$T$_2$T$_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	T$_1$T$_2$T$_3$	1st operand address	n	0	1	0	1	1	1	1
	MC$_3$	T$_1$T$_2$T$_3$	2nd op-code address	m	0	1	0	1	1	1	1

Note: Interrutps are not detected at the end of the LD A, I or LD A, R instruction.

Instruction	Machine Cycle	States	Address	Data	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{LIR}$	$\overline{HALT}$	ST
LD IX, mn LD IY, mn	MC_1	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code address	2nd op-code	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	1st operand address	n	0	1	0	1	1	1	1
	MC_4	$T_1T_2T_3$	2nd operand address	m	0	1	0	1	1	1	1
LD HL, (mn)	MC_1	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	1st operand address	n	0	1	0	1	1	1	1
	MC_3	$T_1T_2T_3$	2nd operand address	m	0	1	0	1	1	1	1
	MC_4	$T_1T_2T_3$	mn	Data	0	1	0	1	1	1	1
	MC_5	$T_1T_2T_3$	mn + 1	Data	0	1	0	1	1	1	1
LD ww, (mn)	MC_1	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code address	2nd op-code	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	1st operand address	n	0	1	0	1	1	1	1
	MC_4	$T_1T_2T_3$	2nd operand address	m	0	1	0	1	1	1	1
	MC_5	$T_1T_2T_3$	mn	Data	0	1	0	1	1	1	1
	MC_6	$T_1T_2T_3$	mn + 1	Data	0	1	0	1	1	1	1
LD IX, (mn) LD IY, (mn)	MC_1	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code address	2nd op-code	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	1st operand address	n	0	1	0	1	1	1	1
	MC_4	$T_1T_2T_3$	2nd operand address	m	0	1	0	1	1	1	1
	MC_5	$T_1T_2T_3$	mn	Data	0	1	0	1	1	1	1
	MC_6	$T_1T_2T_3$	mn + 1	Data	0	1	0	1	1	1	1

Instruction	Machine Cycle	States	Address	Data	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{LIR}$	$\overline{HALT}$	ST
LD (mn), HL	MC_1	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	1st operand address	n	0	1	0	1	1	1	1
	MC_3	$T_1T_2T_3$	2nd operand address	m	0	1	0	1	1	1	1
	MC_4	Ti	—	Z	1	1	1	1	1	1	1
	MC_5	$T_1T_2T_3$	mn	L	1	0	0	1	1	1	1
	MC_6	$T_1T_2T_3$	mn + 1	H	1	0	0	1	1	1	1
LD (mn), ww	MC_1	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code address	2nd op-code	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	1st operand address	n	0	1	0	1	1	1	1
	MC_4	$T_1T_2T_3$	2nd operand address	m	0	1	0	1	1	1	1
	MC_5	Ti	—	Z	1	1	1	1	1	1	1
	MC_6	$T_1T_2T_3$	mn	wwL	1	0	0	1	1	1	1
	MC_7	$T_1T_2T_3$	mn + 1	wwH	1	0	0	1	1	1	1
LD (mn), IX LD (mn), IY	MC_1	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code address	2nd op-code	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	1st operand address	n	0	1	0	1	1	1	1
	MC_4	$T_1T_2T_3$	2nd operand address	m	0	1	0	1	1	1	1
	MC_5	Ti	—	Z	1	1	1	1	1	1	1
	MC_6	$T_1T_2T_3$	mn	IXL IYL	1	0	0	1	1	1	1
	MC_7	$T_1T_2T_3$	mn + 1	IXH IYH	1	0	0	1	1	1	1

Instruction	Machine Cycle	States	Address	Data	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{LIR}$	$\overline{HALT}$	ST
LD SP, HL	MC_1	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC_2	Ti	—	Z	1	1	1	1	0	1	1
LD SP, IX LD SP, IY	MC_1	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code address	2nd op-code	0	1	0	1	0	1	1
	MC_3	Ti	—	Z	1	1	1	1	1	1	1
LDI LDD	MC_1	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code address	2nd op-code	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	HL	Data	0	1	0	1	1	1	1
	MC_4	$T_1T_2T_3$	DE	Data	1	0	0	1	1	1	1
LDIR LDDR (If BCR ≠ 0)	MC_1	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code address	2nd op-code	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	HL	Data	0	1	0	1	1	1	1
	MC_4	$T_1T_2T_3$	DE	Data	1	0	0	1	1	1	1
	MC_5 to MC_6	TiTi	—	Z	1	1	1	1	1	1	1
LDIR LDDR (If BCR = 0)	MC_1	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code address	2nd op-code	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	HL	Data	0	1	0	1	1	1	1
	MC_4	$T_1T_2T_3$	DE	Data	1	0	0	1	1	1	1
MLT ww	MC_1	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code address	2nd op-code	0	1	0	1	0	1	1
	MC_3 to MC_{13}	TiTiTiTi TiTiTiTi TiTiTi	—	Z	1	1	1	1	1	1	1

Instruction	Machine Cycle	States	Address	Data	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{LIR}$	$\overline{HALT}$	ST
NEG	MC_1	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code address	2nd op-code	0	1	0	1	0	1	1
NOP	MC_1	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
OUT (m), A	MC_1	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	1st operand address	m	0	1	0	1	1	1	1
	MC_3	Ti	—	Z	1	1	1	1	1	1	1
	MC_4	$T_1T_2T_3$	m to A_0–A_7 A to A_8–A_{15}	A	1	0	1	0	1	1	1
OUT (C), g	MC_1	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code address	2nd op-code	0	1	0	1	0	1	1
	MC_3	Ti	—	Z	1	1	1	1	1	1	1
	MC_4	$T_1T_2T_3$	BC	g	1	0	1	0	1	1	1
OUT0 (m), g	MC_1	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code address	2nd op-code	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	1st operand address	m	0	1	0	1	1	1	1
	MC_4	Ti	—	Z	1	1	1	1	1	1	1
	MC_5	$T_1T_2T_3$	m to A_0–A_7 00H to A_8–A_{15}	g	1	0	1	0	1	1	1

Instruction	Machine Cycle	States	Address	Data	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{LIR}$	$\overline{HALT}$	ST
OTIM OTDM	MC$_1$	T$_1$T$_2$T$_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	T$_1$T$_2$T$_3$	2nd op-code address	2nd op-code	0	1	0	1	0	1	1
	MC$_3$	Ti	—	Z	1	1	1	1	1	1	1
	MC$_4$	T$_1$T$_2$T$_3$	HL	Data	0	1	0	1	1	1	1
	MC$_5$	T$_1$T$_2$T$_3$	C to A$_0$–A$_7$ 00H to A$_8$–A$_{15}$	Data	1	0	1	0	1	1	1
	MC$_6$	Ti	—	Z	1	1	1	1	1	1	1
OTIMR OTDMR (If Br ≠ 0)	MC$_1$	T$_1$T$_2$T$_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	T$_1$T$_2$T$_3$	2nd op-code address	2nd op-code	0	1	0	1	0	1	1
	MC$_3$	Ti	—	Z	1	1	1	1	1	1	1
	MC$_4$	T$_1$T$_2$T$_3$	HL	Data	0	1	0	1	1	1	1
	MC$_5$	T$_1$T$_2$T$_3$	C to A$_0$–A$_7$ 00H to A$_8$–A$_{15}$	Data	1	0	1	0	1	1	1
	MC$_6$ to MC$_8$	TiTiTi	—	Z	1	1	1	1	1	1	1
OTIMR OTDMR (If Br = 0)	MC$_1$	T$_1$T$_2$T$_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	T$_1$T$_2$T$_3$	2nd op-code address	2nd op-code	0	1	0	1	0	1	1
	MC$_3$	Ti	—	Z	1	1	1	1	1	1	1
	MC$_4$	T$_1$T$_2$T$_3$	HL	Data	0	1	0	1	1	1	1
	MC$_5$	T$_1$T$_2$T$_3$	C to A$_0$–A$_7$ 00H to A$_8$–A$_{15}$	Data	1	0	1	0	1	1	1
	MC$_6$ to	Ti	—	Z	1	1	1	1	1	1	1

Instruction	Machine Cycle	States	Address	Data	$\overline{\text{RD}}$	$\overline{\text{WR}}$	$\overline{\text{ME}}$	$\overline{\text{IOE}}$	$\overline{\text{LIR}}$	$\overline{\text{HALT}}$	ST
OUTI OUTD	MC$_1$	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	$T_1T_2T_3$	2nd op-code address	2nd op-code	0	1	0	1	0	1	1
	MC$_3$	$T_1T_2T_3$	HL	Data	0	1	0	1	1	1	1
	MC$_4$	$T_1T_2T_3$	BC	Data	1	0	1	0	1	1	1
OTIR OTDR (If Br ≠ 0)	MC$_1$	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	$T_1T_2T_3$	2nd op-code address	2nd op-code	0	1	0	1	0	1	1
	MC$_3$	$T_1T_2T_3$	HL	Data	0	1	0	1	1	1	1
	MC$_4$	$T_1T_2T_3$	BC	Data	1	0	1	0	1	1	1
	MC$_5$ to MC$_6$	TiTi	—	Z	1	1	1	1	1	1	1
OTIR OTDR (If Br = 0)	MC$_1$	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	$T_1T_2T_3$	2nd op-code address	2nd op-code	0	1	0	1	0	1	1
	MC$_3$	$T_1T_2T_3$	HL	Data	0	1	0	1	1	1	1
	MC$_4$	$T_1T_2T_3$	BC	Data	1	0	1	0	1	1	1
POP zz	MC$_1$	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	$T_1T_2T_3$	SP	Data	0	1	0	1	1	1	1
	MC$_3$	$T_1T_2T_3$	SP + 1	Data	0	1	0	1	1	1	1
POP IX POP IY	MC$_1$	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC$_2$	$T_1T_2T_3$	2nd op-code address	2nd op-code	0	1	0	1	0	1	1
	MC$_3$	$T_1T_2T_3$	SP	Data	0	1	0	1	1	1	1
	MC$_4$	$T_1T_2T_3$	SP + 1	Data	0	1	0	1	1	1	1

Instruction	Machine Cycle	States	Address	Data	$\overline{\text{RD}}$	$\overline{\text{WR}}$	$\overline{\text{ME}}$	$\overline{\text{IOE}}$	$\overline{\text{LIR}}$	$\overline{\text{HALT}}$	ST
PUSH zz	MC_1	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC_2 to MC_3	TiTi	—	Z	1	1	1	1	1	1	1
	MC_4	$T_1T_2T_3$	SP – 1	zzH	1	0	0	1	1	1	1
	MC_5	$T_1T_2T_3$	SP – 1	zzL	1	0	0	1	1	1	1
PUSH IX PUSH IY	MC_1	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code address	2nd op-code	0	1	0	1	0	1	1
	MC_3 to MC_4	TiTi	—	Z	1	1	1	1	1	1	1
	MC_5	$T_1T_2T_3$	SP – 1	IXH IYH	1	0	0	1	1	1	1
	MC_6	$T_1T_2T_3$	SP – 2	IXL IYL	1	0	0	1	1	1	1
RET	MC_1	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	SP	Data	0	1	0	1	1	1	1
	MC_3	$T_1T_2T_3$	SP + 2	Data	0	1	0	1	1	1	1
RET f (If condition is false)	MC_1	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC_2 to MC_3	TiTi	—	Z	1	1	1	1	1	1	1
RET f (If condition is true)	MC_1	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC_2	Ti	—	Z	1	1	1	1	1	1	1
	MC_3	$T_1T_2T_3$	SP	Data	0	1	0	1	1	1	1
	MC_4	$T_1T_2T_3$	SP + 1	Data	0	1	0	1	1	1	1

Instruction	Machine Cycle	States	Address	Data	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{LIR}$	$\overline{HALT}$	ST
RETI (R1) RETN	MC_1	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code address	2nd op-code	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	SP	Data	0	1	0	1	1	1	1
	MC_4	$T_1T_2T_3$	SP + 1	Data	0	1	0	1	1	1	1
RETI (Z)	MC_1	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0^{Note} 1	1	0
	MC_2	$T_1T_2T_3$	2nd op-code address	2nd op-code	0	1	0	1	0^{Note} 1	1	1
	MC_3 to MC_5	TiTiTi	—	Z	1	1	1	1	1^{Note} 1	1	1
	MC_6	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0^{Note} 0	1	1
	MC_7	Ti	—	Z	1	1	1	1	1^{Note} 1	1	1
	MC_8	$T_1T_2T_3$	2nd op-code address	2nd op-code	0	1	0	1	0^{Note} 0	1	1
	MC_9	$T_1T_2T_3$	SP	Data	0	1	0	1	1^{Note} 1	1	1
	MC_{10}	$T_1T_2T_3$	SP + 1	Data	0	1	0	1	1^{Note} 1	1	1
RLCA RLA RRCA RRA	MC_1	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
RLC g RL g RRC g RR g SLA g SRA g SRL g	MC_1	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code address	2nd op-code	0	1	0	1	0	1	1
	MC_3	Ti	—	Z	1	1	1	1	1	1	1

Note: Upper value is $\overline{LIR}$ status when LIRE = 1. Lower value is $\overline{LIR}$ status when LIRE = 0.

Instruction	Machine Cycle	States	Address	Data	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{LIR}$	$\overline{HALT}$	ST
RLC (HL) RL (HL) RRC (HL) RR (HL) SLA (HL) SRA (HL) SRL (HL)	MC_1	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code address	2nd op-code	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	HL	Data	0	1	0	1	1	1	1
	MC_4	Ti	—	Z	1	1	1	1	1	1	1
	MC_5	$T_1T_2T_3$	HL	Data	1	0	0	1	1	1	1
RLC (IX + d) RLC (IY + d) RL (IX + d) RL (IY + d) RRC (IX + d) RRC (IY + d) RR (IX + d) RR (IY + d) SLA (IX + d) SLA (IY + d) SRA (IX + d) SRA (IY + d) SRL (IX + d) SRL (IY + d)	MC_1	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code address	2nd op-code	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	1st operand address	d	0	1	0	1	1	1	1
	MC_4	$T_1T_2T_3$	3rd op-code address	3rd op-code	0	1	0	1	0	1	1
	MC_5	$T_1T_2T_3$	IX + d IY + d	Data	0	1	0	1	1	1	1
	MC_6	Ti	—	Z	1	1	1	1	1	1	1
	MC_7	$T_1T_2T_3$	IX + d IY + d	Data	1	0	0	1	1	1	1
RLD RRD	MC_1	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code address	2nd op-code	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	HL	Data	0	1	0	1	1	1	1
	MC_4 to MC_7	TiTiTiTi	—	Z	1	1	1	1	1	1	1
	MC_8	$T_1T_2T_3$	HL	Data	1	0	0	1	1	1	1

Instruction	Machine Cycle	States	Address	Data	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{LIR}$	$\overline{HALT}$	ST
RST v	MC_1	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC_2 to MC_3	TiTi	—	Z	1	1	1	1	1	1	1
	MC_4	$T_1T_2T_3$	SP – 1	PCH	1	0	0	1	1	1	1
	MC_5	$T_1T_2T_3$	SP – 2	PCL	1	0	0	1	1	1	1
SCF	MC_1	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
SET b, g RES b, g	MC_1	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code address	2nd op-code	0	1	0	1	0	1	1
	MC_3	Ti	—	Z	1	1	1	1	1	1	1
SET b, (HL) RES b, (HL)	MC_1	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code address	2nd op-code	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	HL	Data	0	1	0	1	1	1	1
	MC_4	Ti	—	Z	1	1	1	1	1	1	1
	MC_5	$T_1T_2T_3$	HL	Data	1	0	0	1	1	1	1

SET b, (IX + d)
SET b, (IY + d)
RES b, (IX + d)
RES b, (IY + d)
	MC_1	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0

	MC_2	$T_1T_2T_3$	2nd op-code address	2nd op-code	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	1st operand address	d	0	1	0	1	1	1	1
	MC_4	$T_1T_2T_3$	3rd op-code address	3rd op-code	0	1	0	1	0	1	1
	MC_5	$T_1T_2T_3$	IX + d IY + d	Data	0	1	0	1	1	1	1
	MC_6	Ti	—	Z	1	1	1	1	1	1	1
	MC_7	$T_1T_2T_3$	IX + d IY + d	Data	1	0	0	1	1	1	1

Instruction	Machine Cycle	States	Address	Data	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{LIR}$	$\overline{HALT}$	ST
SLP	MC_1	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code address	2nd op-code	0	1	0	1	0	1	1
	—	—	FFFFFH	Z	1	1	1	1	1	0	1
TSTIO m	MC_1	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code address	2nd op-code	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	1st operand address	m	0	1	0	1	1	1	1
	MC_4	$T_1T_2T_3$	C to A_0–A_7 00H to A_8–A_{15}	Data	0	1	1	1	1	1	1
TST g	MC_1	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code address	2nd op-code	0	1	0	1	0	1	1
	MC_3	Ti	—	Z	1	1	1	1	1	1	1
TST m	MC_1	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code address.	2nd op-code	0	1	0	1	0	1	1
	MC_3	$T_1T_2T_3$	1st operand address	m	1	1	1	1	1	1	1
TST (HL)	MC_1	$T_1T_2T_3$	1st op-code address	1st op-code	0	1	0	1	0	1	0
	MC_2	$T_1T_2T_3$	2nd op-code address	2nd op-code	0	1	0	1	0	1	1
	MC_3	Ti	—	Z	1	1	1	1	1	1	1
	MC_4	$T_1T_2T_3$	HL	Data	0	1	0	1	1	1	1

Instruction	Machine Cycle	States	Address	Data	$\overline{\text{RD}}$	$\overline{\text{WR}}$	$\overline{\text{ME}}$	$\overline{\text{IOE}}$	$\overline{\text{LIR}}$	$\overline{\text{HALT}}$	ST
$\overline{\text{NMI}}$	MC_1	$T_1T_2T_3$	Next op-code address (PC)		0	1	0	1	0	1	0
	MC_2 to MC_3	TiTi	—	Z	1	1	1	1	1	1	1
	MC_4	$T_1T_2T_3$	SP – 1	PCH	1	0	0	1	1	1	1
	MC_5	$T_1T_2T_3$	SP – 2	PCL	1	0	0	1	1	1	1
$\overline{\text{INT}_0}$ mode 0 (RST inserted)	MC_1	$T_1T_2T_W$ T_WT_3	Next op-code address (PC)	1st op-code	1	1	1	0	0	1	0
	MC_2 to MC_3	TiTi	—	Z	1	1	1	1	1	1	1
	MC_4	$T_1T_2T_3$	SP – 1	PCH	1	0	0	1	1	1	1
	MC_5	$T_1T_2T_3$	SP – 2	PCL	1	0	0	1	1	1	1
$\overline{\text{INT}_0}$ mode 0 (CALL inserted)	MC_1	$T_1T_2T_W$ T_WT_3	Next op-code address (PC)	1st op-code	1	1	1	0	0	1	0
	MC_2	$T_1T_2T_3$	PC	n	0	1	0	1	1	1	1
	MC_3	$T_1T_2T_3$	PC + 1	m	0	1	0	1	1	1	1
	MC_4	Ti	—	Z	1	1	1	1	1	1	1
	MC_5	$T_1T_2T_3$	SP – 1	PC + 2 (H)	1	0	0	1	1	1	1
	MC_6	$T_1T_2T_3$	SP – 2	PC + 2 (L)	1	0	0	1	1	1	1
$\overline{\text{INT}_0}$ mode 1	MC_1	$T_1T_2T_W$ T_WT_3	Next op-code address (PC)		1	1	1	0	0	1	0
	MC_2	$T_1T_2T_3$	SP – 1	PCH	1	0	0	1	1	1	1
	MC_3	$T_1T_2T_3$	SP + 2	PCL	1	0	0	1	1	1	1
$\overline{\text{INT}_0}$ mode 2	MC_1	$T_1T_2T_W$ T_WT_3	Next op-code address (PC)	vector	1	1	1	0	0	1	0
	MC_2	Ti	—	Z	1	1	1	1	1	1	1
	MC_3	$T_1T_2T_3$	SP – 1	PCH	1	0	0	1	1	1	1
	MC_4	$T_1T_2T_3$	SP – 2	PCL	1	0	0	1	1	1	1
	MC_5	$T_1T_2T_3$	I, vector	Data	0	1	0	1	1	1	1
	MC_6	$T_1T_2T_3$	I, vector + 1	Data	0	1	0	1	1	1	1

Instruction	Machine Cycle	States	Address	Data	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{LIR}$	$\overline{HALT}$	ST
$\overline{INT_1}$, $\overline{INT}_2$, internal interrupts	MC_1	$T_1T_2T_W$ T_WT_3	Next op-code address (PC)		1	1	1	1	1	1	0
	MC_2	Ti	—	Z	1	1	1	1	1	1	1
	MC_3	$T_1T_2T_3$.	SP – 1	PCH	1	0	0	1	1	1	1
	MC_4	$T_1T_2T_3$	SP – 2	PCL	1	0	0	1	1	1	1
	MC_5	$T_1T_2T_3$	I, vector	Data	0	1	0	1	1	1	1
	MC_6	$T_1T_2T_3$	I, vector + 1	Data	0	1	0	1	1	1	1

■ Acceptable Requests in Each Mode

	Current Status							
Action Request	**Normal Mode (CPU Mode) (IOSTOP Mode)**	**Wait State**	**Refresh Cycle**	**Interrupt Acknowledge Cycle**	**DMA Cycle**	**Bus Release Mode**	**Sleep Mode**	**Standby Mode**
$\overline{\text{WAIT}}$	Accepted	Accepted	Not accepted	Accepted	Accepted	Not accepted	Not accepted	Not accepted
Refresh request (request for insertion of refresh cycle by on-chip refresh circuit)	Refresh cycle inserted at machine cycle break	Not accepted	Not accepted	Refresh cycle inserted at machine cycle break	Refresh cycle inserted at machine cycle break	Not accepted	Not accepted	Not accepted
$\overline{\text{DREQ}_0}$ $\overline{\text{DREQ}_1}$	DMA cycle inserted at machine cycle break	Accepted, but DMA cycle not inserted until machine cycle break	Accepted * Following completion of refresh cycle, 1 machine cycle is executed and then DMA cycle is inserted.	DMA cycle inserted at machine cycle break	Accepted (Consult this manual for full details.)	Accepted * Following completion of bus release cycle, 1 machine cycle is executed and then DMA cycle is inserted.	Not accepted	Not accepted
$\overline{\text{BUSREQ}}$	Bus release mode entered at machine cycle break	Not accepted	Not accepted	Bus release mode entered at machine cycle break	Bus release mode entered at machine cycle break	Bus release mode continues	Accepted	Accepted

Note: * Not accepted when $\overline{\text{DREQ}_0}$ and $\overline{\text{DREQ}_1}$ are set for level detection.

		Current Status							
	Action Request	Normal Mode (CPU Mode) (IOSTOP Mode)	Wait State	Refresh Cycle	Interrupt Acknowledge Cycle	DMA Cycle	Bus Release Mode	Sleep Mode	Standby Mode
Interrupts	$\overline{INT_0}$, $\overline{INT_1}$, INT_2	Accepted in final machine cycle of instruction	Accepted in final machine cycle of instruction	Not accepted	Not accepted	Not accepted	Not accepted	Accepted, sleep mode exited and normal status recovered	Not accepted
	Internal I/O interrupt request	Accepted in final machine cycle of instruction	Accepted in final machine cycle of instruction	Not accepted	Not accepted	Not accepted	Not accepted	Accepted, sleep mode exited and normal status recovered	Not accepted
	$\overline{NMI}$	Accepted in final machine cycle of instruction	Accepted in final machine cycle of instruction	Not accepted	Not accepted **Accepted in final machine cycle of instruction following completion of acknowledge cycle.	Accepted, DMA suspended	Not accepted	Accepted, sleep mode exited and normal status recovered	Accepted, standby mode exited and normal status recovered. Note that IOSTOP bit remains set to 1.

Note: ** Accepted when $\overline{INT_0}$ is being used in mode 0. The NMI acknowledge cycle begins following execution of the instruction placed on the bus.

■ Request Priority Sequence

Three types of requests can be input to the CPU.

1. Requests that can be accepted and executed at the end of a state (WAIT)

2. Requests that can be accepted and executed at the end of a machine cycle (refresh requests, DMA requests, $\overline{\text{BUSREQ}}$)

3. Requests that can be accepted and executed at the end of an instruction (all interrupts)

   Basically, the priority sequence gives 1 the highest priority, and 3 the lowest.

   Within group 2 above, the descending priority sequence is:

   $\overline{\text{BUSREQ}}$
   Refresh request
   DMA request

   If a $\overline{\text{BUSREQ}}$ and refresh request are input simultaneously, the $\overline{\text{BUSREQ}}$ is given priority.

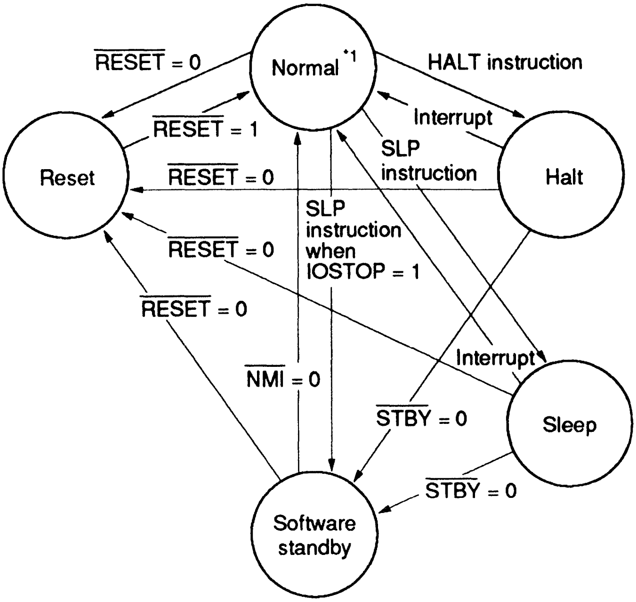
   For the priority sequence of the interrupts in group 3 above, see the section of this manual that details interrupt operations.

4. Among requests that are accepted and executed in the final machine cycle of an instruction, the descending priority sequence is:

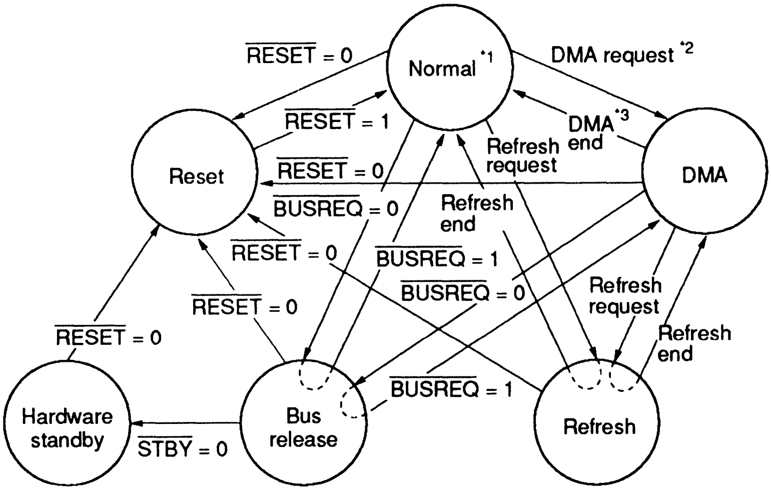
   Bus requests ($\overline{\text{BUSREQ}}$, refresh request, DMA request)

   Interrupt

■ Mode Transitions



Normal [*1]
$\overline{\text{RESET}} = 0$
HALT instruction
Interrupt
$\overline{\text{RESET}} = 1$
SLP instruction
Reset
$\overline{\text{RESET}} = 0$
Halt
SLP instruction when IOSTOP = 1
$\overline{\text{RESET}} = 0$
$\overline{\text{RESET}} = 0$
Interrupt
$\overline{\text{NMI}} = 0$
$\overline{\text{STBY}} = 0$
Sleep
$\overline{\text{STBY}} = 0$
Software standby



$\overline{\text{RESET}} = 0$
Normal [*1]
DMA request [*2]
DMA [*3] end
$\overline{\text{RESET}} = 1$
Refresh request
Reset
$\overline{\text{RESET}} = 0$
DMA
$\overline{\text{BUSREQ}} = 0$
Refresh end
$\overline{\text{RESET}} = 0$
$\overline{\text{BUSREQ}} = 1$
$\overline{\text{BUSREQ}} = 0$
Refresh request
$\overline{\text{RESET}} = 0$
Refresh end
$\overline{\text{RESET}} = 0$
$\overline{\text{BUSREQ}} = 1$
Hardware standby
Bus release
Refresh
$\overline{\text{STBY}} = 0$

Notes: 1. Normal: Normal CPU instruction execution mode

2. DMA request: $\overline{\text{DREQ}_0}$ or $\overline{\text{DREQ}_1}$ = 0 (for memory ↔ (memory-mapped) I/O transfers)
DE0 = 1 (for Memory ↔ Memory transfers)

3. DMA end: $\overline{\text{DREQ}_0}$ or $\overline{\text{DREQ}_1}$ = 1 (for memory ↔ (memory-mapped) I/O transfers)
BCR0 and BCR1 = 0000H (for all transfer modes)
$\overline{\text{NMI}}$ = 0 (for all transfer modes)

In addition to the above, the following mode transitions are also possible.

1. From halt to DMA, refresh, or bus release, and vice versa.

2. From sleep to bus release, and vice versa.

■ Status Signal List

The following list shows the status signal output for each mode.

Mode		$\overline{\text{LIR}}$	$\overline{\text{ME}}$	$\overline{\text{IOE}}$	$\overline{\text{RD}}$	$\overline{\text{WR}}$	$\overline{\text{REF}}$	$\overline{\text{HALT}}$	$\overline{\text{BUSACK}}$	ST	Address Bus	Data Bus
CPU operation	1st op code fetch	0	0	1	0	1	1	1	1	0	A	In
	Other op code fetch	0	0	1	0	1	1	1	1	1	A	In
	Memory read	1	0	1	0	1	1	1	1	1	A	In
	Memory write	1	0	1	1	0	1	1	1	1	A	Out
	I/O read	1	1	0	0	1	1	1	1	1	A	In
	I/O write	1	1	0	1	0	1	1	1	1	A	Out
	Internal operation	1	1	1	1	1	1	1	1	1	A	In
Refresh		1	0	1	1	1	0	1	1	*	A	In
Interrupt acknowl-edge (1st machine cycle)	$\overline{\text{NMI}}$	0	0	1	0	1	1	1	1	0	A	In
	$\overline{\text{INT}_0}$	0	1	0	1	1	1	1	1	0	A	In
	$\overline{\text{INT}_1}$, $\overline{\text{INT}_2}$, and internal interrupts	1	1	1	1	1	1	1	1	0	A	In
Bus release		1	Z	Z	Z	Z	1	1	0	*	Z	In
Halt		0	0	1	0	1	1	0	1	0	A	In
Sleep		1	1	1	1	1	1	0	1	1	1	In
Internal DMA	Memory read	1	0	1	0	1	1	*	1	0	A	In
	Memory write	1	0	1	1	0	1	*	1	0	A	Out
	I/O read	1	1	0	0	1	1	*	1	0	A	In
	I/O write	1	1	0	1	0	1	1	1	0	A	Out
Reset	1	1	1	1	1	1	1	1	1	1	Z	In

1 = high level; 0 = low level; * = undefined; A = any value; Z = high impedance; IN = input; OUT = output

Internal I/O Register Reference

The upper 8 bits of the I/O register address are all 0. The most significant bit of the lower 8 bits can be set using the IOA7 bit of the IO control register. The table below shows the addresses when IOA7 = 0.

<table>
<tr><th>Register</th><th>Address</th><th>Remarks</th></tr>
<tr><td>DMA source address register channel 0 L: SAR0L</td><td>20</td><td></td></tr>
<tr><td>DMA source address register channel 0 H: SAR0H</td><td>21</td><td></td></tr>
<tr><td>DMA source address register channel 0 B: SAR0B</td><td>22</td><td>Only bits 0, 1, 2, and 3 are used
<table>
<tr><th>A_{19},</th><th>A_{18},</th><th>A_{17},</th><th>A_{16}</th><th>DMA Transfer Request</th></tr>
<tr><td>×</td><td>×</td><td>0</td><td>0</td><td>$\overline{\text{DREQ}_0}$ (external)</td></tr>
<tr><td>×</td><td>×</td><td>0</td><td>1</td><td>Not used</td></tr>
<tr><td>×</td><td>×</td><td>1</td><td>0</td><td>Not used</td></tr>
<tr><td>×</td><td>×</td><td>1</td><td>1</td><td>Not used</td></tr>
</table></td></tr>
<tr><td>DMA destination address register channel 0 L: DAR0L</td><td>23</td><td></td></tr>
<tr><td>DMA destination address register channel 0 H: DAR0H</td><td>24</td><td></td></tr>
<tr><td>DMA destination address register channel 0 B: DAR0B</td><td>25</td><td>Only bits 0, 1, 2, and 3 are used
<table>
<tr><th>A_{19},</th><th>A_{18},</th><th>A_{17},</th><th>A_{16}</th><th>DMA Transfer Request</th></tr>
<tr><td>×</td><td>×</td><td>0</td><td>0</td><td>$\overline{\text{DREQ}_0}$ (external)</td></tr>
<tr><td>×</td><td>×</td><td>0</td><td>1</td><td>Not used</td></tr>
<tr><td>×</td><td>×</td><td>1</td><td>0</td><td>Not used</td></tr>
<tr><td>×</td><td>×</td><td>1</td><td>1</td><td>Not used</td></tr>
</table></td></tr>
<tr><td>DMA byte count register channel 0 L: BCR0L</td><td>26</td><td></td></tr>
<tr><td>DMA byte count register channel 0 H: BCR0H</td><td>27</td><td></td></tr>
</table>

<table>
<tr><th>Register</th><th>Address</th><th>Remarks</th></tr>
<tr><td>DMA memory address register channel 1 L: MAR1L</td><td>28</td><td></td></tr>
<tr><td>DMA memory address register channel 1 H: MAR1H</td><td>29</td><td></td></tr>
<tr><td>DMA memory address register channel 1 B: MAR1B</td><td>2A</td><td>Only bits 0, 1, 2, and 3 are used</td></tr>
<tr><td>DMA I/O address register channel 1 L: IAR1L</td><td>2B</td><td></td></tr>
<tr><td>DMA I/O address register channel 1 H: IAR1H</td><td>2C</td><td></td></tr>
<tr><td>DMA byte count register channel 1 L: BCR1L</td><td>2E</td><td></td></tr>
<tr><td>DMA byte count register channel 1 H: BCR1H</td><td>2F</td><td></td></tr>
<tr><td>DMA status register: DSTAT</td><td>30</td><td>
<table>
<tr><td>Bit</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td></td><td>DE1</td><td>DE0</td><td>$\overline{\text{DWE1}}$</td><td>$\overline{\text{DWE0}}$</td><td>DIE1</td><td>DIE0</td><td>—</td><td>DME</td></tr>
<tr><td>Initial value</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td></tr>
<tr><td>Read/Write</td><td>R/W</td><td>R/W</td><td>W</td><td>W</td><td>R/W</td><td>R/W</td><td>—</td><td>R</td></tr>
</table>
</td></tr>
<tr><td>DMA mode register: DMODE</td><td>31</td><td>
<table>
<tr><td>Bit</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td></td><td>—</td><td>—</td><td>DM1</td><td>DM0</td><td>SM1</td><td>SM0</td><td>MMOD</td><td>—</td></tr>
<tr><td>Initial value</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td></tr>
<tr><td>Read/Write</td><td>—</td><td>—</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>—</td></tr>
</table>
</td></tr>
<tr><td>DMA/WAIT control register: DCNTL</td><td>32</td><td>
<table>
<tr><td>Bit</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td></td><td>MWI1</td><td>MWI0</td><td>IWI1</td><td>IWI0</td><td>DMS1</td><td>DMS0</td><td>DIM1</td><td>DIM0</td></tr>
<tr><td>Initial value</td><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>Read/Write</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td></tr>
</table>
</td></tr>
<tr><td>Interrupt vector low register: IL</td><td>33</td><td>
<table>
<tr><td>Bit</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td></td><td>IL7</td><td>IL6</td><td>IL5</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td></tr>
<tr><td>Initial value</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td></tr>
<tr><td>Read/Write</td><td>R/W</td><td>R/W</td><td>R/W</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td></tr>
<tr><td></td><td colspan="3">Arbitrary values</td><td colspan="5">Fixed code</td></tr>
</table>
</td></tr>
</table>

Register	Address	Remarks								
INT/TRAP control register: ITC	34	Bit	7	6	5	4	3	2	1	0
			TRAP	UFO	—	—	—	ITE2	ITE1	ITE0
		Initial value	0	0	1	1	1	0	0	1
		Read/Write	R/W	R	—	—	—	R/W	R/W	R/W
Refresh control register: RCR	36	Bit	7	6	5	4	3	2	1	0
			REFE	REFW	—	—	—	—	CYC1	CYC0
		Initial value	1	1	1	1	1	1	0	0
		Read/Write	R/W	R/W	—	—	—	—	R/W	R/W
MMU common base register: CBR	38	Bit	7	6	5	4	3	2	1	0
			CB7	CB6	CB5	CB4	CB3	CB2	CB1	CB0
		Initial value	0	0	0	0	0	0	0	0
		Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
MMU bank base register: BBR	39	Bit	7	6	5	4	3	2	1	0
			BB7	BB6	BB5	BB4	BB3	BB2	BB1	BB0
		Initial value	0	0	0	0	0	0	0	0
		Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
MMU common/bank area register: CBAR	3A	Bit	7	6	5	4	3	2	1	0
			CA3	CA2	CA1	CA0	BA3	BA2	BA1	BA0
		Initial value	1	1	1	1	0	0	0	0
		Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Operating mode control register: OMCR	3E	Bit	7	6	5	4	3	2	1	0
			LIRE	$\overline{\text{LIRTE}}$	$\overline{\text{IOC}}$	—	—	—	—	—
		Initial value	1	1	1	1	1	1	1	1
		Read/Write	R/W	W	R/W	—	—	—	—	—
I/O control register: IOCR	3F	Bit	7	6	5	4	3	2	1	0
			IOAR	—	IOSTOP	—	—	—	—	—
		Initial value	0	1	0	1	1	1	1	1
		Read/Write	R/W	—	R/W	—	—	—	—	—

Register	Address	Remarks							
Free-running counter H: FRCH	40	Bit	7	6	5	4	3	2	1	0
			FRC15	FRC14	FRC13	FRC12	FRC11	FRC10	FRC9	FRC8
		Initial value	0	0	0	0	0	0	0	0
		Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Free-running counter L: FRCL	41	Bit	7	6	5	4	3	2	1	0
			FRC7	FRC6	FRC5	FRC4	FRC3	FRC2	FRC1	FRC0
		Initial value	0	0	0	0	0	0	0	0
		Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Timer control/status register 1: TCSR1	42	Bit	7	6	5	4	3	2	1	0
			ICF	OCF	TOF	EICI	EOCI	ETOI	IEDG	OLVL
		Initial value	0	0	0	0	0	0	0	0
		Read/Write	R	R	R	R/W	R/W	R/W	R/W	R/W
Output compare register 1 H: OCR1H	43	Bit	7	6	5	4	3	2	1	0
			OCR15	OCR14	OCR13	OCR12	OCR11	OCR10	OCR9	OCR8
		Initial value	1	1	1	1	1	1	1	1
		Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Output compare register 1 L: OCR1L	44	Bit	7	6	5	4	3	2	1	0
			OCR7	OCR6	OCR5	OCR4	OCR3	OCR2	OCR1	OCR0
		Initial value	1	1	1	1	1	1	1	1
		Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Input capture register H: ICRH[Note]	45	Bit	7	6	5	4	3	2	1	0
			ICR15	ICR14	ICR13	ICR12	ICR11	ICR10	ICR9	ICR8
		Initial value	*	*	*	*	*	*	*	*
		Read/Wnte	R	R	R	R	R	R	R	R
										* Undefined
Input capture register L: ICRL[Note]	46	Bit	7	6	5	4	3	2	1	0
			ICR7	ICR6	ICR5	ICR4	ICR3	ICR2	ICR1	ICR0
		Initial value	*	*	*	*	*	*	*	*
		Read/Wnte	R	R	R	R	R	R	R	R
										* Undefined

Note: ICRH and ICRL are not initialized by reset.

Register	Address	Remarks								
Transmit/receive control/status register A 0: TRCSRA0	47	Bit	7	6	5	4	3	2	1	0
			RDRF0	ORFE0	TDRE0	RIE0	RE0	TIE0	TE0	WU0
		Initial value	0	0	1	0	0	0	0	0
		Read/Write	R	R	R	R/W	R/W	R/W	R/W	R/W
Transmit/receive control/status register B 0: TRCSRB0	48	Bit	7	6	5	4	3	2	1	0
			RDRF0	ORFE0	TDRE0	PER0	—	PEN0	EOP0	SBL0
		Initial value	0	0	1	0	1	0	0	0
		Read/Write	R	R	R	R	—	R/W	R/W	R/W
Rate/mode control register 0: RMCR0	49	Bit	7	6	5	4	3	2	1	0
			—	—	SS02	CC02	CC01	CC00	SS01	SS00
		Initial value	1	1	0	0	0	0	0	0
		Read/Write	—	—	R/W	R/W	R/W	R/W	R/W	R/W
Receive data register 0: RDR0	4A									
Transmit data register 0: TDR0	4B									
Serial port control register: SCIPCR	4C	Bit	7	6	5	4	3	2	1	0
			SCKHL0	SOUTM0	SINM0	SCKM0	SCKHL1	SOUTM1	SINM1	SCKM1
		Initial value	0	0	0	0	0	0	0	0
		Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
A/D control register: ADCR	4D	Bit	7	6	5	4	3	2	1	0
			—	—	—	—	—	—	ANE	$\overline{\text{AVREF}}$
		Initial value	1	1	1	1	1	1	0	0
		Read/Write	—	—	—	—	—	—	R/W	R/W
A/D control/status register: ADCSR	4E	Bit	7	6	5	4	3	2	1	0
			ADEF	ADS	EADEI	CH2	CH1	CH0	ACS1	ACS0
		Initial value	0	0	0	0	0	0	0	0
		Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Register	Address	Remarks								
A/D result register: ADRR	4F	Bit	7	6	5	4	3	2	1	0
			ADRR7	ADRR6	ADRR5	ADRR4	ADRR3	ADRR2	ADRR1	ADRR0
		Initial value	*	*	*	*	*	*	*	*
		Read/Write	R	R	R	R	R	R	R	R
										* Undefined
Transmit/receive control/status register A 1: TRCSRA1	50	Bit	7	6	5	4	3	2	1	0
			RDRF1	ORFE1	TDRE1	RIE1	RE1	TIE1	TE1	WU1
		Initial value	0	0	1	0	0	0	0	0
		Read/Write	R	R	R	R/W	R/W	R/W	R/W	R/W
Transmit/receive control/status register B 1: TRCSRB1	51	Bit	7	6	5	4	3	2	1	0
			RDRF1	ORFE1	TDRE1	PER1	—	PEN1	EOP1	SBL1
		Initial value	0	0	1	0	1	0	0	0
		Read/Write	R	R	R	R	—	R/W	R/W	R/W
Rate/mode control register 1: RMCR1	52	Bit	7	6	5	4	3	2	1	0
			—	—	SS12	CC12	CC11	CC10	SS11	SS10
		Initial value	1	1	0	0	0	0	0	0
		Read/Write	—	—	R/W	R/W	R/W	R/W	R/W	R/W
Receive data register 1: RDR1	53									
Transmit data register 1: TDR1	54									
Timer 2 up-counter H: T2CNTH	55	Bit	7	6	5	4	3	2	1	0
			T2C15	T2C14	T2C13	T2C12	T2C11	T2C10	T2C9	T2C8
		Initial value	0	0	0	0	0	0	0	0
		Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Timer 2 up-counter L: T2CNTL	56	Bit	7	6	5	4	3	2	1	0
			T2C7	T2C6	T2C5	T2C4	T2C3	T2C2	T2C1	T2C0
		Initial value	0	0	0	0	0	0	0	0
		Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Register	Address	Remarks								
Timer 2 time constant register H: T2CONRH	57	Bit	7	6	5	4	3	2	1	0
			T2CN15	T2CN14	T2CN13	T2CN12	T2CN11	T2CN10	T2CN9	T2CN8
		Initial value	1	1	1	1	1	1	1	1
		Read/Write	W	W	W	W	W	W	W	W
Timer 2 time constant register L: T2CONRL	58	Bit	7	6	5	4	3	2	1	0
			T2CN7	T2CN6	T2CN5	T2CN4	T2CN3	T2CN2	T2CN1	T2CN0
		Initial value	1	1	1	1	1	1	1	1
		Read/Wnte	W	W	W	W	W	W	W	W
Timer control/status register 2: TCSR2	59	Bit	7	6	5	4	3	2	1	0
			CMF2	ECM2I	—	—	T2OS1	T2OS0	CK2S1	CK2S0
		Initial value	0	0	1	1	0	0	0	0
		Read/Write	R/W	R/W	—	—	R/W	R/W	R/W	R/W
Timer 3 up-counter H: T3CNTH	5A	Bit	7	6	5	4	3	2	1	0
			T3C15	T3C14	T3C13	T3C12	T3C11	T3C10	T3C9	T3C8
		Initial value	0	0	0	0	0	0	0	0
		Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Timer 3 up-counter L: T3CNTL	5B	Bit	7	6	5	4	3	2	1	0
			T3C7	T3C6	T3C5	T3C4	T3C3	T3C2	T3C1	T3C0
		Initial value	0	0	0	0	0	0	0	0
		Read/Wnte	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Timer 3 time constant register H: T3CONRH	5C	Bit	7	6	5	4	3	2	1	0
			T3CN15	T3CN14	T3CN13	T3CN12	T3CN11	T3CN10	T3CN9	T3CN8
		Initial value	1	1	1	1	1	1	1	1
		Read/Wnte	W	W	W	W	W	W	W	W
Timer 3 time constant register L: T3CONRL	5D	Bit	7	6	5	4	3	2	1	0
			T3CN7	T3CN6	T3CN5	T3CN4	T3CN3	T3CN2	T3CN1	T3CN0
		Initial value	1	1	1	1	1	1	1	1
		Read/Write	W	W	W	W	W	W	W	W
Timer control/status register 3: TCSR3	5E	Bit	7	6	5	4	3	2	1	0
			CMF3	ECM3I	—	—	T3OS1	T3OS0	—	—
		Initial value	0	0	1	1	0	0	1	1
		Read/Write	R/W	R/W	—	—	R/W	R/W	—	—

Register	Address	Remarks								
Timer 4 up-counter H: T4CNTH	5F	Bit	7	6	5	4	3	2	1	0
			T4C15	T4C14	T4C13	T4C12	T4C11	T4C10	T4C9	T4C8
		Initial value	0	0	0	0	0	0	0	0
		Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Timer 4 up-counter L: T4CNTL	60	Bit	7	6	5	4	3	2	1	0
			T4C7	T4C6	T4C5	T4C4	T4C3	T4C2	T4C1	T4C0
		Initial value	0	0	0	0	0	0	0	0
		Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Timer 4 time constant register H: T4CONRH	61	Bit	7	6	5	4	3	2	1	0
			T4CN15	T4CN14	T4CN13	T4CN12	T4CN11	T4CN10	T4CN9	T4CN8
		Initial value	1	1	1	1	1	1	1	1
		Read/Write	W	W	W	W	W	W	W	W
Timer 4 time constant register L: T4CONRL	62	Bit	7	6	5	4	3	2	1	0
			T4CN7	T4CN6	T4CN5	T4CN4	T4CN3	T4CN2	T4CN1	T4CN0
		Initial value	1	1	1	1	1	1	1	1
		Read/Write	W	W	W	W	W	W	W	W
Timer control/status register 4: TCSR4	63	Bit	7	6	5	4	3	2	1	0
			CMF4	ECM4I	—	—	T4OS1	T4OS0	CK4S1	CK4S0
		Initial value	0	0	1	1	0	0	0	0
		Read/Write	R/W	R/W	—	—	R/W	R/W	R/W	R/W
Output data register 0: ODR0	64	Bit	7	6	5	4	3	2	1	0
			$ODR0_7$	$ODR0_6$	$ODR0_5$	$ODR0_4$	$ODR0_3$	$ODR0_2$	$ODR0_1$	$ODR0_0$
		Initial value	0	0	0	0	0	0	0	0
		Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
		Note Reading obtains input port values								
Output data register 1: ODR1	65	Bit	7	6	5	4	3	2	1	0
			$ODR1_7$	$ODR1_6$	$ODR1_5$	$ODR1_4$	$ODR1_3$	$ODR1_2$	$ODR1_1$	$ODR1_0$
		Initial value	0	0	0	0	0	0	0	0
		Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
		Note Reading obtains input port values								
Output data register 2: ODR2	66	Bit	7	6	5	4	3	2	1	0
			—	$ODR2_6$	$ODR2_5$	$ODR2_4$	$ODR2_3$	$ODR2_2$	$ODR2_1$	$ODR2_0$
		Initial value	1	0	0	0	0	0	0	0
		Read/Write	—	R/W	R/W	R/W	R/W	R/W	R/W	R/W
		Note Reading obtains input port values								

Register	Address	Remarks								
Output data register 3: ODR3	67	Bit	7	6	5	4	3	2	1	0
			ODR3$_7$	ODR3$_6$	ODR3$_5$	ODR3$_4$	ODR3$_3$	ODR3$_2$	ODR3$_1$	ODR3$_0$
		Initial value	0	0	0	0	0	0	0	0
		Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
		Note Reading obtains input port values								
Port 4: PORT4	68									
Port 0 data direction register: DDR0	69	Bit	7	6	5	4	3	2	1	0
			DDR0$_7$	DDR0$_6$	DDR0$_5$	DDR0$_4$	DDR0$_3$	DDR0$_2$	DDR0$_1$	DDR0$_0$
		Initial value	0	0	0	0	0	0	0	0
		Read/Write	W	W	W	W	W	W	W	W
Port 1 data direction register: DDR1	6A	Bit	7	6	5	4	3	2	1	0
			DDR1$_7$	DDR1$_6$	DDR1$_5$	DDR1$_4$	DDR1$_3$	DDR1$_2$	DDR1$_1$	DDR1$_0$
		Initial value	0	0	0	0	0	0	0	0
		Read/Write	W	W	W	W	W	W	W	W
Port 2 data direction register: DDR2	6B	Bit	7	6	5	4	3	2	1	0
			—	—	DDR2$_5$	DDR2$_4$	DDR2$_3$	DDR2$_2$	DDR2$_1$	DDR2$_0$
		Initial value	—	—	0	0	0	0	0	0
		Read/Write	—	—	W	W	W	W	W	W
Port 3 data direction register: DDR3	6C	Bit	7	6	5	4	3	2	1	0
			DDR3$_7$	DDR3$_6$	DDR3$_5$	DDR3$_4$	DDR3$_3$	DDR3$_2$	DDR3$_1$	DDR3$_0$
		Initial value	0	0	0	0	0	0	0	0
		Read/Write	W	W	W	W	W	W	W	W
I/O port control register 1: IOPCR1	6D	Bit	7	6	5	4	3	2	1	0
			DREQ$_0$E	TEND$_0$E	P2$_6$E	BUSE	AOUTE	TOUT$_1$E	SCK$_1$E	SCK$_0$E
		Initial value	0	0	0	0	0	0	0	0
		Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
I/O port control register 2: IOPCR2	6E	Bit	7	6	5	4	3	2	1	0
			—	P3$_7$C1	P3$_7$C0	INT$_2$E	INT$_0$E	P3$_3$C1	P3$_3$C0	TEND$_1$E
		Initial value	1	0	0	0	0	0	0	0
		Read/Write	—	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Register	Address	Remarks								
EEPROM control register 1: EEC1	70	Bit	7	6	5	4	3	2	1	0
			BUSY	BYTE/ PAGE	PERM	—	—	—	—	—
		Initial value	0	0	0	1	1	1	1	1
		Read/Write	R	R/W	R/W	—	—	—	—	—
EEPROM control register 2: EEC2	71	Bit	7	6	5	4	3	2	1	0
			PW	—	—	—	—	—	—	—
		Initial value	0	1	1	1	1	1	1	1
		Read/Write	W	—	—	—	—	—	—	—
Memory relocate register: MRR	72	Bit	7	6	5	4	3	2	1	0
			RMR3	RMR2	RMR1	RMR0	EPR3	EPR2	EPR1	EPR0
		Initial value	0	0	0	0	0	0	0	0
		Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
System control register: SYSCR	7F	Bit	7	6	5	4	3	2	1	0
			RAME	—	—	—	STBYE	CKC2	CKC1	CKC0
		Initial value	0	1	1	1	0	0	0	0
		Read/Write	R/W	—	—	—	R/W	R/W	R/W	R/W

Section Four

HD68000 16-Bit Microprocessor Family

HD68000/HD68HC000

MPU (Micro Processing Unit)

– **HD68000** –

The HD68000 is the first in a family of advanced microprocessors from Hitachi. Utilizing VLSI technology, the HD68000 is a fully-implemented 16-bit microprocessor with 32-bit registers, a rich basic instruction set, and versatile addressing modes.

The HD68000 possesses an asynchronous bus structure with a 24-bit address bus and a 16-bit data bus.

FEATURES

- 32-Bit Data and Address Registers
- 16 Megabyte Direct Addressing Range
- 56 Powerful Instruction Types
- Operations of Five Main Data Types
- Memory Mapped I/O
- 14 Addressing Modes

– **HD68HC000** –

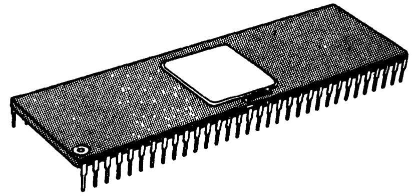
The HD68HC000 is a 16-bit microprocessor of HD68000 family, which is exactly compatible with the conventional HD68000.

The HD68HC000 is a complete CMOS device and the power dissipation is extremely low.

FEATURES

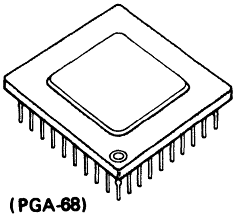
- Instruction Compatible with NMOS HD68000
- Pin Compatible with NMOS HD68000
- AC Timing Compatible with NMOS HD68000
- Low Power Dissipation (I_{CC} typ = 20 mA, I_{CC} max =35 mA at f = 12.5 MHz)

HD68000-8, HD68000-10, HD68000-12
HD68HC000-8, HD68HC000-10, HD68HC000-12



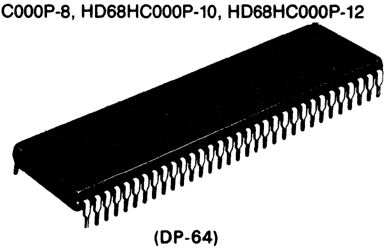
(DC-64)

HD68000Y-8, HD68000Y-10, HD68000Y-12
HD68HC000Y-8, HD68HC000Y-10, HD68HC000Y-12



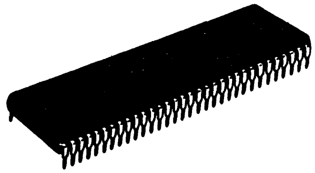
(PGA-68)

HD68000P-8
HD68HC000P-8, HD68HC000P-10, HD68HC000P-12



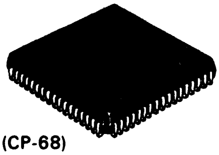
(DP-64)

HD68000PS-8
HD68HC000PS-8, HD68HC000PS-10, HD68HC000PS-12



(DP-64S)

HD68000CP-8
HD68HC000CP-8, HD68HC000CP-10, HD68HC000CP-12



(CP-68)

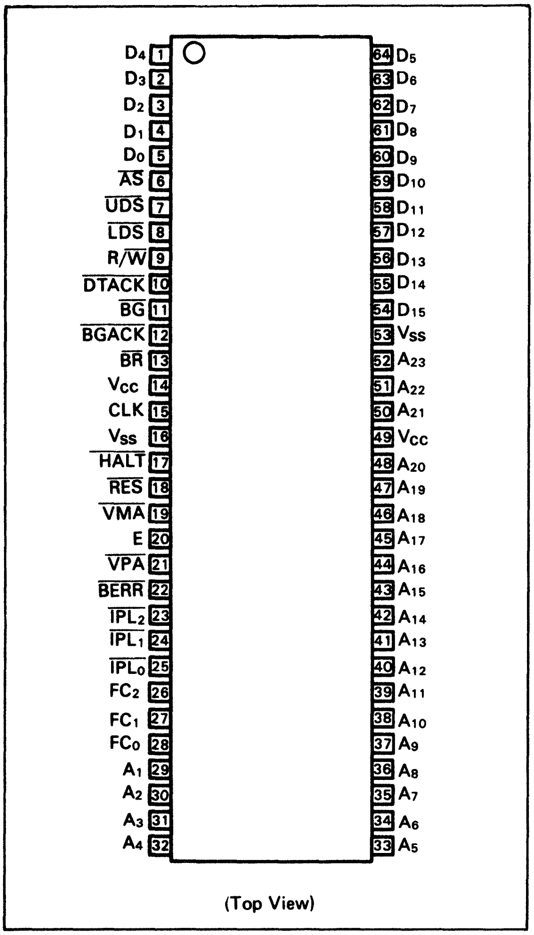
■ TYPE OF PRODUCTS

Type No.	Process	Clock Frequency (MHz)	Package
HD68000-8	NMOS	8.0	DC-64
HD68000-10		10.0	
HD68000-12		12.5	
HD68000Y-8		8.0	PGA-68
HD68000Y-10		10.0	
HD68000Y-12		12.5	
HD68000P-8		8.0	DP-64
HD68000PS-8		8.0	DP-64S
HD68000CP-8		8.0	CP-68
HD68HC000-8	CMOS	8.0	DC-64
HD68HC000-10		10.0	
HD68HC000-12		12.5	
HD68HC000Y-8		8.0	PGA-68
HD68HC000Y-10		10.0	
HD68HC000Y-12		12.5	
HD68HC000P-8		8.0	DP-64
HD68HC000P-10		10.0	
HD68HC000P-12		12.5	
HD68HC000PS-8		8.0	DP-64S
HD68HC000PS-10		10.0	
HD68HC000PS-12		12.5	
HD68HC000CP-8		8.0	CP-68
HD68HC000CP-10		10.0	
HD68HC000CP-12		12.5	

(Note) HD68000 refers to the NMOS version 68000, and HD68HC000 refers to the CMOS version 68000. 68000 stands for NMOS and CMOS version.

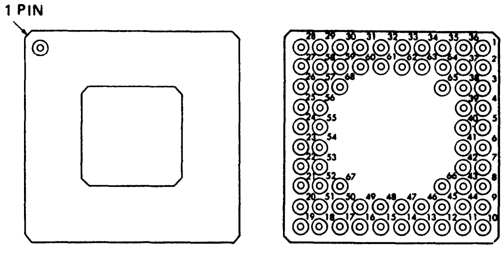
■ PIN ARRANGEMENT

● DC-64, DP-64, DP-64S



(Top View)

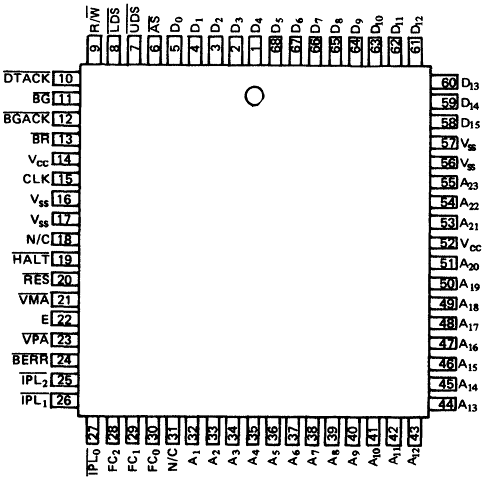
● PGA-68



(Top View) (Bottom View)

Pin No	Function	Pin No	Function	Pin No	Function	Pin No	Function
1	N/C	18	A_9	35	D_1	52	A_{12}
2	$\overline{DTACK}$	19	N/C	36	$\overline{AS}$	53	A_{15}
3	$\overline{BGACK}$	20	A_{14}	37	$\overline{LDS}$	54	A_{18}
4	$\overline{BR}$	21	A_{16}	38	$\overline{BG}$	55	V_{CC}
5	CLK	22	A_{17}	39	V_{CC}	56	V_{SS}
6	$\overline{HALT}$	23	A_{19}	40	V_{SS}	57	A_{23}
7	$\overline{VMA}$	24	A_{20}	41	$\overline{RES}$	58	D_{14}
8	E	25	A_{21}	42	$\overline{VPA}$	59	D_{11}
9	$\overline{BERR}$	26	A_{22}	43	$\overline{IPL_2}$	60	D_9
10	N/C	27	D_{15}	44	$\overline{IPL_0}$	61	D_6
11	FC_2	28	D_{12}	45	FC_1	62	D_3
12	FC_0	29	D_{10}	46	N/C	63	D_0
13	A_1	30	D_8	47	A_2	64	$\overline{UDS}$
14	A_3	31	D_7	48	A_5	65	$R/\overline{W}$
15	A_4	32	D_5	49	A_8	66	$\overline{IPL_1}$
16	A_6	33	D_4	50	A_{10}	67	A_{13}
17	A_7	34	D_2	51	A_{11}	68	D_{13}

● CP-68



(Top View)

■ ABSOLUTE MAXIMUM RATINGS

Item	Symbol	HD68000	HD68HC000	Unit
		Value	Value	
Supply Voltage	V_{CC}*	–0.3 ~ +7.0	–0.3 ~ +6.5	V
Input Voltage	V_{in}*	–0.3 ~ +7.0	–0.3 ~ +6.5	V
Operating Temperature Range	T_{opr}	0 ~ +70	0 ~ +70	°C
Storage Temperature	T_{stg}	–55 ~ +150	–55 ~ +150	°C

*With respect to V_{SS} (SYSTEM GND)

(NOTE) Permanent LSI damage may occur if maximum ratings are exceeded. Normal operation should be under recommended operating conditions. If these conditions are exceeded, it could affect reliability of LSI.

Since the HD68HC000 is a C-MOS device, users are expected to be cautious on "latch-up" problem caused by voltage fracturations.

■ RECOMMENDED OPERATING CONDITIONS

Item		Symbol	HD68000			HD68HC000			Unit
			min	typ	max	min	typ	max	
Supply Voltage		V_{CC}*	4.75	5.0	5.25	4.75	5.0	5.25	V
Input Voltage	CLK	V_{IH}*	2.0	–	V_{CC}	2.8	–	V_{CC}	V
	Other Inputs					2.0	–	V_{CC}	
	All Inputs	V_{IL}*	–0.3	–	0.8	–0.3	–	0.8	V
Operating Temperature		T_{opr}	0	25	70	0	25	70	°C

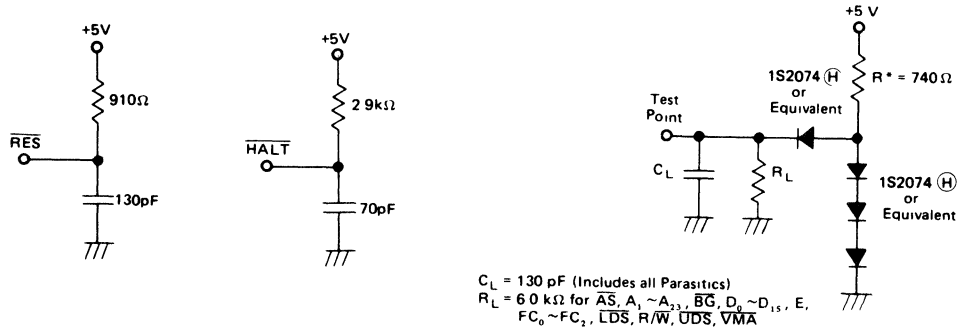
* With respect to V_{SS} (SYSTEM GND)

■ ELECTRICAL CHARACTERISTICS

● DC CHARACTERISTICS (V_{CC} = 5V ± 5%, V_{SS} = 0V, Ta = 0 ~ +70°C, Fig. 1, unless otherwise noted.)

Item		Symbol	Test Condition	HD68000		HD68HC000		Unit
				min	max	min	max	
Input "High" Voltage	CLK	V_{IH}		2.0	V_{CC}	2.8	V_{CC}	V
	Other Inputs					2.0	V_{CC}	
Input "Low" Voltage		V_{IL}		V_{SS}-0.3	0.8	V_{SS} -0.3	0.8	V
Input Leakage Current	$\overline{BERR}$, $\overline{BGACK}$, $\overline{BR}$, $\overline{DTACK}$, $\overline{IPL_0}$ ~ $\overline{IPL_2}$, $\overline{VPA}$, CLK	I_{in}	@ 5.25V	–	2.5	–	2.5	μA
	$\overline{HALT}$, $\overline{RES}$			–	20	–	20	
Three-State (Off State) Input Current	$\overline{AS}$, A_1~ A_{23}, D_0 ~ D_{15}, FC_0 ~ FC_2, $\overline{LDS}$, R/$\overline{W}$, $\overline{UDS}$, $\overline{VMA}$	I_{TSI}	@ 2.4V/0.4V	–	20	–	20	μA
Output "High" Voltage	$\overline{AS}$, A_1 ~ A_{23}, $\overline{BG}$, D_0 ~ D_{15}, FC_0 ~ FC_2, $\overline{LDS}$, R/$\overline{W}$, $\overline{UDS}$, $\overline{VMA}$, E	V_{OH}	I_{OH} = -400μA	2.4	–	V_{CC}-0.75	–	V
	E*			V_{CC}-0.75	–			
Output "Low" Voltage	$\overline{HALT}$	V_{OL}	I_{OL}=1.6 mA	–	0.5	–	0.5	V
	A_1~A_{23}, $\overline{BG}$, FC_0 ~ FC_2		I_{OL}=3.2 mA	–	0.5	–	0.5	
	$\overline{RES}$		I_{OL}=5.0 mA	–	0.5	–	0.5	
	$\overline{AS}$, D_0 ~ D_{15}, $\overline{LDS}$, R/$\overline{W}$, E, $\overline{UDS}$, $\overline{VMA}$		I_{OL}=5.3 mA	–	0.5	–	0.5	
Power Dissipation	CERAMIC PACKAGE	P_D	f = 6 MHz	–	1.5	–	–	W
			f = 8 MHz					
			f = 10 MHz					
			f = 12.5 MHz	–	1.75			
	PLASTIC PACKAGE		f = 8 MHz, V_{CC} = 5V, Ta = 25°C	–	0.9			
Current Dissipation		I_D**	f = 8 MHz	–	–	–	25	mA
			f = 10 MHz	–	–	–	30	
			f = 12.5 MHz	–	–	–	35	
Capacitance (Package Type Dependent)		C_{in}	V_{in} = 0V, Ta = 25°C, f = 1 MHz	–	20.0	–	20.0	pF

*With external pull up resistor of 1.1 kΩ.
**Without load.



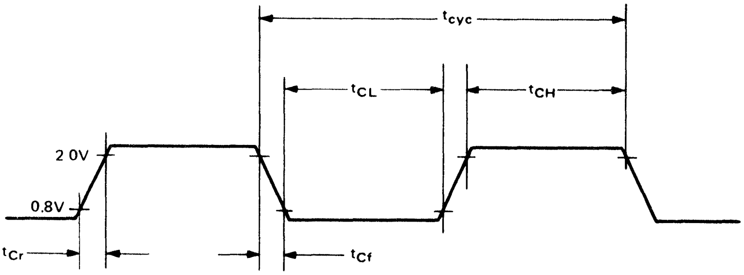
C_L = 130 pF (Includes all Parasitics)
R_L = 6 0 kΩ for $\overline{AS}$, A_1 ~A_{23}, $\overline{BG}$, D_0 ~D_{15}, E, FC_0 ~FC_2, $\overline{LDS}$, R/$\overline{W}$, $\overline{UDS}$, $\overline{VMA}$
*R = 1 22 kΩ for A_1 ~A_{23}, $\overline{BG}$, FC_0 ~FC_2

Figure 1 Test Loads

- **AC CHARACTERISTICS** **(V_{CC} = 5V ± 5%, V_{SS} = 0V, Ta = 0 ~ +70°C, unless otherwise noted.)**

CLOCK TIMING

Item	Symbol	Test Condition	8 MHz		10 MHz		12.5 MHz		Unit
			min	max	min	max	min	max	
Frequency of Operation	f	Fig. 2	4.0	8.0	4.0	10.0	4.0	12.5	MHz
Cycle Time	t_{cyc}		125	250	100	250	80	250	ns
Clock Pulse Width	t_{CL}		55	125	45	125	35	125	ns
	t_{CH}		55	125	45	125	35	125	ns
Rise and Fall Times	t_{Cr}		–	10	–	10	–	5	ns
	t_{Cf}		–	10	–	10	–	5	ns



(NOTE)

Timing measurements are referenced to and from a low voltage of 0.8 volt and high a voltage of 2.0 volts, unless otherwise noted.

The voltage swing through this range should start outside and pass through the range such that the rise or fall will be linear between 0.8 volt and 2.0 volts.

Figure 2 Clock Input Timing

READ AND WRITE CYCLES

Num.	Item	Symbol	Test Condition	8 MHz		10 MHz		12.5 MHz		Unit
				min	max	min	max	min	max	
1	Clock Period	t_{cyc}	Fig. 3, Fig. 4	125	250	100	250	80	250	ns
2	Clock Width Low	t_{CL}		55	125	45	125	35	125	ns
3	Clock Width High	t_{CH}		55	125	45	125	35	125	ns
4	Clock Fall Time	t_{Cf}		–	10	–	10	–	5	ns
5	Clock Rise Time	t_{Cr}		–	10	–	10	–	5	ns
6	Clock Low to Address Valid	t_{CLAV}		–	70	–	60	–	55*	ns
6A	Clock High to FC Valid	t_{CHFCV}		–	70	–	60	–	55	ns
7	Clock High to Address, Data Bus High Impedance (Maximum)	t_{CHADZ}		–	80	–	70	–	60	ns
8	Clock High to Address, FC Invalid (Minimum)	t_{CHAFI}		0	–	0	–	0	–	ns
9[1]	Clock High to AS, DS Low	t_{CHSL}		0	60	0	55	0	55	ns
11[2]	Address Valid to AS, DS Low (Read)/ AS Low (Write)	t_{AVSL}		30	–	20	–	0	–	ns
11A[2]	FC Valid to AS, DS Low (Read)/ AS Low (Write)	t_{FCVSL}		60	–	50	–	40	–	ns
12[1]	Clock Low to AS, DS High	t_{CLSH}		–	70	–	55	–	50	ns
13[2]	AS, DS High to Address/FC Invalid	t_{SHAFI}		30	–	20	–	10	–	ns
14[2]	AS, DS Width Low (Read)/AS Low (Write)	t_{SL}		240	–	195	–	160	–	ns
14A[2]	DS Width Low (Write)	t_{DSL}		115	–	95	–	80	–	ns
15[2]	AS, DS Width High	t_{SH}		150	–	105	–	65	–	ns
16	Clock High to Control Bus High Impedance	t_{CHCZ}		–	80	–	70	–	60	ns
17[2]	AS, DS High to R/W High (Read)	t_{SHRH}		40	–	20	–	10	–	ns
18[1]	Clock High to R/W High	t_{CHRH}		0	70	0	60	0	60	ns
20[1]	Clock High to R/W Low (Write)	t_{CHRL}		–	70	–	60	–	60	ns
20A[6]	AS Low to R/W Valid (Write)	t_{ASRV}		–	20	–	20	–	20	ns
21[2]	Address Valid to R/W Low (Write)	t_{AVRL}		20	–	0	–	0	–	ns
21A[2]	FC Valid to R/W Low (Write)	t_{FCVRL}		60	–	50	–	30	–	ns
22[2]	R/W Low to DS Low (Write)	t_{RLSL}		80	–	50	–	30	–	ns
23	Clock Low to Data Out Valid (Write)	t_{CLDO}		–	70	–	55	–	55	ns
25[2]	AS, DS High to Data Out Invalid (Write)	t_{SHDOI}		30	–	20	–	15	–	ns
26[2]	Data Out Valid to DS Low (Write)	t_{DOSL}		30	–	20	–	15	–	ns
27[5]	Data In to Clock Low (Setup Time on Read)	t_{DICL}		15	–	10	–	10	–	ns
28[2]	AS, DS High to DTACK High	t_{SHDAH}		0	245	0	190	0	150	ns
29	AS, DS High to Data In Invalid (Hold Time on Read)	t_{SHDII}		0	–	0	–	0	–	ns
30	AS, DS High to BERR High	t_{SHBEH}		0	–	0	–	0	–	ns
31[2, 5]	DTACK Low to Data In (Setup Time)	t_{DALDI}		–	90	–	65	–	50	ns
32	HALT and RESET Input Transition Time	$t_{RHr, f}$		0	200	0	200	0	200	ns
33	Clock High to BG Low	t_{CHGL}		–	70	–	60	–	50	ns
34	Clock High to BG High	t_{CHGH}		–	70	–	60	–	50	ns
35	BR Low to BG Low	t_{BRLGL}		1.5	90 ns +3.5	1.5	80 ns +3.5	1.5	70 ns +3.5	Clk. Per.

* 57 for HD68HC000

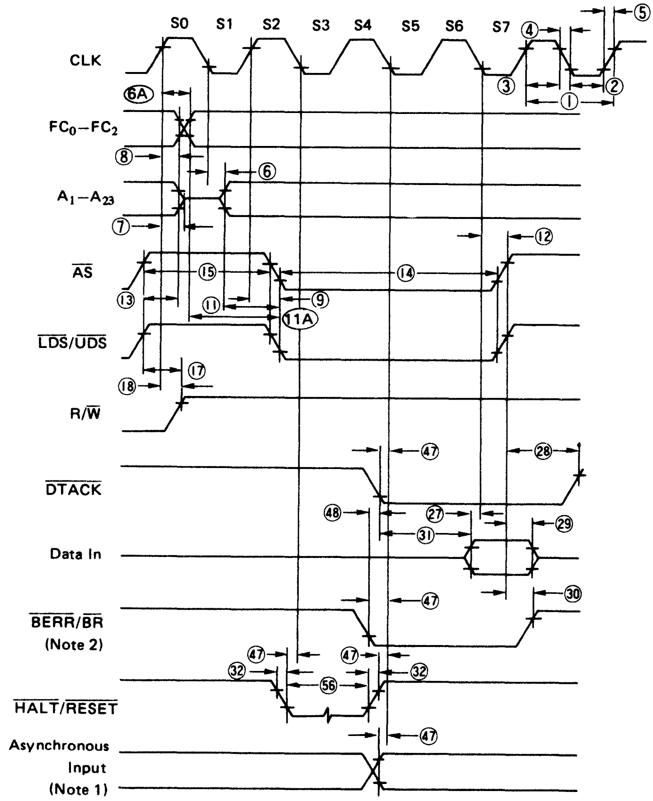
READ AND WRITE CYCLES (CONTINUED)

Num.	Item	Symbol	Test Condition	8 MHz		10 MHz		12.5 MHz		Unit
				min	max	min	max	min	max	
36[7]	$\overline{\text{BR}}$ High to $\overline{\text{BG}}$ High	t_{BRHGH}	Fig. 3, Fig. 4	1.5	90 ns +3.5	1.5	80 ns +3.5	1.5	70 ns +3.5	Clk.Per.
37	$\overline{\text{BGACK}}$ Low to $\overline{\text{BG}}$ High	t_{GALGH}		1.5	90 ns +3.5	1.5	80 ns +3.5	1.5	70 ns +3.5	Clk.Per.
37A[8]	$\overline{\text{BGACK}}$ Low to $\overline{\text{BR}}$ High	t_{GALBRH}		20	1.5 Clocks	20	1.5 Clocks	20	1.5 Clocks	ns
38	$\overline{\text{BG}}$ Low to Control, Address, Data Bus High Impedance ($\overline{\text{AS}}$ High)	t_{GLZ}		–	80	–	70	–	60	ns
39	$\overline{\text{BG}}$ Width High	t_{GH}		1.5	–	1.5	–	1.5	–	Clk.Per.
40	Clock Low to $\overline{\text{VMA}}$ Low	t_{CLVML}		–	70	–	70	–	70	ns
41	Clock Low to E Transition	t_{CLET}		–	70	–	55	–	45	ns
42	E Output Rise and Fall Time	$t_{Er,\ f}$		–	25	–	25	–	25	ns
43	$\overline{\text{VMA}}$ Low to E High	t_{VMLEH}		200	–	150	–	90	–	ns
44	$\overline{\text{AS}}$, $\overline{\text{DS}}$ High to $\overline{\text{VPA}}$ High	t_{SHVPH}		0	120	0	90	0	70	ns
45	E Low to Control, Address Bus Invalid (Address Hold Time)	t_{ELCAI}		30	–	10	–	10	–	ns
46	$\overline{\text{BGACK}}$ Width Low	t_{GAL}		1.5	–	1.5	–	1.5	–	Clk.Per.
47[5]	Asynchronous Input Setup Time	t_{ASI}		20	–	20	–	20	–	ns
48[3]	$\overline{\text{BERR}}$ Low to $\overline{\text{DTACK}}$ Low	t_{BELDAL}		20	–	20	–	20	–	ns
49[9]	$\overline{\text{AS}}$, $\overline{\text{DS}}$ High to E Low	t_{SHEL}		–70	70	–55	55	–45	45	ns
50	E Width High	t_{EH}		450	–	350	–	280	–	ns
51	E Width Low	t_{EL}		700	–	550	–	440	–	ns
53	Clock High to Data Out Invalid	t_{CHDOI}		0	–	0	–	0	–	ns
54	E Low to Data Out Invalid	t_{ELDOI}		30	–	20	–	15	–	ns
55	R/$\overline{\text{W}}$ to Data Bus Driven	t_{RLDBD}		30	–	20	–	10	–	ns
56[4]	$\overline{\text{HALT}}$/$\overline{\text{RESET}}$ Pulse Width	t_{HRPW}		10	–	10	–	10	–	Clk.Per.
57	$\overline{\text{BGACK}}$ High to Control Bus Driven	t_{GABD}		1.5	–	1.5	–	1.5	–	Clk.Per.
58[7]	$\overline{\text{BG}}$ High to Control Bus Driven	t_{GHBD}		1.5	–	1.5	–	1.5	–	Clk.Per.

NOTES:

1. For a loading capacitance of less than or equal to 50 picofarads, substract 5 nanoseconds from the value given in the maximum columns.
2. Actual value depends on clock period.
3. If #47 is satisfied for both $\overline{\text{DTACK}}$ and $\overline{\text{BERR}}$, #48 may be 0 nanoseconds.
4. For power up, the MPU must be held in $\overline{\text{RES}}$ state for 100 ms to allow stabilization of on-chip circuitry. After the system is powered up, #56 refers to the minimum pulse width required to reset the system.
5. If the asynchronous setup time (#47) requirements are satisfied, the $\overline{\text{DTACK}}$ low-to-data setup time (#31) requirement can be ignored. The data must only satisfy the date-in clock-low setup time (# 27) for the following cycle.
6. When $\overline{\text{AS}}$ and R/$\overline{\text{W}}$ are equally loaded (±20%), subtract 10 nanoseconds from the values given in these columns.
7. The processor will negate $\overline{\text{BG}}$ and begin driving the bus again if external arbitration logic negates $\overline{\text{BR}}$ before asserting $\overline{\text{BGACK}}$.
8. The minimum value must be met to guarantee proper operation. If the maximum value is exceeded, $\overline{\text{BG}}$ may be reasserted.
9. The falling edge of S6 triggers both the negation of the strobes ($\overline{\text{AS}}$ and $\text{x}\overline{\text{DS}}$) and the falling edge of E. Either of these events can occur first, depending upon the loading on each signal. Specification #49 indicates the absolute maximum skew that will occur between the rising edge of the strobes and the falling edge of the E clock.

These waveforms should only be referenced in regard to the edge-to-edge measurement of the timing specifications. They are not intended as a functional description of the input and output signals. Refer to other functional descriptions and their related diagrams for device operation.

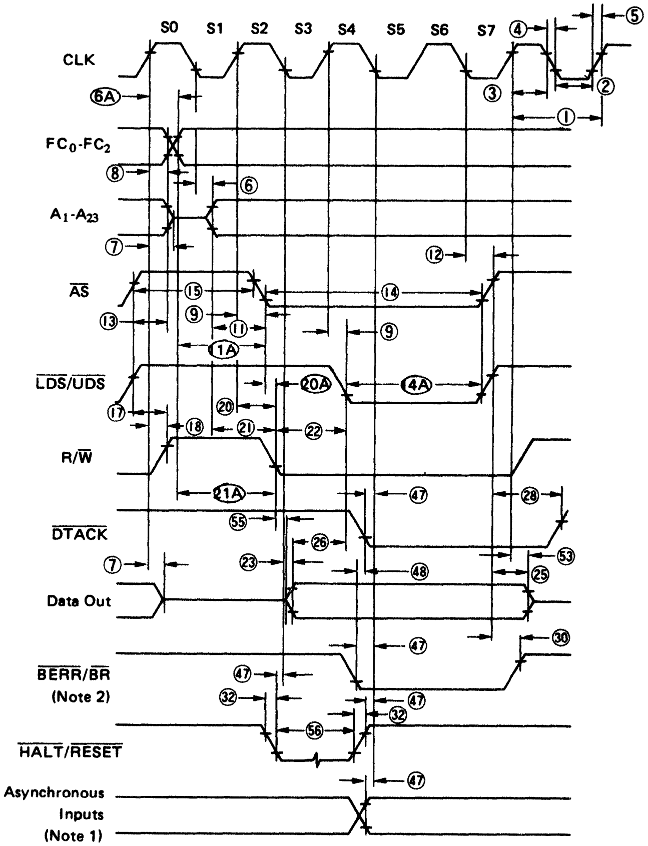


NOTES:

1. Setup time for the synchronous inputs $\overline{\text{BGACK}}$, $\overline{\text{IPL}_{0\text{-}2}}$ and $\overline{\text{VPA}}$ guarantees their recognition at the next falling edge of the clock.
2. $\overline{\text{BR}}$ need fall at this time only in order to insure being recognized at the end of this bus cycle.
3. Timing measurements are referenced to and from a low voltage of 0.8 volt and a high voltage 2.0 volts, unless otherwise noted. The voltage swing through this range should start outside and pass through the range such that the rise or fall will be linear between 0.8 volt and 2.0 volts.

Figure 3. Read Cycle Timing

These waveforms should only be referenced in regard to the edge-to-edge measurement of the timing specifications. They are not intended as a functional description of the input and output signals. Refer to other functional descriptions and their related diagrams for device operation.



NOTES:

1. Timing measurements are referenced to and from a low voltage of 0.8 volt and a high voltage of 2.0 volts, unless otherwise noted. The voltage swing through this range should start outside and pass through the range such that the rise or fall will be linear between 0.8 volt and 2.0 volts.
2. Because of loading variations, R/W may be valid after AS even though both are initiated by the rising edge of S2 (Specification 20A).

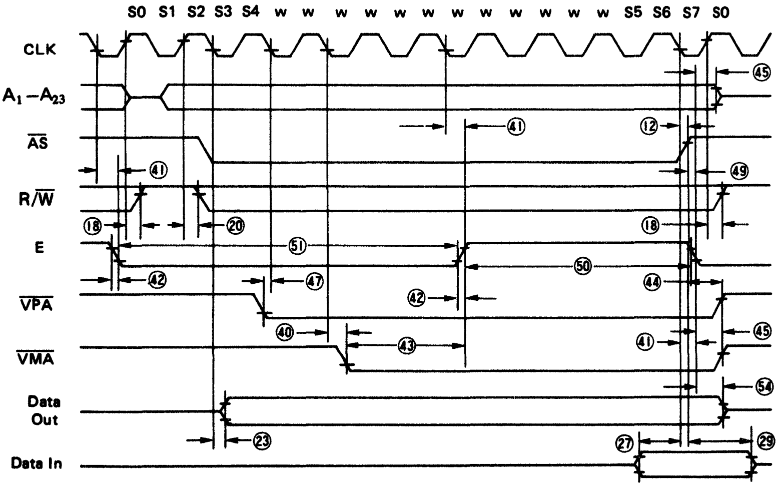
Figure 4. Write Cycle Timing

- **HMCS6800 TIMING**

Num.	Item	Symbol	Test Condition	6 MHz		8 MHz		10 MHz		12.5 MHz		Unit
				min	max	min	max	min	max	min	max	
12	Clock Low to $\overline{AS}$, $\overline{DS}$ High	t_{CLSH}	Fig. 5, Fig. 6	–	80	–	70	–	55	–	50	ns
18	Clock High to R/$\overline{W}$ High	t_{CHRH}		0	80	0	70	0	60	0	60	ns
20	Clock High to R/$\overline{W}$ Low (Write)	t_{CHRL}		–	80	–	70	–	60	–	60	ns
23	Clock Low to Data Out Valid (Write)	t_{CLDO}		–	80	–	70	–	55	–	55	ns
27	Data In to Clock Low (Setup Time on Read)	t_{DICL}		25	–	15	–	10	–	10	–	ns
29	$\overline{AS}$, $\overline{DS}$ High to Data In Invalid (Hold Time on Read)	t_{SHDII}		0	–	0	–	0	–	0	–	ns
40	Clock Low to $\overline{VMA}$ Low	t_{CLVML}		–	80	–	70	–	70	–	70	ns
41	Clock Low to E Transition	t_{CLET}		–	35	–	70	–	55	–	45	ns
42	E Output Rise and Fall Time	$t_{Er,f}$		–	25	–	25	–	25	–	25	ns
43	$\overline{VMA}$ Low to E High	t_{VMLEH}		240	–	200	–	150	–	90	–	ns
44	$\overline{AS}$, $\overline{DS}$ High to $\overline{VPA}$ High	t_{SHVPH}		0	160	0	120	0	90	0	70	ns
45	E Low to Control, Address Bus Invalid (Address Hold Time)	t_{ELCAI}		35	–	30	–	10	–	10	–	ns
47	Asynchronous Input Setup Time	t_{ASI}		25	–	20	–	20	–	20	–	ns
49[1]	$\overline{AS}$, $\overline{DS}$ High to E Low	t_{SHEL}		–80	–	–70	70	–55	55	–45	45	ns
50	E Width High	t_{EH}		600	–	450	–	350	–	280	–	ns
51	E Width Low	t_{EL}		900	–	700	–	550	–	440	–	ns
54	E Low to Data Out Invalid	t_{ELDOI}		40	–	30	–	20	–	15	–	ns

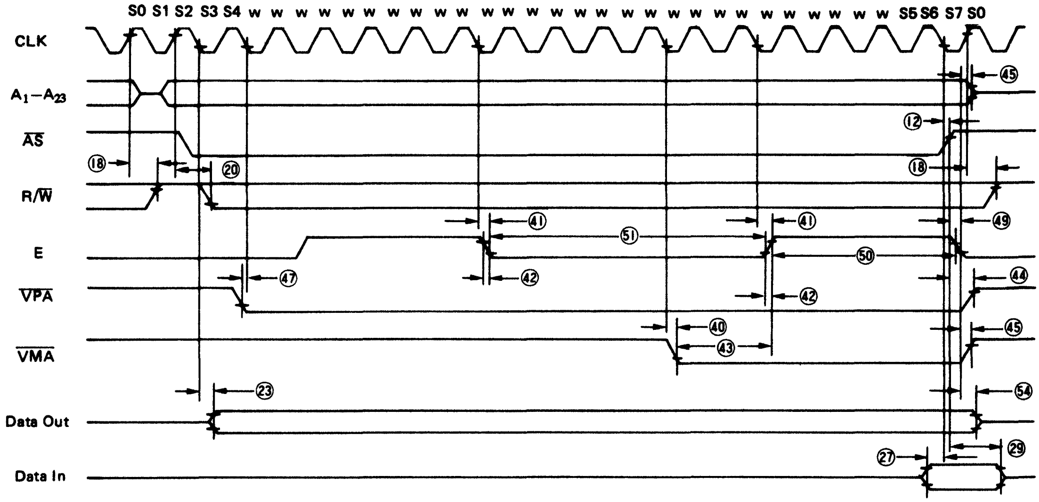
NOTE:

1. The falling edge of S6 triggers both the negation of the strobes ($\overline{AS}$ and x$\overline{DS}$) and the falling edge of E. Either of these events can occur first, depending upon the loading on each signal. Specification #49 indicates the absolute maximum skew that will occur between the rising edge of the strobes and the falling edge of the E clock.



NOTE: This timing diagram is included for those who wish to design their own circuit to generate $\overline{VMA}$. It shows the best case possibly attainable.

Figure 5. HD6800 Timing—Best Case



NOTE: This timing diagram is included for those who wish to design their own circuit to generate $\overline{\text{VMA}}$. It shows the worst case possibly attainable.

Figure 6. HD6800 Timing—Worst Case

BUS ARBITRATION

Num.	Item	Symbol	Test Condition	8 MHz		10 MHz		12.5 MHz		Unit
				min	max	min	max	min	max	
7	Clock High to Address, Data Bus High Impedance	t_{CHADZ}	Fig. 7 ~ Fig. 9	–	80	–	70	–	60	ns
16	Clock High to Control Bus High Impedance	t_{CHCZ}		–	80	–	70	–	60	ns
33	Clock High to $\overline{\text{BG}}$ Low	t_{CHGL}		–	70	–	60	–	50	ns
34	Clock High to $\overline{\text{BG}}$ High	t_{CHGH}		–	70	–	60	–	50	ns
35	$\overline{\text{BR}}$ Low to $\overline{\text{BG}}$ Low	t_{BRLGL}		1.5	90ns +3.5	1.5	80ns +3.5	1.5	70ns +3.5	Clk.Per.
36[1]	$\overline{\text{BR}}$ High to $\overline{\text{BG}}$ High	t_{BRHGH}		1.5	90ns +3.5	1.5	80ns +3.5	1.5	70ns +3.5	Clk.Per.
37	$\overline{\text{BGACK}}$ Low to $\overline{\text{BG}}$ High	t_{GALGH}		1.5	90ns +3.5	1.5	80ns +3.5	1.5	70ns +3.5	Clk.Per.
37A[2]	$\overline{\text{BGACK}}$ Low to $\overline{\text{BR}}$ High	t_{GALBRH}		20	1.5 Clocks	20	1.5 Clocks	20	1.5 Clocks	ns
38	$\overline{\text{BG}}$ Low to Control, Address, Data Bus High Impedance ($\overline{\text{AS}}$ High)	t_{GLZ}		–	80	–	70	–	60	ns
39	$\overline{\text{BG}}$ Width High	t_{GH}		1.5	–	1.5	–	1.5	–	Clk.Per.
46	$\overline{\text{BGACK}}$ Width Low	t_{GAL}		1.5	–	1.5	–	1.5	–	Clk.Per.
47	Asynchronous Input Setup Time	t_{ASI}		20	–	20	–	20	–	ns
57	$\overline{\text{BGACK}}$ High to Control Bus Driven	t_{GABD}		1.5	–	1.5	–	1.5	–	Clk.Per.
58[1]	$\overline{\text{BG}}$ High to Control Bus Driven	t_{GHBD}		1.5	–	1.5	–	1.5	–	Clk.Per.

NOTES:

1. The processor will negate $\overline{\text{BG}}$ and begin driving the bus again if external arbitration logic negates $\overline{\text{BR}}$ before asserting $\overline{\text{BGACK}}$.
2. The minimum value must be met to guarantee proper operation. If the maximum value is exceeded, $\overline{\text{BG}}$ may be reasserted.

Figures 7, 8, and 9 depict the three bus arbitration cases that can arise. Figure 7 shows the timing where AS is negated when the processor asserts BG (Idle Bus Case). Figure 8 shows the timing where AS is asserted when the processor asserts BG (Active Bus Case). Figure 9 shows the timing where more than one bus master are requesting the bus. Refer to Bus Arbitration for a complete discussion of bus arbitration.

The waveforms shown in Figures 7, 8, and 9 should only be referenced in regard to the edge-to-edge measurement of the timing specifications. They are not intended as a functional description of the input and output signals. Refer to other functional descriptions and their related diagrams for device operation.

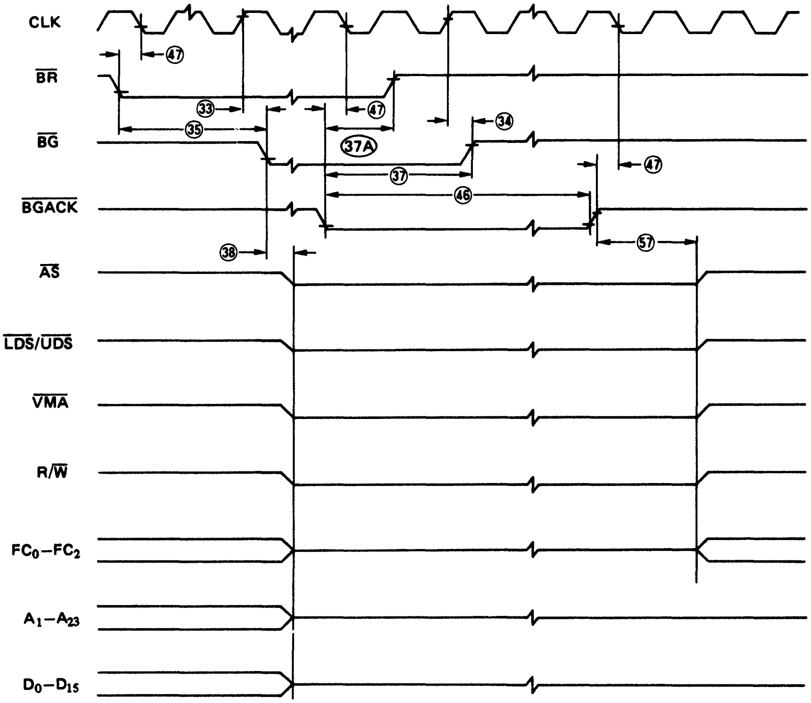


Figure 7. Bus Arbitration Timing Diagram — Idle Bus Case

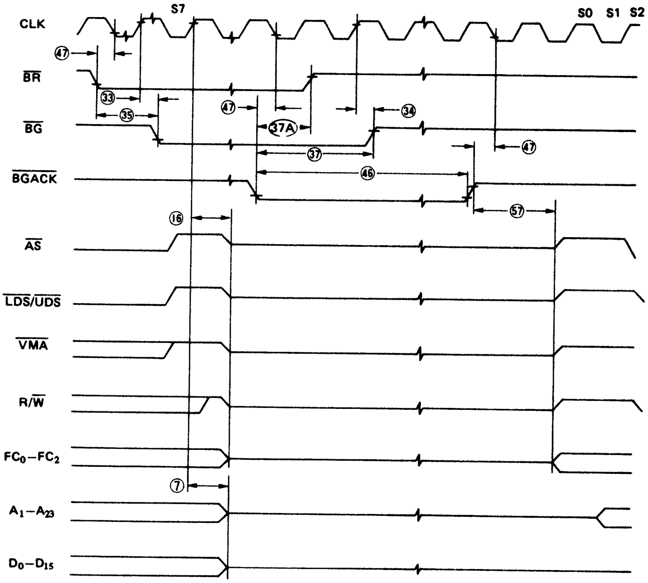


Figure 8. Bus Arbitration Timing Diagram – Active Bus Case

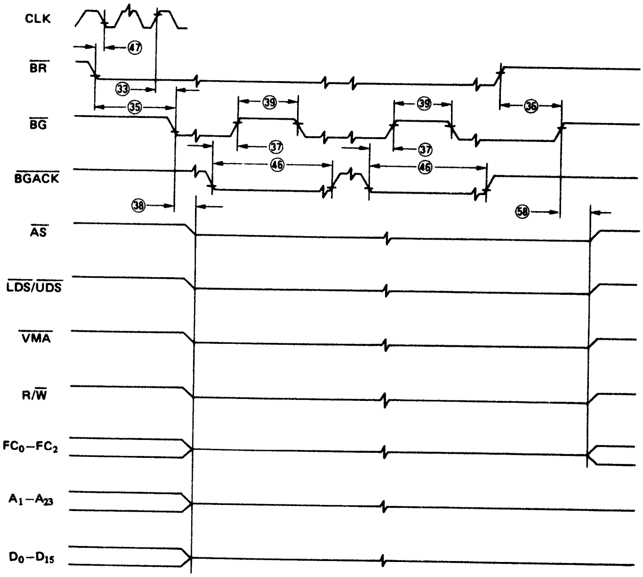


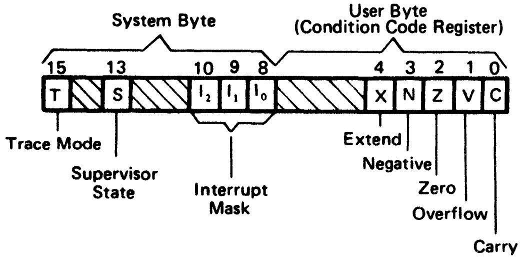
Figure 9. Bus Arbitration Timing Diagram – Multiple Bus Requests

■ INTRODUCTION

As shown in the programming model, the 68000 offers seventeen 32-bit registers in addition to the 32-bit program counter and a 16-bit status register. The first eight registers (D0 ~ D7) are used as data registers for byte (8-bit), word (16-bit), and long word (32-bit) data operations. The second set of seven registers (A0 ~ A6) and the system stack pointer may be used as software stack pointers and base address registers. In addition, these registers may be used for word and long word address operations. All 17 registers may be used as index registers.

The status register contains the interrupt mask (eight levels available) as well as the condition codes; extend (X), negative (N), zero (Z), overflow (V), and carry (C). Additional status bits indicate that the processor is in a trace (T) mode and/or in a supervisor (S) state.

Status Register



Unused, read as zero.

● DATA TYPES AND ADDRESSING MODES

Five basic data types are supported. These data types are:

(1) Bits
(2) BCD Digits (4 bits)
(3) Bytes (8 bits)
(4) Word (16 bits)
(5) Long Words (32 bits)

In addition, operations on other data types such as memory address, status word data, etc., are provided for in the instruction set.

The 14 addressing modes, shown in Table 1, includes six basic types:

(1) Register Direct
(2) Register Indirect
(3) Absolute
(4) Immediate
(5) Program Counter Relative
(6) Implied

Included in the register indirect addressing modes is the capability to do postincrementing, predecrementing, offsetting and indexing. Program counter relative mode can also be modified via indexing and offsetting.

Programming Model

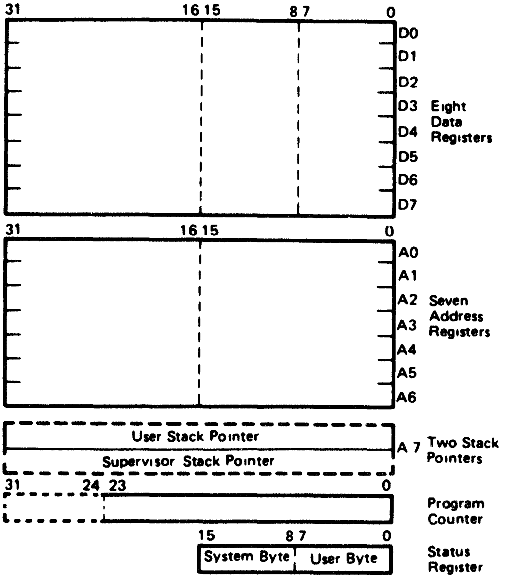


Table 1 Addressing Modes

Mode	Generation
Register Direct Addressing	
Data Register Diredt	EA = Dn
Address Register Direct	EA = An
Absolute Data Addressing	
Absolute Short	EA = (Next Word)
Absolute Long	EA = (Next Two Words)
Program Counter Relative Addressing	
Relative with Offset	EA = (PC) + d_{16}
Relative with Index and Offset	EA = (PC) + (Xn) + d_8
Register Indirect Addressing	
Register Indirect	EA = (An)
Postincrement Register Indirect	EA = (AN), An ← An + N
Predecrement Register Indirect	An ← An − N, EA = (An)
Register Indirect with Offset	EA = (An) + d_{16}
Indexed Register Indirect with Offset	EA = (An) + (Xn) + d_8
Immediate Data Addressing	
Immediate	DATA = Next Word(s)
Quick Immediate	Inherent Data
Implied Addressing	
Implied Register	EA = SR, USP, SP, PC

(NOTES)

EA = Effective Address
An = Address Register
Dn = Data Register
Xn = Address or Data Register used as Index Register
SR = Status Register
PC = Program Counter
( ) = Contents of
d_8 = Eight-bit Offset (displacement)
d_{16} = Sixteen-bit Offset (displacement)
N = 1 for Byte, 2 for Words and 4 for Long Words. If An is the stack pointer and the operand size is byte, N=2 to keep the stack pointer on a word boundary.
← = Replaces

• INSTRUCTION SET OVERVIEW

The 68000 instruction set is shown in Table 2. Some additional instructions are variations, or subsets, of these and they appear in Table 3. Special emphasis has been given to the instruction set's support of structured high-level languages to facilitate ease of programming. Each instruction, with few exceptions, operates on bytes, words, and long words and most instructions can use any of the 14 addressing modes. Combining instruction types, data types, and addressing modes, over 1000 useful instructions are provided. These instructions include signed and unsigned multiply and divide, "quick" arithmetic operations, BCD arithmetic and expanded operations (through traps).

Table 2 Instruction Set

Mnemonic	Description
ABCD	Add Decimal with Extend
ADD	Add
AND	Logical And
ASL	Arithmetic Shift Left
ASR	Arithmetic Shift Right
B$_{CC}$	Branch Conditionally
BCHG	Bit Test and Change
BCLR	Bit Test and Clear
BRA	Branch Always
BSET	Bit Test and Set
BSR	Branch to Subroutine
BTST	Bit Test
CHK	Check Register Against Bounds
CLR	Clear Operand
CMP	Compare
DB$_{CC}$	Test Condition, Decrement and Branch
DIVS	Signed Divide
DIVU	Unsigned Divide
EOR	Exclusive Or
EXG	Exchange Registers
EXT	Sign Extend
JMP	Jump
JSR	Jump to Subroutine
LEA	Load Effective Address
LINK	Link Stack
LSL	Logical Shift Left
LSR	Logical Shift Right
MOVE	Move
MOVEM	Move Multiple Registers
MOVEP	Move Peripheral Data
MULS	Signed Multiply
MULU	Unsigned Multiply
NBCD	Negate Decimal with Extend
NEG	Negate
NOP	No Operation
NOT	One's Complement
OR	Logical Or
PEA	Push Effective Address
RESET	Reset External Devices
ROL	Rotate Left without Extend
ROR	Rotate Right without Extend
ROXL	Rotate Left with Extend
ROXR	Rotate Right with Extend
RTE	Return from Exception
RTR	Return and Restore
RTS	Return from Subroutine
SBCD	Subtract Decimal with Extend
S$_{CC}$	Set Conditional
STOP	Stop
SUB	Subtract
SWAP	Swap Data Register Halves
TAS	Test and Set Operand
TRAP	Trap
TRAPV	Trap on Overflow
TST	Test
UNLK	Unlink

Table 3 Variations of Instruction Types

Instruction Type	Variation	Description
ADD	**ADD**	Add
	ADDA	Add Address
	ADDQ	Add Quick
	ADDI	Add Immediate
	ADDX	Add with Extend
AND	**AND**	Logical And
	ANDI	And Immediate
	ANDI to CCR	And Immediate to Condition Codes
	ANDI to SR	And Immediate to Status Register
CMP	**CMP**	Compare
	CMPA	Compare Address
	CMPM	Compare Memory
	CMPI	Compare Immediate
EOR	**EOR**	Exclusive Or
	EORI	Exclusive Or Immediate
	EORI to CCR	Exclusive Or Immediate to Condition Codes
	EORI to SR	Exclusive Or Immediate to Status Register
MOVE	**MOVE**	Move
	MOVEA	Move Address
	MOVEQ	Move Quick
	MOVE from SR	Move from Status Register
	MOVE to SR	Move to Status Register
	MOVE to CCR	Move to Condition Codes
	MOVE USP	Move User Stack Pointer
NEG	**NEG**	Negate
	NEGX	Negate with Extend
OR	**OR**	Logical Or
	ORI	Or Immediate
	ORI to CCR	Or Immediate to Condition Codes
	ORI to SR	Or Immediate to Status Register
SUB	**SUB**	Subtract
	SUBA	Subtract Address
	SUBI	Subtract Immediate
	SUBQ	Subtract Quick
	SUBX	Subtract with Extend

■ REGISTER DESCRIPTION AND DATA ORGANIZATION

The following paragraphs describe the registers and data organization of the 68000.

• OPERAND SIZE

Operand sizes are defined as follows: a byte equals 8 bits, a word equals 16 bits, and a long word equals 32 bits. The operand size for each instruction is either explicitly encoded in the instruction or implicitly defined by the instruction operation. Implict instructions support some subset of all three sizes.

• DATA ORGANIZATION IN REGISTERS

The eight data registers support data operands of 1, 8, 16, or 32 bits. The seven address registers together with the active stack pointer support address operands of 32 bits.

DATA REGISTERS

Each data register is 32 bits wide. Byte operands occupy the low order 8 bits, word operands the low order 16 bits, and long word operands the entire 32 bits. The least significant bit is addressed as bit zero, the most significant bit is addressed as bit 31.

When a data register is used as either a source or destination operand, only the appropriate low-order portion is changed; the remaining high-order portion is neither used nor changed.

ADDRESS REGISTERS

Each address register and the stack pointer is 32 bits wide and holds a full 32 bit address. Address registers do not support byte sized operands. Therefore, when an address register is used as a source operand, either the low order word or the entire long word operand is used depending upon the operation size. When an address register is used as the destination operand, the entire register is affected regardless of the operation size. If the operation size is word, any other operands are sign extended to 32 bits before the operation is performed.

• DATA ORGANIZATION IN MEMORY

Bytes are individually addressable with the high order byte having an even address the same as the word, as shown in Figure 10. The low order byte has an odd address that is one count higher than the word address. Instructions and multibyte data are accessed only on word (even byte) boundaries. If a long word datum is located at address n (n even), then the second word of that datum is located at address n + 2.

The data types supported by the 68000 are: bit data, integer data of 8, 16, or 32 bits, 32-bit addresses and binary coded decimal data. Each of these data types is put in memory, as shown in Figure 11. The numbers indicate the order in which the data would be accessed from the processor.

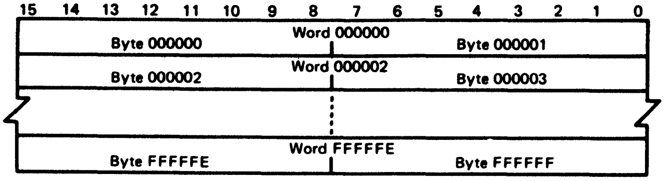


Figure 10 Word Organization in Memory

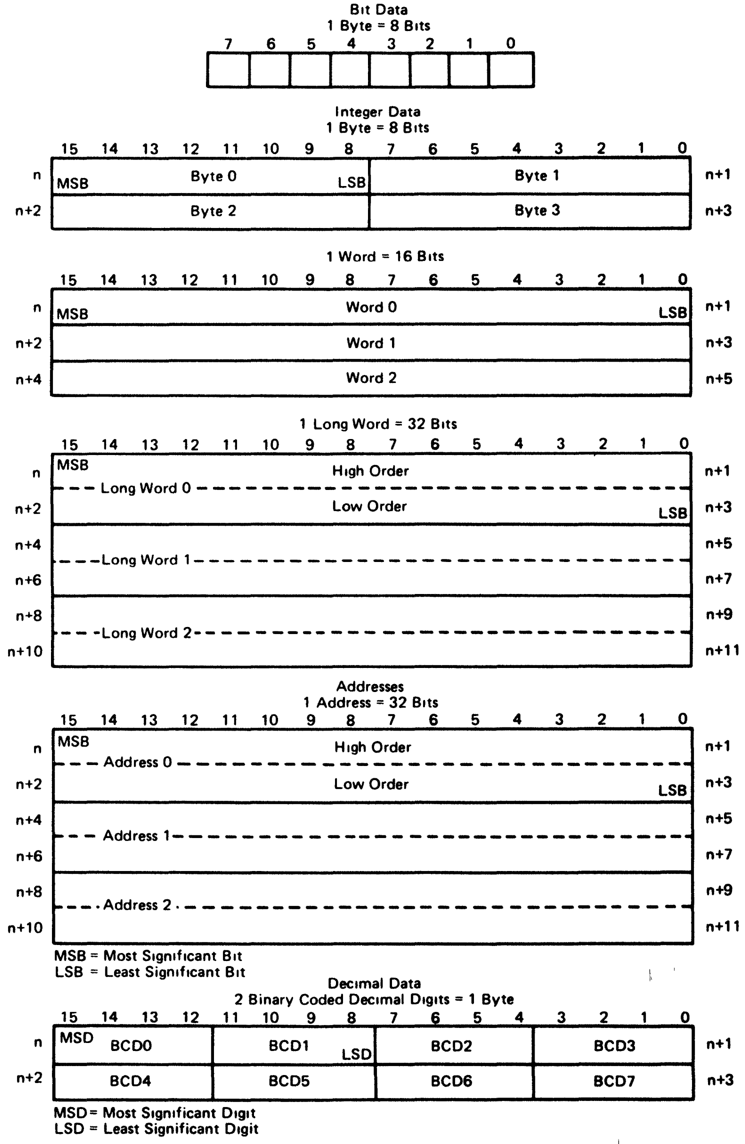


Figure 11 Data Organization in Memory

• ADDRESSING

Instructions for the 68000 contain two kinds of information: the type of function to be performed, and the location of the operand(s) on which to perform that function. The methods used to locate (address) the operand(s) are explained in the following paragraphs.

Instructions specify an operand location in one of three ways:

Register Specification – the number of the register is given in the register field of the instruction.

Effective Address – use of the different effective address modes.

Implicit Reference – the definition of certain instructions implies the use of specific registers.

• INSTRUCTION FORMAT

Instructions are from one to five words in length, as shown in Figure 12. The length of the instruction and the operation to be performed is specified by the first word of the instruction which is called the operation word. The remaining words further specify the operands. These words are either immediate operands or extensions to the effective address mode specified in the operation word.

• PROGRAM/DATA REFERENCES

The 68000 separates memory references into two classes: program references, and data references. Program references, as the name implies, are references to that section of memory that contains the program being executed. Data references refer to that section of memory that contains data. Operand reads are from the data space except in the case of the program counter relative addressing mode. All operand writes are to the data space.

• REGISTER SPECIFICATION

The register field within an instruction specifies the register to be used. Other fields within the instruction specify whether the register selected is an address or data register and how the register is to be used.

• EFFECTIVE ADDRESS

Most instructions specify the location of an operand by using the effective address field in the operation word. For example, Figure 13 shows the general format of the single effective address instruction operation word. The effective address is composed of two 3-bit fields: the mode field, and the register field. The value in the mode field selects the different address modes. The register field contains the number of a register.

The effective address field may require additional information to fully specify the operand. This additional information, called the effective address extension, is contained in the following word or words and is considered part of the instruction, as shown in Figure 12. The effective address modes are grouped into three categories: register direct, memory addressing, and special.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Operation Word (First Word Specifies Operation and Modes)															
Immediate Operand (If Any, One or Two Words)															
Source Effective Address Extension (If Any, One or Two Words)															
Destination Effective Address Extension (If Any, One or Two Words)															

Figure 12 Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	X	X	X	X	X	X	X	X	X	Effective Address Mode			Register		

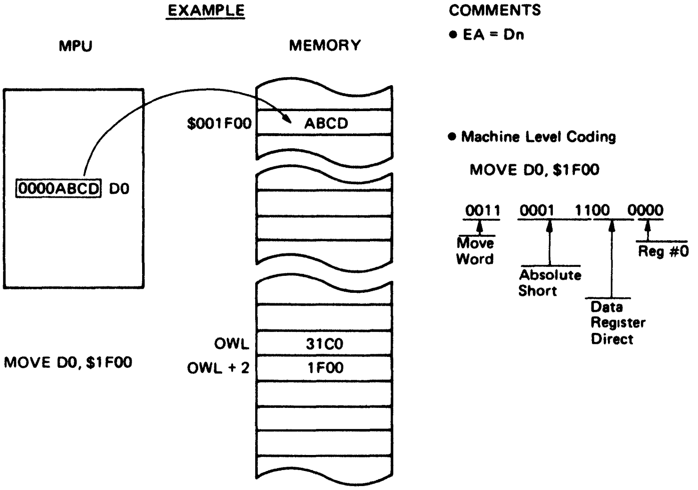
Figure 13 Single-Effective-Address Instruction Operation Word General Format

REGISTER DIRECT MODES

These effective addressing modes specify that the operand is in one of the 16 multifunction registers.

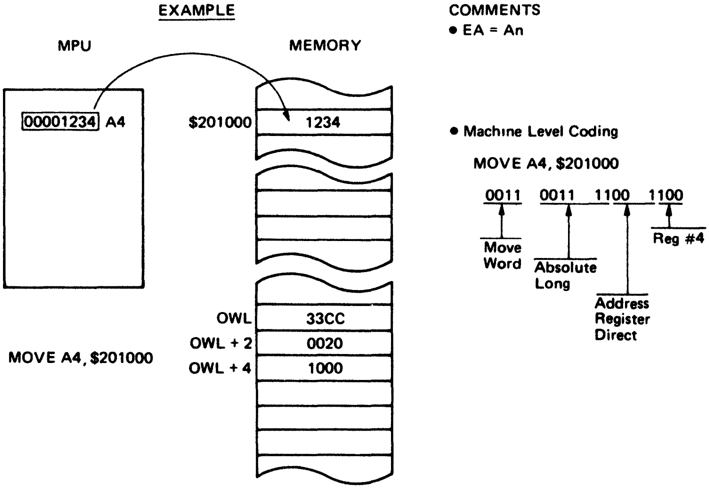
Data Register Direct

The operand is in the data register specified by the effective address register field.



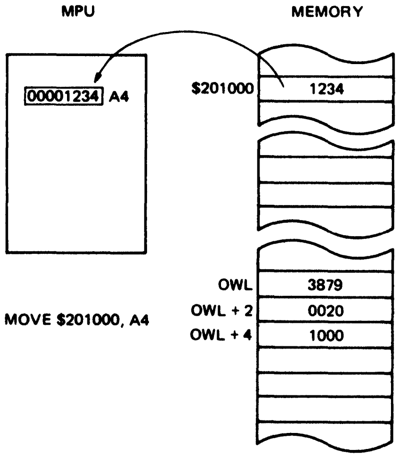
Address Register Direct

The operand is in the address register specified by the effective address register field.



HITACHI

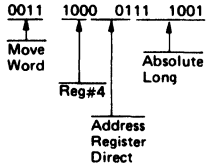
EXAMPLE



COMMENTS

- EA = An
- Address Register Sign Extended
- Machine Level Coding

MOVE $201000, A4



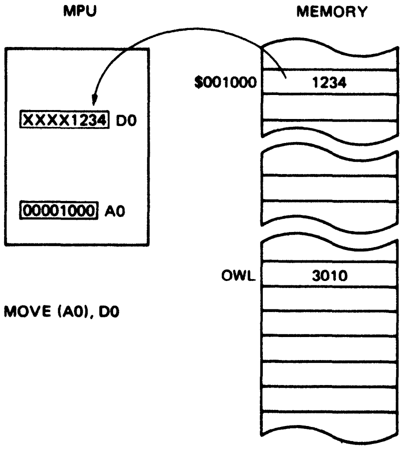
MEMORY ADDRESS MODES

These effective addressing modes specify that the operand is in memory and provide the specific address of the operand.

Address Register Indirect

The address of the operand is in the address register specified by the register field. The reference is classified as a data reference with the exception of the jump and jump to subroutine instructions.

EXAMPLE

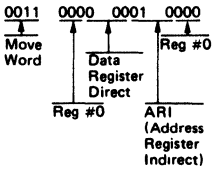


COMMENTS

- EA = (An)

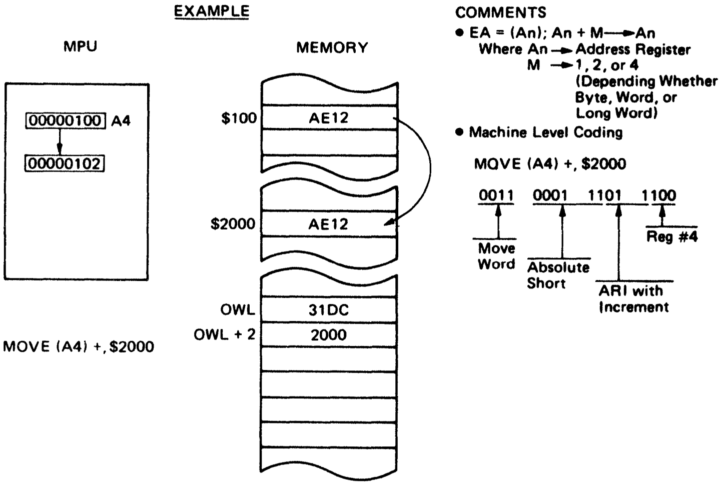
- Machine Level Coding

MOVE (A0), D0



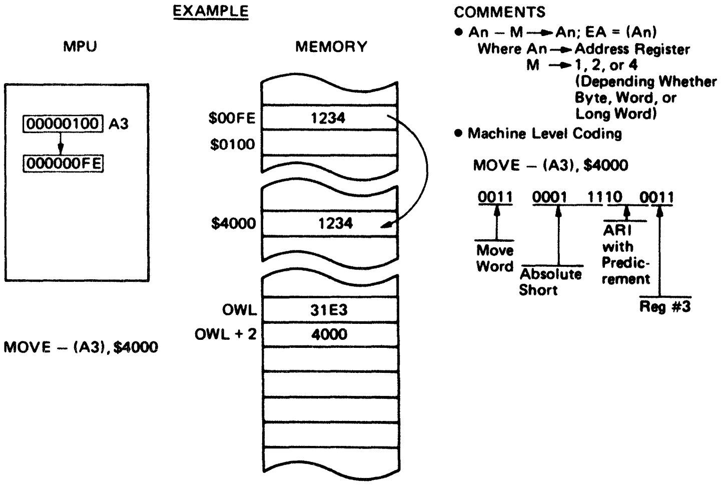
Address Register Indirect With Postincrement

The address of the operand is in the address register specified by the register field. After the operand address is used, it is incremented by one, two, or four depending upon whether the size of the operand is byte, word, or long word. If the address register is the stack pointer and the operand size is byte, the address is incremented by two rather than one to keep the stack pointer on a word boundary. The reference is classified as a data reference.



Address Register Indirect With Predecrement

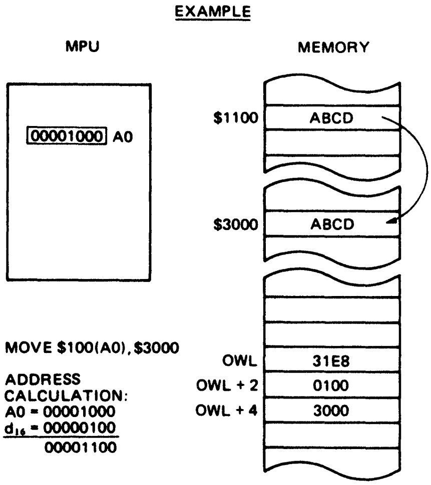
The address of the operand is in the address register specified by the register field. Before the operand address is used, it is decremented by one, two, or four depending upon whether the operand size is byte, word, or long word. If the address register is the stack pointer and the operand size is byte, the address is decremented by two rather than one to keep the stack pointer on a word boundary. The reference is classified as a data reference.



Address Register Indirect With Displacement

This address mode requires one word of extension. The address of the operand is the sum of the address in the address register and the sign-extended 16-bit displacement integer in the extension word. The reference is classified as a data reference with the exception of the jump to subroutine instructions.

EXAMPLE



COMMENTS

- EA = An + d_{16}
  Where An ⟶ Pointer Register
  d_{16} ⟶ 16-Bit Displacement
- d_{16} Displacement is Sign Extended
- Machine Level Coding

MOVE $100(A0),$3000

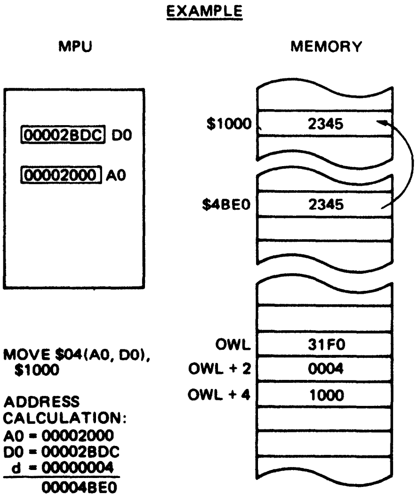
0011 0001 1110 1000

Move Word — Absolute Short — ARI with Displacement — Reg #0

Address Register Indirect With Index

This address mode requires one word of extension. The address of the operand is the sum of the address in the address register, the sign-extended displacement integer in the low order eight bits of the extension word, and the contents of the index register. The reference is classified as a data reference with the exception of the jump and jump to subroutine instructions.

EXAMPLE



COMMENTS

- EA = An + Rx + d_8
  Where
  An ⟶ Pointer Register
  Rx ⟶ Designated Index Register, (Either Address Register or Data Register)
  d_8 ⟶ 8-Bit Displacement
- Rx & d_8 are Sign Extended
- Rx may be Word or Long Word
  Long Word may be Designated with Rx.L
- Machine Level Coding

MOVE $04(A0, D0), $1000

0011 0001 1111 0000

Move Word — Absolute Short — ARI with Index — Reg #0

0000 0000 0000 0100

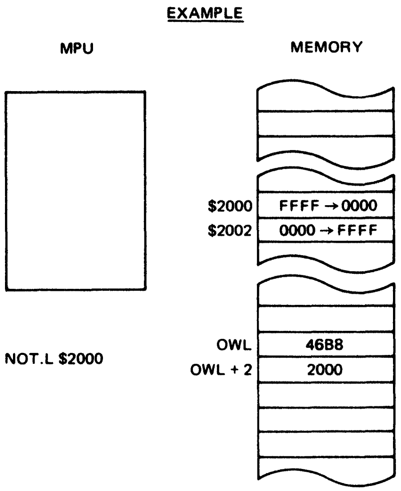
D/A — Reg #0 — Word — Constant Zeros — Offset

SPECIAL ADDRESS MODE

The special address modes use the effective address register field to specify the special addressing mode instead of a register number.

Absolute Short Address

This address mode requires one word of extension. The address of the operand is the extension word. The 16-bit address is sign extended before it is used. The reference is classified as a data reference with the exception of the jump and jump to subroutine instructions.



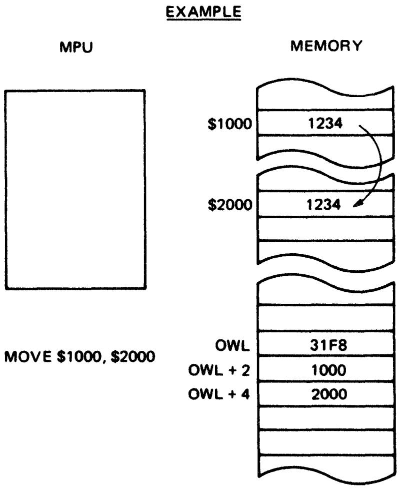
COMMENTS
- EA = (Next Word)
- 16-Bit Word is Sign Extended

- Machine Level Coding

NOT.L $2000

0100 0110 1011 1000

Not Instruction — L.W. — Absolute Short



COMMENTS
- EA = (Next Word)
- 16-Bit Word is Sign Extended

- Machine Level Coing

MOVE $1000, $2000

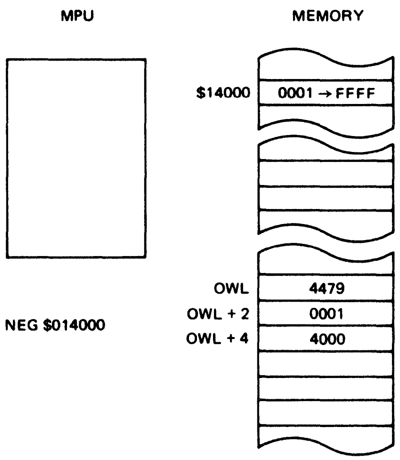
0011 0001 1111 1000

Move Word — Absolute Short — Absolute Short

Absolute Long Address

This address mode requires two words of extension. The address of the operand is developed by the concatenation of the extension words. The high-order part of the address is the first extension word; the low-order part of the address is the second extension word. The reference is classified as a data reference with the exception of the jump and jump to subroutine instructions.

EXAMPLE



COMMENTS

- EA = (Next Two Words)
- Machine Level Coding

NEG $014000

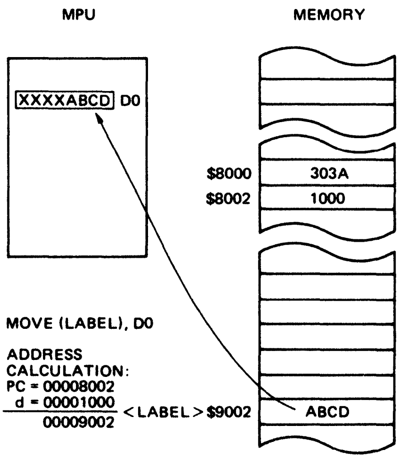
0100 0100 0111 1001

NEG Instruction — Size — Absolute Long

Program Counter With Displacement

This address mode requires one word of extension. The address of the operand is the sum of the address in the program counter and the sign-extended 16-bit displacement integer in the extension word. The value in the program counter is the address of the extension word. The reference is classified as a program reference.

EXAMPLE



COMMENTS

- EA = (PC) + d_{16}
- d_{16} is Sign Extended
- Machine Level Coding

MOVE (LABEL), D0

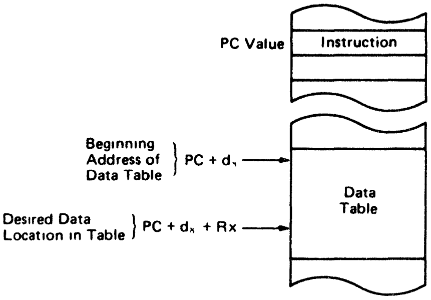
0011 0000 0011 1010

Move Word — Data Register Direct — PC with Displacement

Program Counter With Index

This address mode requires one word of extension. This address is the sum of the address in the program counter, the sign-extended displacement integer in the lower eight bits of the extension word, and the contents of the index register. The value in the program counter is the address of the extension word. This reference is classified as a program reference.

EA = (PC) + (Rx) + d_n


PC Value
Instruction
Beginning Address of Data Table
PC + d_n
Data Table
Desired Data Location in Table
PC + d_n + Rx

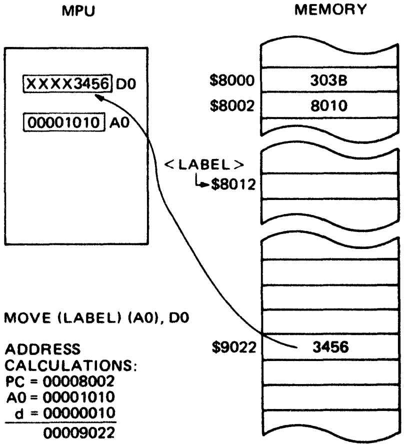
(NOTE)

Extension Word

15	14 13 12 11	10	9	8	7 6 5 4 3 2 1 0
D/A	Register	W/L	0	0	0	Displacement Integer

D/A : Data Register = 0, Address Register = 1
Register Index Register Number
W/L : Sign-extented, low order Word integer in Index Register = 0
Long Word in Index Register = 1

EXAMPLE



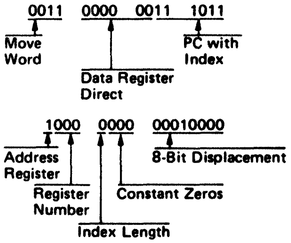
MOVE (LABEL) (A0), D0

ADDRESS CALCULATIONS:
PC = 00008002
A0 = 00001010
d = 00000010
00009022

COMMENTS

- EA = (PC) + (Rx) + d_8
  Where
  PC → Current Program Counter
  Rx → Designated Index Register (Either Data or Address Register)
  d_8 → 8-Bit Displacement
- Rx and d_8 are Sign Extended
- Rx may be Word or Long Word
  Long Word is Designated with Rx.L
- Machine Level Coding

MOVE (LABEL) (A0), D0



Immediate Data

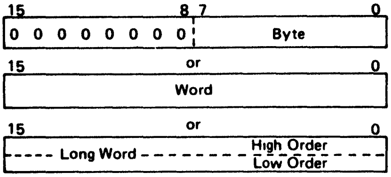
This address mode requires either one or two words of extension depending on the size of the operation.

Byte operation – operand is low order byte of extension word

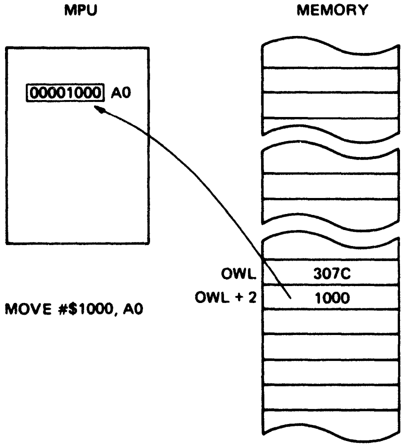
Word operation – operand is extension word

Long word operation – operand is in the two extension words, high-order 16 bits are in the first extension word, low-order 16 bits are in the second extension word.

Extension Word



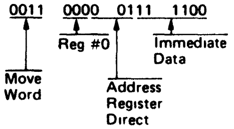
EXAMPLE



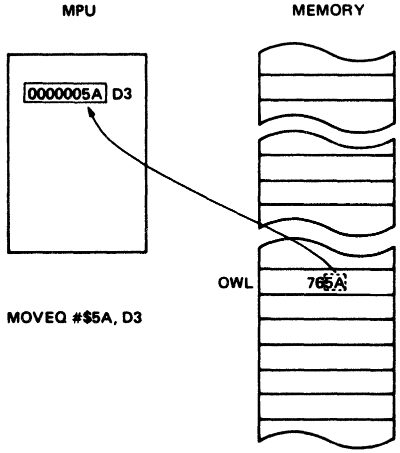
COMMENTS

- Data = Next Word(s)
- Data is Sign Extended for Address Register but not Data Register
- Machine Level Coding

MOVE #$1000, A0



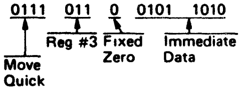
EXAMPLE



COMMENTS

- Inherent Data
- Data is Sign Extended to Long Word
- Destination must be a Data Register
- Machine Level Coding

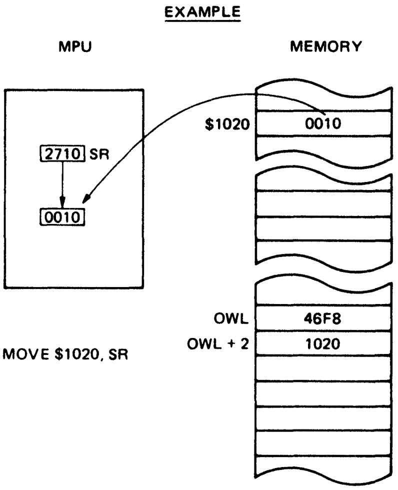
MOVEQ #$5A, D3



Condition Codes or Status Register

A selected set of instructions may reference the status register by means of the effective address field. These are:

ANDI to CCR
ANDI to SR
EORI to CCR
EORI to SR
ORI to CCR
ORI to SR
MOVE to CCR
MOVE to SR
MOVE from SR



COMMENTS
- EA = (Next Word)
- Note: This Example is a Privileged Instruction
- Machine Level Coding

MOVE $1020, SR

0100 0110 1111 1000

Move to SR　Absolute Short

• EFFECTIVE ADDRESS ENCODING SUMMARY

Table 4 is a summary of the effective addressing modes discussed in the previous paragraphs.

Table 4 Effective Address Encoding Summary

Addressing Mode	Mode	Register
Data Register Direct	000	register number
Address Register Direct	001	register number
Address Register Indirect	010	register number
Address Register Indirect with Postincrement	011	register number
Address Register Indirect with Predecrement	100	register number
Address Register Indirect with Displacement	101	register number
Address Register Indirect with Index	110	register number
Absolute Short	111	000
Absolute Long	111	001
Program Counter with Displacement	111	010
Program Counter with Index	111	011
Immediate	111	100

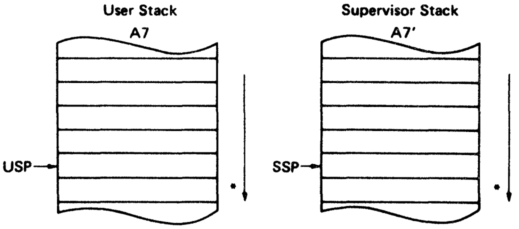
• IMPLICIT REFERENCE

Some instructions make implicit reference to the program counter (PC), the system stack pointer (SP), the supervisor stack pointer (SSP), the user stack pointer (USP), or the status register (SR).

SYSTEM STACK

The system stack is used implicitly by many instructions; user stacks and queues may be created and maintained through the addressing modes. Address register seven (A7) is the system stack pointer (SP). The system stack pointer is either the supervisor stack pointer (SSP) or the user stack pointer (USP), depending on the state of the S-bit in the status register. If the S-bit indicates supervisor state, SSP is the active system stack pointer, and the USP cannot be referenced as an address register. If the S-bit indicates user state, the USP is the active system stack pointer, and the SSP cannot be referenced. Each system stack fills from high memory to low memory.

SYSTEM STACK POINTERS



- Accessed when S = 0
- PC is Stacked on Subroutine Calls in User State

* Increasing Addresses

- Accessed when S = 1
- PC is Stacked on Subroutine Calls in Supervisor State
- Used for Exception Processing

The address mode SP @ - creates a new item on the active system stack, and the address mode SP @ + deletes an item from the active system stack.

The program counter is saved on the active system stack on subroutine calls, and restored from the active system stack on returns. On the other hand, both the program counter and the status register are saved on the supervisor stack during the processing of traps and interrupts. Thus, the correct execution of the supervisor state code is not dependent on the behavior of user code and user programs may use the user stack pointer arbitrarily.

In order to keep data on the system stack aligned properly, data entry on the stack is restricted so that data is always put in the stack on a word boundary. Thus byte data is pushed on or pulled from the system stack in the high order half of the word; the lower half is unchanged.

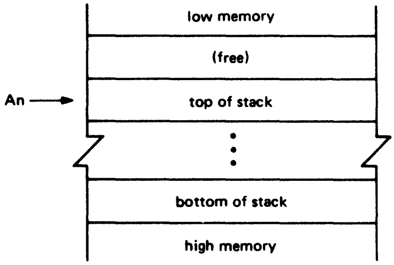
USER STACKS

User stacks can be implemented and manipulated by employing the address register indirect with postincrement and predecrement addressing modes. Using an address register (on of A0 through A6), the user may implement stacks which are filled either from high memory to low memory, or vice versa. The important things to remember are:

- using predecrement, the register is decremented before its contents are used as the pointer into the stack,
- using postincrement, the register is incremented after its contents are used as the pointer into the stack,
- byte data must be put on the stack in pairs when mixed with word or long data so that the stack will not get misaligned when the data is retrieved. Word and long accesses must be on word boundary (even) addresses.

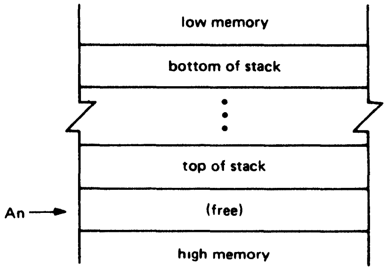
Stack growth from high to low memory is implemented with
An@- to push data on the stack,
An@+ to pull data from the stack.

After eigher a push or a pull operation, register An points to the last (top) item on the stack. This is illustrated as:



Stack growth from low to high memory is implemented with
An@+ to push data on the stack,
An@- to pull data from the stack.

After either a push or a pull operation, register An points to the next available space on the stack. This is illustrated as:

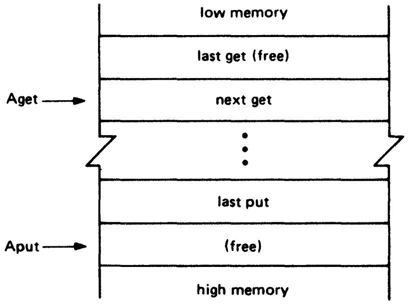


QUEUES

User queues can be implemented and manipulated with the address register indirect with postincrement or predecrement addressing modes. Using a pair of address registers (two of A0 through A6), the user may implement queues which are filled either from high memory to low memory, or vice versa. Because queues are pushed from one end and pulled from the other, two registers are used: the put and get pointers.

Queue growth from low to high memory is implemented with
Aput@+ to put data into the queue,
Aget@+ to get data from the queue.

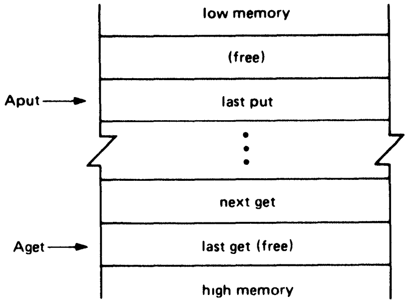
After a put operation, the put address register points to the next available space in the queue and the unchanged get address register points to the next item to remove from the queue. After a get operation, the get address register points to the next item to remove from the queue and the unchanged put address register points to the next available space in the queue. This is illustrated as.



If the queue is to be implemented as a circular buffer, the address register should be checked and, if necessary, adjusted before the put or get operation is performed. The address register is adjusted by subtracting the buffer length (in bytes).

Queue growth from high to low memory is implemented with
Aput@- to put data into the queue,
Aget@ - to get data from the queue.

After a put operation, the put address register points to the last item put in the queue, and the unchanged get address register points to the last item removed from the queue. After a get operation, the get address register points to the last item removed from the queue and the unchanged put address register points to the last item put in the queue. This is illustrated as:



If the queue is to be implemented as a circular buffer, the get or put operation should be performed first, and then the address register should be checked and, if necessary, adjusted The address register is adjusted by adding the buffer length (in bytes).

■INSTRUCTION SET SUMMARY

The following paragraphs contain an overview of the form and structure of the 68000 instruction set. The instructions form a set of tools that include all the machine functions to perform the following operations:

Data Movement
Integer Arithmetic
Logical
Shift and Rotate
Bit Manipulation
Binary Coded Decimal
Program Control
System Control

The complete range of instruction capabilities combined with the flexible addressing modes described previously provide a very flexible base for program development.

• DATA MOVEMENT OPERATIONS

The basic method of data acquisition (transfer and storage) is provided by the move (MOVE) instruction. The move instruction and the effective addressing modes allow both address and data manipulation. Data move instructions allow byte, word, and long word operands to be transferred from memory to memory, memory to register, register to memory, and register to memory, and register to register. Address move instructions allow word and long word operand transfers and ensure that only legal address manipulations are executed. In addition to the general move instruction there are several special data movement instructions: move multiple registers (MOVEM), move peripheral data (MOVEP), exchange registers (EXG), load effective address (LEA), push effective address (PEA), link stack (LINK), unlink stack (UNLK), and move quick (MOVEQ). Table 5 is a summary of the data movement operations.

Table 5 Data Movement Operations

Instruction	Operand Size	Operation
EXG	32	Rx ↔ Ry
LEA	32	EA → An
LINK	–	An → – (SP) SP → An ; SP + d → SP
MOVE	8, 16, 32	(EA)s → EAd
MOVEM	16, 32	(EA) → An, Dn An, Dn → EA
MOVEP	16, 32	(EA) → Dn Dn → EA
MOVEQ	8	#xxx → Dn
PEA	32	EA → –(SP)
SWAP	32	Dn[31:16] ↔ Dn[15:0]
UNLK	–	An → Sp ; (SP) + → An

(NOTES)
s = source –( ) = indirect with predecrement
d = destination ( ) + = indirect with postincrement
[ ] = bit numbers # = immediate data

• INTEGER ARITHMETIC OPERATIONS

The arithmetic operations include the four basic operations of add (ADD), subtract (SUB), multiply (MUL), and divide (DIV) as well as arithmetic compare (CMP), clear (CLR), and negate (NEG). The add and subtract instructions are available for both address and data operations, with data operations accepting all operand sizes. Address operations are limited to legal address size operands (16 or 32 bits). Data, address, and memory compare operations are also available. The clear and negate instructions may be used on all sizes of data operands.

The multiply and divide operations are available for signed and unsigned operands using word multiply to produce a long word product, and a long word dividend with word divisor to produce a word quotien with a word remainder.

Multiprecision and mixed size arithmetic can be accomplished using a set of extended instructions. These instructions are: add extended (ADDX), subtract extended (SUBX), sign extend (EXT), and negate binary with extend (NEGX).

A test operand (TST) instruction that will set the condition codes as a result of a compare of the operand with zero is also available. Test and set (TAS) is a synchronization instruction useful in multiprocessor systems. Table 6 is a summary of the integer arithmetic operations.

Table 6 Integer Arithmetic Operations

Instruction	Operand Size	Operation
ADD	8, 16, 32 16, 32	Dn + (EA) → Dn (EA)+ Dn → EA (EA) + #xxx → EA AN + (EA) → An
ADDX	8, 16, 32 16, 32	Dx + Dy + X → Dx −(Ax) + − (Ay) + X → (Ax)
CLR	8, 16, 32	(EA) → MPU 0 → EA
CMP	8, 16, 32 16, 32	Dn - (EA) (EA) - #xxx (Ax) + − (Ay) + An - (EA)
DIVS	32 ÷ 16	Dn÷(EA)→ Dn
DIVU	32 ÷ 16	Dn÷(EA)→ Dn
EXT	8 → 16 16 → 32	$(Dn)_8 \rightarrow Dn_{16}$ $(Dn)_{16} \rightarrow Dn_{32}$
MULS	16×16 → 32	Dn×(EA) → Dn
MULU	16×16 → 32	Dn×(EA) → Dn
NEG	8, 16, 32	0 - (EA) → EA
NEGX	8, 16, 32	0 - (EA) - X - EA
SUB	8, 16, 32 16, 32	Dn - (EA) → Dn (EA) - Dn → EA (EA) - #xxx → EA An - (EA) → An
SUBX	8, 16, 32	Dx - Dy - X → Dx −(Ax) − − (Ay) − X → (Ax)
TAS	8	(EA) - 0, 1 → EA[7]
TST	8, 16, 32	(EA) - 0

(NOTE) [ ] = bit number
− ( ) = indirect with predecrement
( ) + = indirect with postincrement
# = immediate data

• LOGICAL OPERATIONS

Logical operation instructions AND, OR, EOR, and NOT are available for all sizes of integer data operands. A similar set of immediate instructions (ANDI, ORI, and EORI) provide these logical operations with all sizes of immediate data. Table 7 is a summary of the logical operations.

Table 7 Logical Operations

Instruction	Operand Size	Operation
AND	8, 16, 32	Dn ∧ (EA) → Dn (EA)∧ Dn → EA (EA) ∧ =xxx → EA
OR	8, 16, 32	Dn v (EA) → Dn (EA) v Dn → EA (EA) v =xxx → EA
EOR	8, 16, 32	(EA)⊕Dy → EA (EA)⊕ #xxx → EA
NOT	8, 16, 32	~ (EA) → EA

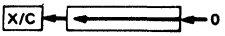
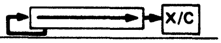
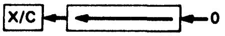
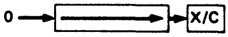
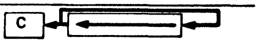
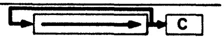
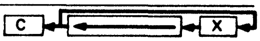
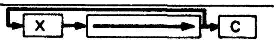
[NOTE] ~ = invert
v = logical OR
# = immediate data
Λ = logical AND
⊕ = exclusive OR

• SHIFT AND ROTATE OPERATIONS

Shift operations in both directions are provided by the arithmetic instructions ASR and ASL and logical shift instructions LSR and LSL. The rotate instructions (with and without extend) available are ROXR, ROXL, ROR, and ROL. All shift and rotate operations can be performed in either registers or memory. Register shifts and rotates support all operand sizes and allow a shift count specified in the instruction of one to eight bits, or 0 to 63 specified in a data register.

Memory shifts and rotates are for word operands only and allow only single-bit shifts or rotates. Table 8 is a summary of the shift and rotate operations.

Table 8 Shift and Rotate Operations

Instruction	Operand Size	Operation
ASL	8, 16, 32	
ASR	8, 16, 32	
LSL	8, 16, 32	
LSR	8, 16, 32	
ROL	8, 16, 32	
ROR	8, 16, 32	
ROXL	8, 16, 32	
ROXR	8, 16, 32	

• BIT MANIPULATION OPERATIONS

Bit manipulation operations are accomplished using the following instructions: bit test (BTST), bit test and set (BSET), bit test and clear (BCLR), and bit test and change (BCHG). Table 9 is a summary of the bit manipulation operations. (Bit 2 of the status register is Z.)

Table 9 Bit Manipulation Operations

Instruction	Operand Size	Operation
BTST	8, 32	~ bit of (EA) → Z
BSET	8, 32	(~ bit of (EA) → Z; 1 → bit of EA
BCLR	8, 32	(~ bit of (EA) → Z; 0 → bit of EA
BCHG	8, 32	(~ bit of (EA) → Z; ~ bit of (EA) → bit of EA

(Note) ~ = invert

• BINARY CODED DECIMAL OPERATIONS

Multiprecision arithmetic operations on binary coded decimal numbers are accomplished using the following instructions: add decimal with extend (ABCD), subtract decimal with extend (SBCD), and negate decimal with extend (NBCD). Table 10 is a summary of the binary coded decimal operations.

Table 10 Binary Coded Decimal Operations

Instruction	Operand Size	Operation
ABCD	8	$Dx_{10} + Dy_{10} + X \rightarrow Dx$ $-(Ax)_{10} + -(Ay)_{10} + X \rightarrow (Ax)$
SBCD	8	$Dx_{10} - Dy_{10} - X \rightarrow Dx$ $-(Ax)_{10} - -(Ay)_{10} - X \rightarrow (Ax)$
NBCD	8	$0 - (EA)_{10} - X \rightarrow EA$

− ( ) = indirect with predecrement

• PROGRAM CONTROL OPERATIONS

Program control operations are accomplished using a series of conditional and unconditional branch instructions and return instructions. These instructions are summarized in Table 11.

The conditional instructions provide setting and branching for the following conditions:

CC	– carry clear	LS	– low or same
CS	– carry set	LT	– less than
EQ	– equal	MI	– minus
F	– never true	NE	– not equal
GE	– greater or equal	PL	– plus
GT	– greater than	T	– always true
HI	– high	VC	– no overflow
LE	– less or equal	VS	– overflow

Table 11 Program Control Operations

Instruction	Operation
Conditional	
B_{CC}	Branch conditionally (14 conditions) 8- and 16-bit displacement
DB_{CC}	Test condition, decrement, and branch 16-bit displacement
S_{CC}	Set byte conditionally (16 conditions)
Unconditional	
BRA	Branch always 8-and 16-bit displacement
BSR	Branch to subroutine 8- and 16-bit displacement
JMP	Jump
JSR	Jump to subroutine
Returns	
RTR	Return and restore condition codes
RTS	Return from subroutine

• SYSTEM CONTROL OPERATIONS

System control operations are accomplished by using privileged instructions, trap generating instructions, and instructions that use or modify the status register. These instructions are summarized in Table 12.

Table 12 System Control Operations

Instruction	Operation
Privileged	
RESET	Reset external devices
RTE	Return from exception
STOP	Stop program execution
ORI to SR	Logical OR to status register
MOVE USP	Move user stack pointer
ANDI to SR	Logical AND to status register
EORI to SR	Logical EOR to status register
MOVE EA to SR	Load new status register
Trap Generating	
TRAP	Trap
TRAPV	Trap on overflow
CHK	Check register against bounds
Status Register	
ANDI to CCR	Logical AND to condition codes
EORI to CCR	Logical EOR to condition codes
MOVE EA to CCR	Load new condition codes
ORI to CCR	Logical OR to condition codes
MOVE SR to EA	Store status register

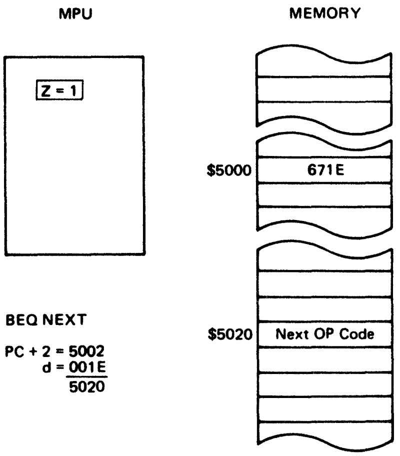
• BRANCH INSTRUCTION ADDRESSING

BRANCH INSTRUCTION FORMAT

	15 8	7 0
Operation Word	Operation Code	8 bit Displacement
Extension Word	16 bit Displacement if 8 bit Displacement = 0	

RELATIVE, FORWARD REFERENCE, 8-BIT OFFSET

EXAMPLE



COMMENTS

- Offset Contained in 8 LSBs of Op Word
- Offset is 2's Complement Number
- If Offset = 0 then Word Offset is Used
- Machine Level Coding

BEQ NEXT

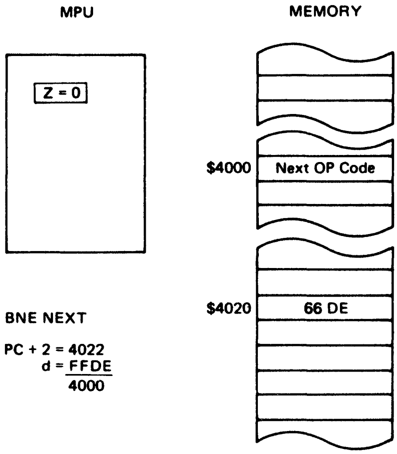
0110 0111 0001 1110

Branch Offset

Branch If
Equal

RELATIVE, BACKWARD REFERENCE 8-BIT OFFSET

EXAMPLE



BNE NEXT

PC + 2 = 4022
d = FFDE
4000

COMMENTS

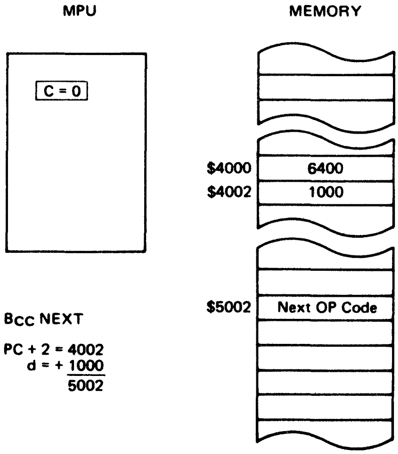
- Offset Contained in 8 LSBs of Op Word
- Offset is 2's Complement Number
- If Offset = 0 then Word Offset is Used
- Machine Level Coding

BNE NEXT

```
0110   0110   1101  1110
 Branch   |          Offset
       Branch If
       Not Equal
```

RELATIVE, FORWARD REFERENCE, 16-BIT OFFSET

EXAMPLE



Bcc NEXT

PC + 2 = 4002
d = + 1000
5002

COMMENTS

- Offset in Next Word
- 8-Bit Offset Field = 0
- 2's Complement Offset
- Machine Level Coding

Bcc NEXT

```
0110   0100   0000  0000
 Branch   |       Zero Offset
       Branch If
       Carry Clear
```

■ SIGNAL AND BUS OPERATION DESCRIPTION

The following paragraphs contain a brief description of the input and output signals. A discussion of bus operation during the various machine cycles and operations is also given.

(NOTE) The terms **assertion** and **negation** will be used extensively. This is done to avoid confusion when dealing with a mixture of "active-low" and "active-high" signals. The term assert or assertion is used to indicate that a signal is active or true independent of whether that voltage is low or high. The term negate or negation is used to indicate that a signal is inactive or false.

• SIGNAL DESCRIPTION

The input and output signals can be functionally organized into the groups shown in Figure 14. The following paragraphs provide a brief description of the signals and also a reference (if applicable) to other paragraphs that contain more detail about the function being performed.

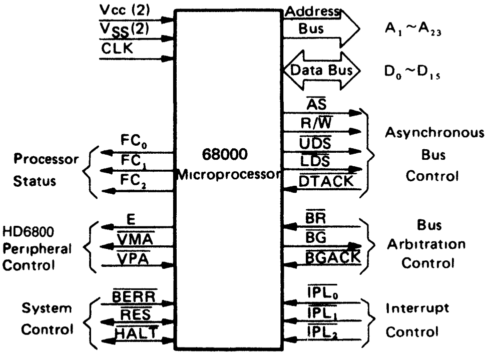


Figure 14 Input and Output Signals

ADDRESS BUS (A_1 through A_{23})

This 23-bit, unidirectional, three-state bus is capable of addressing 8 megawords of data. It provides the address for bus operation during all cycles except interrupt cycles. During interrupt cycles, address lines A_1, A_2, and A_3. Provide information about what level interrupt is being serviced while address lines A_4 through A_{23} are all set to a logic high.

DATA BUS (D_0 through D_{15})

This 16-bit, bidirectional, three-state bus is the general purpose data path. It can transfer and accept data in either word or byte length. During an interrupt acknowledge cycle, an external device supplies the vector number on data lines D_0 through D_7.

ASYNCHRONOUS BUS CONTROL

Asynchronous data transfer are handled using the following control signals: address strobe, read/write, upper and lower data strobes, and data transfer acknowledge. These signals are explained in the following paragraphs.

Address Strobe ($\overline{AS}$)

This signal indicates that there is a valid address on the address bus.

Read/Write (R/$\overline{W}$)

This signal defines the data bus transfer as a read or write cycle. The R/$\overline{W}$ signal also works in conjunction with the upper and lower data strobes as explained in the following paragraph.

Upper and Lower Data Strobes ($\overline{UDS}$, $\overline{LDS}$)

These signals control the data on the data bus, as shown in Table 13. When the R/$\overline{W}$ line is high, the processor will read from the data bus as indicated. When the R/$\overline{W}$ line is low, the processor will write to the data bus as shown.

Table 13 Data Strobe Control of Data Bus

$\overline{UDS}$	$\overline{LDS}$	R/$\overline{W}$	$D_8 \sim D_{15}$	$D_0 \sim D_7$
High	High	—	No valid data	No valid data
Low	Low	High	Valid data bits 8 ~ 15	Valid data bits 0 ~ 7
High	Low	High	No valid data	Valid data bits 0 ~ 7
Low	High	High	Valid data bits 8 ~ 15	No valid data
Low	Low	Low	Valid data bits 8 ~ 15	Valid data bits 0 ~ 7
High	Low	Low	Valid data bits 0 ~ 7*	Valid data bits 0 ~ 7
Low	High	Low	Valid data bits 8 ~ 15	Valid data bits 8 ~ 15*

* These conditions are a result of current implementation and may not appear on future devices

Data Transfer Acknowledge ($\overline{DTACK}$)

This input indicates that the data transfer is completed. When the processor recognizes $\overline{DTACK}$ during a read cycle, data is latched and the bus cycle terminated When $\overline{DTACK}$ is recognized during a write cycle, the bus cycle is terminated. (Refer to ASYNCHRONOUS VERSUS SYNCHRONOUS OPERATION)

BUS ARBITRATION CONTROL

These three signals form a bus arbitration circuit to determine which device will be the bus master device.

Bus Request ($\overline{BR}$)

This input is wire ORed with all other devices that could be bus masters. This input indicates to the processor that some other device desires to become the bus master.

Bus Grant ($\overline{BG}$)

This output indicates to all other potential bus master devices that the processor will release bus control at the end of the current bus cycle.

Bus Grand Acknowledge ($\overline{BGACK}$)

This input indicates that some other device has become the bus master This signal cannot be asserted until the following four conditions are met

(1) A Bus Grant has been received

(2) Address Strobe is inactive which indicates that the microprocessor is not using the bus

(3) Data Transfer Acknowledge is inactive which indicates

that neither memory nor peripherals are using the bus

(4) Bus Grant Acknowledge is inactive which indicates that no other device is still claiming bus mastership.

INTERRUPT CONTROL ($\overline{IPL_0}$, $\overline{IPL_1}$, $\overline{IPL_2}$)

These input pins indicate the encoded priority lèvel of the device requesting an interrupt Level seven is the highest priority while level zero indicates that no interrupts are requested. Level seven can not be masked. The least significant bit is given in $\overline{IPL_0}$ and the most significant bit is contained in $\overline{IPL_2}$. These lines must remain stable until the processor signals interrupt acknowledge (FC0 ~ FC2 are all high) to insure that the interrupt is recognized.

SYSTEM CONTROL

The system control inputs are used to either reset or halt the processor and to indicate to the processor that bus errors have occurred The three system control inputs are explained in the following paragraphs.

Bus Error ($\overline{BERR}$)

This input informs the processor that there is a problem with the cycle currently being executed. Problems may be a result of.

(1) Nonresponding devices

(2) Interrupt vector number acquisition failure

(3) Illegal access request as determined by a memory management unit

(4) Other application dependent errors.

The bus error signal interacts with the halt signal to determine if exception processing should be performed or if the current bus cycle should be retried.

Refer to **BUS ERROR AND HALT OPERATION** paragraph for additional information about the interaction of the bus error and halt signals.

Reset ($\overline{RES}$)

This bidirectional signal line acts to reset (initiate a system initialization sequence) the processor in response to an external reset signal. An internally generated reset (result of a RESET instruction) causes all external devices to be reset and the internal state of the processor is not affected. A total system reset (processor and external devices) is the result of external HALT and RESET signals applied at the same time. Refer to **RESET OPERATION** paragraph for additional information about reset operation.

Halt ($\overline{HALT}$)

When this bidirectional line is driven by an external device, it will cause the processor to stop at the completion of the current bus cycle. When the processor has been halted using this input, all control signals are inactive and all three-state lines are put in their high-impedance state. Refer to **BUS ERROR AND HALT OPERATION** paragraph for additional information about the interaction between the halt and bus error signals.

When the processor has stopped executing instructions, such as in a double bus fault condition, the halt line is driven by the processor to indicate to external devices that the processor has stopped.

HMCS6800 PERIPHERAL CONTROL

These control signals are used to allow the interfacing of synchronous HD6800 peripheral devices with the asynchronous 68000. These signals are explained in the following paragraphs.

Enable (E)

This signal is the standard enable signal common to all HD6800 type peripheral devices. The period for this output is ten 68000 clock periods (six clocks low; four clocks high). Enable is generated by an internal ring counter which may come up in any state (i.e., at power on, it is impossible to guarantee phase relationship of E to CLK), E is a free-running clock and runs regardless of the state of the bus on the MPU.

Valid Peripheral Address ($\overline{VPA}$)

This input indicates that the device or region addressed is a HD6800 family device and that data transfer should be synchronized with the enable (E) signal. This input also indicates that the processor should use automatic vectoring for an interrupt. Refer to **INTERFACE WITH HD6800 PERIPHERALS. ALS.**

Valid Memory Address ($\overline{VMA}$)

This output is used to indicate to HD6800 peripheral devices that there is a valid address on the address bus and the processor is synchronized to enable. This signal only responds to a valid peripheral address ($\overline{VPA}$) input which indicates that the peripheral is a HD6800 family device.

PROCESSOR STATUS (FC_0, FC_1, FC_2)

These function code outputs indicate the state (user or supervisor) and the cycle type currently being executed, as shown in Table14. The information indicated by the function code outputs is valid whenever address strobe ($\overline{AS}$) is active.

Table 14 Function Code Outputs

FC_2	FC_1	FC_0	Cycle Type
Low	Low	Low	(Undefined, Reserved)
Low	Low	High	User Data
Low	High	Low	User Program
Low	High	High	(Undefined, Reserved)
High	Low	Low	(Undefined, Reserved)
High	Low	High	Superviser Data
High	High	Low	Supervisor Program
High	High	High	Interrupt Acknowledge

CLOCK (CLK)

The clock input is a TTL-compatible signal that is internally buffered for development of the internal clocks needed by the processor. The clock input should not be gated off at any time, and the clock signal must conform to minimum and maximum pulse width time.

SIGNAL SUMMARY

Table 15 is a summary of all the signals discussed in the previous paragraphs.

• BUS OPERATION

The following paragraphs explain control signal and bus operation during data transfer operations, bus arbitration, bus error and halt conditions, and reset operation.

Table 15 Signal Summary

Signal Name	Mnemonic	Input/Output	Active State	Three State	
				On $\overline{BGACK}$	On $\overline{HALT}$
Address Bus	$A_1 \sim A_{23}$	output	high	yes	yes
Data Bus	$D_0 \sim D_{15}$	input/output	high	yes	yes
Address Strobe	$\overline{AS}$	output	low	yes	no
Read/Write	$R/\overline{W}$	output	read-high write-low	yes	no
Upper and Lower Data Strobes	$\overline{UDS}, \overline{LDS}$	output	low	yes	no
Data Transfer Acknowledge	$\overline{DTACK}$	input	low	no	no
Bus Request	$\overline{BR}$	input	low	no	no
Bus Grant	$\overline{BG}$	output	low	no	no
Bus Grant Acknowledge	$\overline{BGACK}$	input	low	no	no
Interrupt Priority Level	$\overline{IPL_0}, \overline{IPL_1}, \overline{IPL_2}$	input	low	no	no
Bus Error	$\overline{BERR}$	input	low	no	no
Reset	$\overline{RES}$	input/output	low	no*	no*
Halt	$\overline{HALT}$	input/output	low	no*	no*
Enable	E	output	high	no	no
Valid Memory Address	$\overline{VMA}$	output	low	yes	no
Valid Peripheral Address	$\overline{VPA}$	input	low	no	no
Function Code Output	FC_0, FC_1, FC_2	output	high	yes	no
Clock	CLK	input	high	no	no
Power Input	V_{CC}	input	–	–	–
Ground	V_{SS}	input	–	–	–

* Open drain

DATA TRANSFER OPERATIONS

Transfer of data between devices involve the following leads:

(1) Address Bus A_1 through A_{23}

(2) Data Bus D_0 through D_{15}

(3) Control Signals

The address and data buses are separate parallel buses used to transfer data using an asynchronous bus structure. In all cycles, the bus master assumes responsibility for deskewing all signals it issues at both the start and end of a cycle. In addition, the bus master is responsible for deskewing the acknowledge and data signals from the slave device.

The following paragraphs explain the read, write, and read-modify-write cycles. The indivisible read-modify-write cycle is the method used by the 68000 for interlocked multiprocessor communications.

Read Cycle

During a read cycle, the processor receives data from memory or a peripheral device. The processor reads bytes of data in all cases. If the instruction specifies a word (or double word) operation, the processor reads both upper and lower bytes simultaneously by asserting both upper and lower data strobes. When the instruction specifies byte operation, the processor uses an internal Ao bit to determine which byte to read and then issues the data strobe required for that byte. For bytes operations, when the Ao bit equals zero, the upper data strobe is issued. When the Ao bit equals one, the lower data strobe is issued. When the data is received, the processor correctly positions it internally.

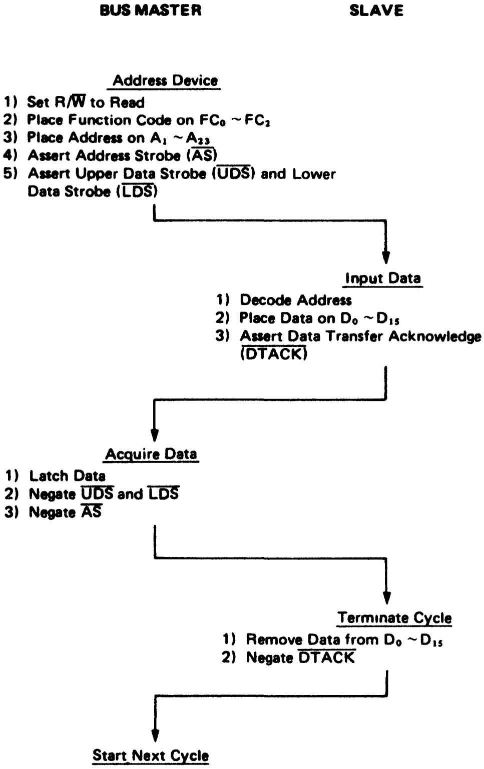
A word read cycle flow chart is given in Figure 15. A byte read cycle flow chart is given in Figure 16. Read cycle timing is given in Figure 17. Figure 18 details word and byte read cycle operations. Refer to these illustrations during the following detailed.

At state zero (S0) in the read cycle, the address bus (A_1 through A_{23}) is in the high impedance state. A function code is asserted on the function code output line (FC_0 through FC_2). The read/write (R/W) signal is switched high to indicate a read cycle. One half clock cycle later, at state 1, the address bus is released from the high impedance state. The function code outputs indicate which address space that this cycle will operate on.

In state 2, the address strobe ($\overline{AS}$) is asserted to indicate that there is a valid address on the address bus and the upper and lower data strobe ($\overline{UDS}$, $\overline{LDS}$) is asserted as required. The memory or peripheral device uses the address bus and the address strobe to determine if it has been selected. The selected device uses the read/write signal and the data strobe to place its information on the data bus. Concurrent with placing data on the data bus, the selected device asserts data transfer acknowledge ($\overline{DTACK}$).

Data transfer acknowledge must be present at the processor at the start of state 5 or the processor will substitute wait states for states 5 and 6. State 5 starts the synchronization of the returning data transfer acknowledge. At the end of state 6 (beginning of state 7) incoming data is latched into an internal data bus holding register.

During state 7, address strobe and the upper and/or lower data strobes are negated. The address bus is held valid through state 7 to allow for static memory operation and signal skew.



BUS MASTER

SLAVE

Address Device

1) Set R/$\overline{W}$ to Read
2) Place Function Code on $FC_0 \sim FC_2$
3) Place Address on $A_1 \sim A_{23}$
4) Assert Address Strobe ($\overline{AS}$)
5) Assert Upper Data Strobe ($\overline{UDS}$) and Lower Data Strobe ($\overline{LDS}$)

Input Data

1) Decode Address
2) Place Data on $D_0 \sim D_{15}$
3) Assert Data Transfer Acknowledge ($\overline{DTACK}$)

Acquire Data

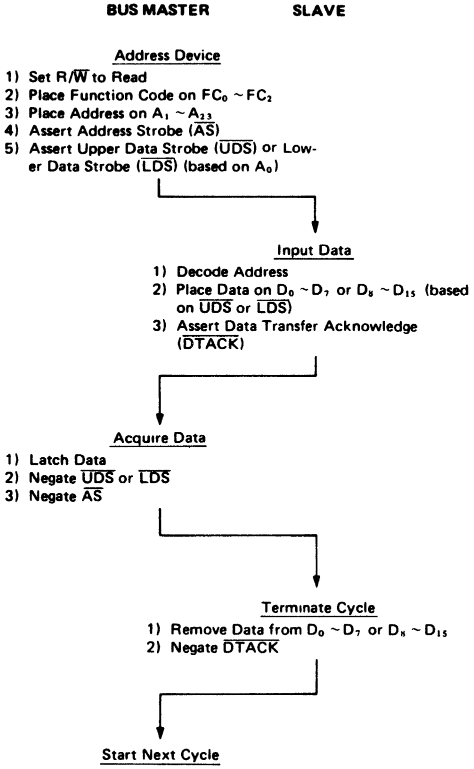
1) Latch Data
2) Negate $\overline{UDS}$ and $\overline{LDS}$
3) Negate $\overline{AS}$

Terminate Cycle

1) Remove Data from $D_0 \sim D_{15}$
2) Negate $\overline{DTACK}$

Start Next Cycle

Figure 15 Word Read Cycle Flow Chart



BUS MASTER

SLAVE

Address Device

1) Set R/$\overline{W}$ to Read
2) Place Function Code on $FC_0 \sim FC_2$
3) Place Address on $A_1 \sim A_{23}$
4) Assert Address Strobe ($\overline{AS}$)
5) Assert Upper Data Strobe ($\overline{UDS}$) or Lower Data Strobe ($\overline{LDS}$) (based on A_0)

Input Data

1) Decode Address
2) Place Data on $D_0 \sim D_7$ or $D_8 \sim D_{15}$ (based on $\overline{UDS}$ or $\overline{LDS}$)
3) Assert Data Transfer Acknowledge ($\overline{DTACK}$)

Acquire Data

1) Latch Data
2) Negate $\overline{UDS}$ or $\overline{LDS}$
3) Negate $\overline{AS}$

Terminate Cycle

1) Remove Data from $D_0 \sim D_7$ or $D_8 \sim D_{15}$
2) Negate $\overline{DTACK}$

Start Next Cycle

Figure 16 Byte Read Cycle Flow Chart

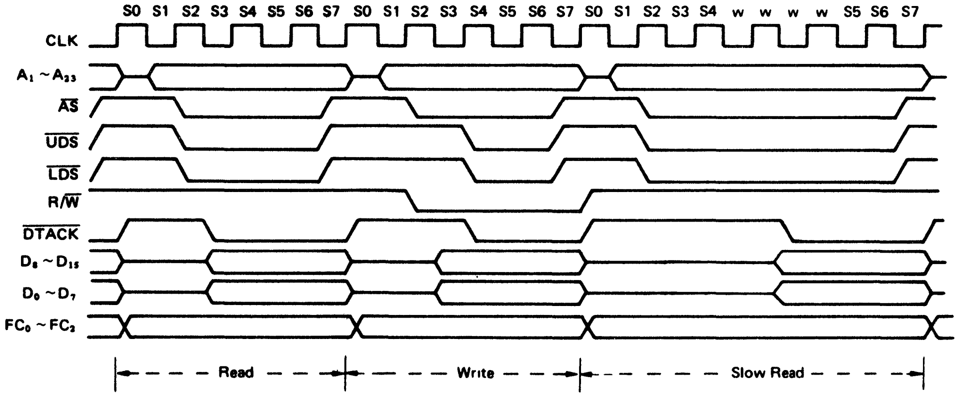


Figure 17 Read and Write Cycle Timing Diagram

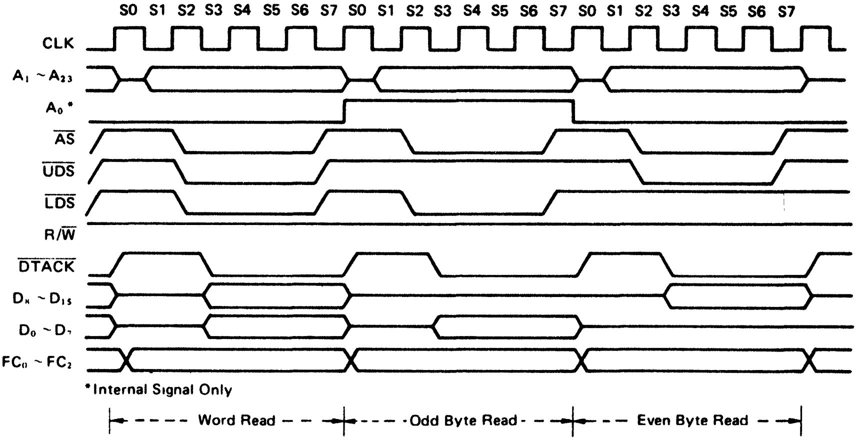


Figure 18 Word and Byte Read Cycle Timing Diagram

The read/write signal and the function code outputs also remain valid through state 7 to ensure a correct transfer operation. The slave device keeps its data asserted until it detects the negation of either the address strobe or the upper and/or lower data strobe. The slave device must remove its data and data transfer acknowledge within one clock period of recognizing the negation of the address or data strobes. Note that the data bus might not become free and data transfer acknowledge might not be removed until state 0 or 1.

When address strobe is negated, the slave device is released. Note that a slave device must remain selected as long as address strobe is asserted to ensure the correct functioning of the read-modify-write cycle.

Write Cycle

During a write cycle, the processor sends data to memory or a peripheral device. The processor writes bytes of data in all cases. If the instruction specifies a word operation, the processor writes both bytes. When the instruction specifies a byte operation, the processor uses an internal A_0 bit to determine which byte to write and then issues the data strobe required for that byte. For byte operations, when the A_0 bit equals zero, the upper data strobe is issued. When the A_0 bit equals one, the lower data strobe is issued. A word write cycle flow chart is given in Figure 19. A byte write cycle flow chart is given in Figure 20. Write cycle timing is given in Figure 17. Figure 21 details word and byte write cycle operation. Refer to these illustrations during the following detailed discussion.

At state zero (S0) in the write cycle, the address bus (A_1 through A_{23}) is in the high impedance state. A function code is asserted on the function code output line (FC_0 through FC_2).

(NOTE) The read/write (R/$\overline{W}$) signal remains high until state 2 to prevent bus conflicts with preceding read cycles. The data bus is not driven until state 3.

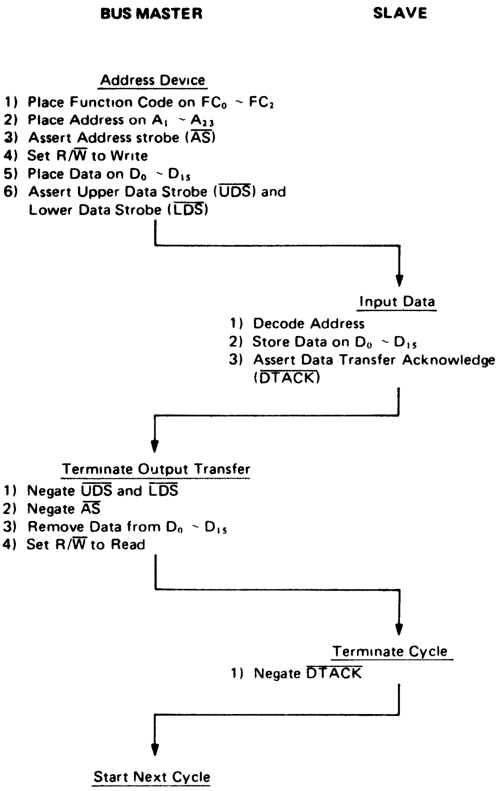
One half clock later, at state 1, the address bus is released from the high impedance state. The function code outputs indicate which address space that this cycle will operate on.

In state 2, the address strobe ($\overline{AS}$) is asserted to indicate that there is a valid address on the address bus. The memory or peripheral device uses the address bus and the address strobe to determine if it has been selected. During state 2, the read/write signal is switched low to indicate a write cycle. When external processor data bus buffers are required, the read/write line provides sufficient directional control. Data is not asserted during this state to allow sufficient turn around time for external data buffers (if used). Data is asserted onto the data bus during state 3.

In state 4, the data strobes are asserted as required to indicate that the data bus is stable. The selected device uses the read/write signal and the data strobes to take its information from the data bus. The selected device asserts data transfer acknowledge ($\overline{DTACK}$) when it has successfully stored the data.

Data transfer acknowledge must be present at the processor at the start of state 5 or the processor will substitute wait states for states 5 and 6. State 5 starts the synchronization of the returning data transfer acknowledge.

During state 7, address strobe and the upper and/or lower data strobes are negated. The address and data buses are held valid through state 7 to allow for static memory operation and signal skew. The read/write signal and the function code outputs also remain valid through state 7 to ensure a correct transfer operation. The slave device keeps its data transfer acknowledge asserted until it detects the negation of either the address strobe or the upper and/or lower data strobe. The slave device must remove its data transfer acknowledge within one clock period after recognizing the negation of the address or data strobes. Note that the processor releases the data bus at the end of state 7 but that data transfer acknowledge might not be removed until state 0 or 1. When address strobe is negated, the slave device is released.



BUS MASTER

SLAVE

Address Device

1) Place Function Code on $FC_0 \sim FC_2$
2) Place Address on $A_1 \sim A_{23}$
3) Assert Address strobe ($\overline{AS}$)
4) Set $R/\overline{W}$ to Write
5) Place Data on $D_0 \sim D_{15}$
6) Assert Upper Data Strobe ($\overline{UDS}$) and Lower Data Strobe ($\overline{LDS}$)

Input Data

1) Decode Address
2) Store Data on $D_0 \sim D_{15}$
3) Assert Data Transfer Acknowledge ($\overline{DTACK}$)

Terminate Output Transfer

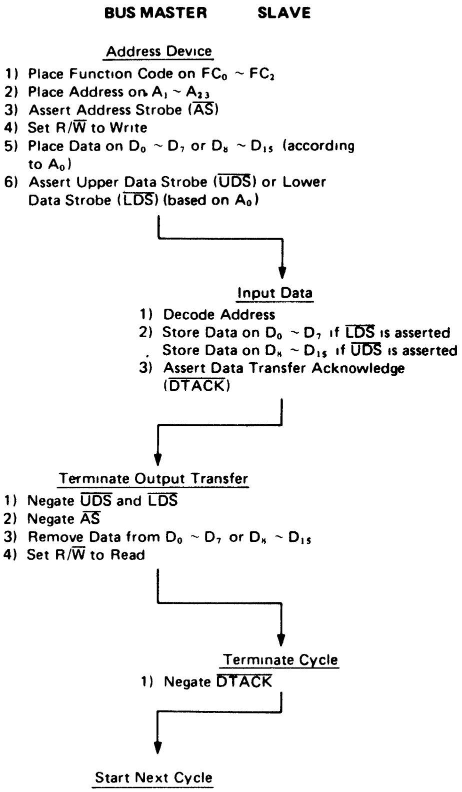
1) Negate $\overline{UDS}$ and $\overline{LDS}$
2) Negate $\overline{AS}$
3) Remove Data from $D_0 \sim D_{15}$
4) Set $R/\overline{W}$ to Read

Terminate Cycle

1) Negate $\overline{DTACK}$

Start Next Cycle

Figure 19 Word Write Cycle Flow Chart



BUS MASTER

SLAVE

Address Device

1) Place Function Code on $FC_0 \sim FC_2$
2) Place Address on $A_1 \sim A_{23}$
3) Assert Address Strobe ($\overline{AS}$)
4) Set $R/\overline{W}$ to Write
5) Place Data on $D_0 \sim D_7$ or $D_8 \sim D_{15}$ (according to A_0)
6) Assert Upper Data Strobe ($\overline{UDS}$) or Lower Data Strobe ($\overline{LDS}$) (based on A_0)

Input Data

1) Decode Address
2) Store Data on $D_0 \sim D_7$ if $\overline{LDS}$ is asserted
   Store Data on $D_8 \sim D_{15}$ if $\overline{UDS}$ is asserted
3) Assert Data Transfer Acknowledge ($\overline{DTACK}$)

Terminate Output Transfer

1) Negate $\overline{UDS}$ and $\overline{LDS}$
2) Negate $\overline{AS}$
3) Remove Data from $D_0 \sim D_7$ or $D_8 \sim D_{15}$
4) Set $R/\overline{W}$ to Read

Terminate Cycle

1) Negate $\overline{DTACK}$

Start Next Cycle

Figure 20 Byte Write Cycle Flow Chart

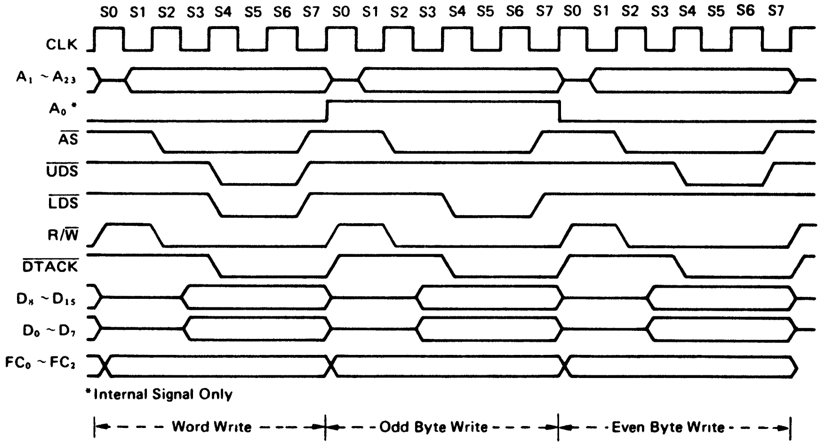


Figure 21 Word and Byte Write Cycle Timing Diagram

Read-Modify-Write Cycle

The read-modify-write cycle performs a read, modifies the data in the arithmetic-logic unit, and writes the data back to the same address. In the 68000 this cycle is indivisible in that the address strobe is asserted throughout the entire cycle. The test and set (TAS) instruction uses this cycle to provide meaningful communication between processors in a multiple processor environment. This instruction is the only instruction that uses the read-modify-write cycle and since the test and set instruction only operates on bytes, all read-modify-write cycles are byte operations. A read-modify-write cycle flow chart is given in Figure 22 and a timing diagram is given in Figure 23. Refer to these illustrations during the following detailed discussions.

At state zero (S0) in the read-modify-write cycle, the address bus (A_1 through A_{23}) is in the high impedance state. A function code is asserted on the function code output line (FC_0 through FC_2). The read/write ($R/\overline{W}$) signal is switched high to indicate a read cycle. One half clock cycle later, at state 1, the address bus is released from the high impedance state. The function code outputs indicate which address space that this cycle will operate on.

In state 2, the address strobe (AS) is asserted to indicate that there is a valid address on the address bus and the upper or lower data strobe ($\overline{\text{UDS}}$, $\overline{\text{LDS}}$) is asserted as required. The memory or peripheral device uses the address bus and the address strobe to determine if it has been selected. The selected device uses the read/write signal and the data strobe to place its information on the data bus. Concurrent with placing data on the data bus, the selected device asserts data transfer acknowledge ($\overline{\text{DTACK}}$).

Data transfer acknowledge must be present at the processor at the start of state 5 or the processor will substitute wait states for states 5 and 6. State 5 starts the synchronization of the returning data transfer acknowledge. At the end of state 6 (beginning of state 7) incoming data is latched into an internal data bus holding register.

During state 7, the upper or lower data strobe is negated. The address bus, address strobe, read/write signal, and function code outputs remain as they were in preparation for the write portion of the cycle. The slave device keeps its data asserted until it detects the negation of the upper or lower data strobe. The slave device must remove its data and data transfer acknowledge within one clock period of recognizing the negation of the data strobes. Internal modification of data may occur from state 8 to state 11.

(NOTE) The read/write signal remains high until state 14 to prevent bus conflicts with the preceding read portion of the cycle and the data bus is not asserted by the processor until state 15.

In state 14, the read/write signal is switched low to indicate a write cycle. When external processor data bus buffers are required, the read/write line provides sufficient directional control. Data is not asserted during this state to allow sufficient turn around time for external data buffers (if used). Data is asserted onto the data bus during state 15.

In state 16, the data strobe is asserted as required to indicate that the data bus is stable. The selected device uses the read/write signal and the data strobe to take its information from the data bus. The selected device asserts data transfer acknowledge ($\overline{\text{DTACK}}$) when it has successfully stored its data.

Data transfer acknowledge must be present at the processor at the start of state 17 or the processor will substitute wait states for states 17 and 18. State 17 starts the synchronization of the returning data transfer acknowledge for the write portion of the cycle. The bus interface circuitry issues requests for subsequent internal cycles during state 18.

During state 19, address strobe and the upper or lower data strobe is negated. The address and data buses are held valid through state 19 to allow for static memory operation and signal skew. The read/write signal and the function code outputs also remain valid through state 19 to ensure a correct transfer operation. The slave device keeps its data transfer acknowledge asserted until it detects the negation of either the address strobe or the upper or lower data strobe. The slave device must remove its data transfer acknowledge within once clock period after recognizing the negation of the address or data strobes. Note that the processor releases the data bus at the end of state 19 but that data transfer acknowledge might not be removed until state 0 or 1. When address strobe is negated the slave device is released.

BUS ARBITRATION

Bus arbitration is a technique used by master-type devices to request, be granted, and acknowledge bus mastership. In its simplest form, it consists of:

(1) Asserting a bus mastership request.

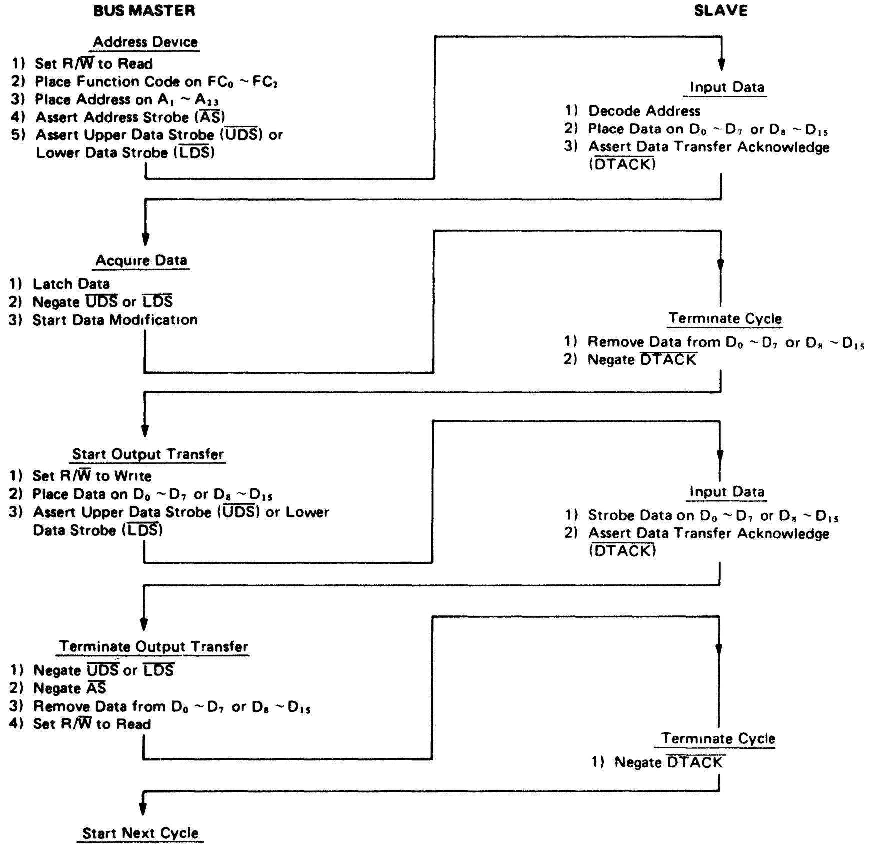
(2) Receiving a grant that the bus is available at the end of the current cycle.

(3) Acknowledging that mastership has been assumed.

Figure 24 is a flow chart showing the detail involved in a request from a single device. Figure 25 is a timing diagram for the same operations. This technique allows processing of bus requests during data transfer cycles.

The timing diagram shows that the bus request is negated at the time that an acknowledge is asserted. This type of operation would be true for a system consisting of the processor and one device capable of bus mastership. In systems having a number of devices capable of bus mastership, the bus request line from each device is wire ORed to the processor. In this system, it is easy to see that there could be more than one bus request being made. The timing diagram shows that the bus grant signal is negated a few clock cycles after the transition of the acknowledge ($\overline{\text{BGACK}}$) signal.

However, if the bus requests are still pending, the processor will assert another bus grant within a few clock cycles after it was negated. This additional assertion of bus grant allows external arbitration circuitry to select the next bus master before the current bus master has completed its requirements. The following paragraphs provide additional information about the three steps in the arbitration process.



BUS MASTER

SLAVE

Address Device

1) Set R/$\overline{W}$ to Read
2) Place Function Code on $FC_0 \sim FC_2$
3) Place Address on $A_1 \sim A_{23}$
4) Assert Address Strobe ($\overline{AS}$)
5) Assert Upper Data Strobe ($\overline{UDS}$) or Lower Data Strobe ($\overline{LDS}$)

Input Data

1) Decode Address
2) Place Data on $D_0 \sim D_7$ or $D_8 \sim D_{15}$
3) Assert Data Transfer Acknowledge ($\overline{DTACK}$)

Acquire Data

1) Latch Data
2) Negate $\overline{UDS}$ or $\overline{LDS}$
3) Start Data Modification

Terminate Cycle

1) Remove Data from $D_0 \sim D_7$ or $D_8 \sim D_{15}$
2) Negate $\overline{DTACK}$

Start Output Transfer

1) Set R/$\overline{W}$ to Write
2) Place Data on $D_0 \sim D_7$ or $D_8 \sim D_{15}$
3) Assert Upper Data Strobe ($\overline{UDS}$) or Lower Data Strobe ($\overline{LDS}$)

Input Data

1) Strobe Data on $D_0 \sim D_7$ or $D_8 \sim D_{15}$
2) Assert Data Transfer Acknowledge ($\overline{DTACK}$)

Terminate Output Transfer

1) Negate $\overline{UDS}$ or $\overline{LDS}$
2) Negate $\overline{AS}$
3) Remove Data from $D_0 \sim D_7$ or $D_8 \sim D_{15}$
4) Set R/$\overline{W}$ to Read

Terminate Cycle

1) Negate $\overline{DTACK}$

Start Next Cycle

Figure 22 Read-Modify-Write Cycle Flow Chart

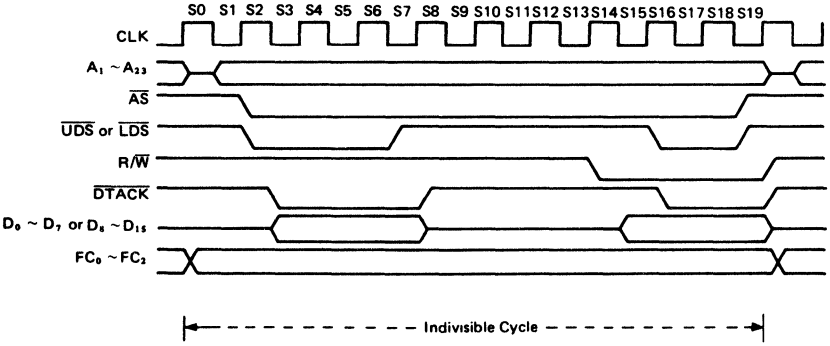
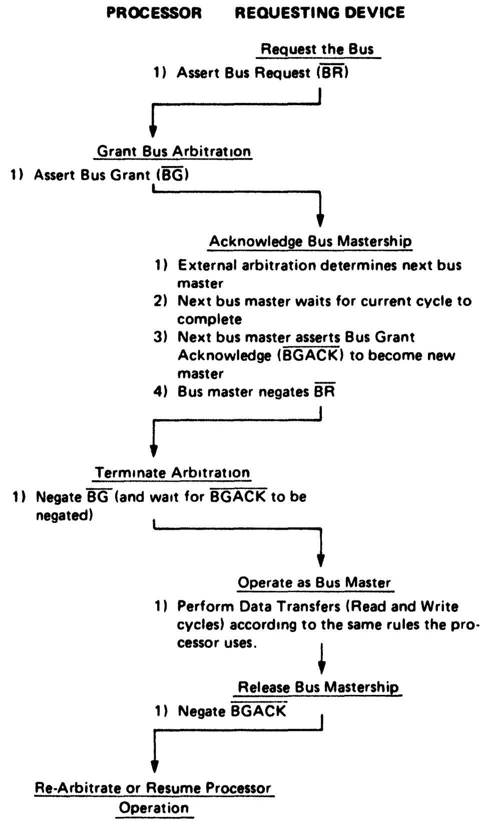

S0 S1 S2 S3 S4 S5 S6 S7 S8 S9 S10 S11 S12 S13 S14 S15 S16 S17 S18 S19
CLK
$A_1 \sim A_{23}$
$\overline{AS}$
$\overline{UDS}$ or $\overline{LDS}$
R/$\overline{W}$
$\overline{DTACK}$
$D_0 \sim D_7$ or $D_8 \sim D_{15}$
$FC_0 \sim FC_2$
Indivisible Cycle

Figure 23 Read-Modify-Write Cycle Timing Diagram



PROCESSOR REQUESTING DEVICE

Request the Bus

1) Assert Bus Request ($\overline{BR}$)

Grant Bus Arbitration

1) Assert Bus Grant ($\overline{BG}$)

Acknowledge Bus Mastership

1) External arbitration determines next bus master
2) Next bus master waits for current cycle to complete
3) Next bus master asserts Bus Grant Acknowledge ($\overline{BGACK}$) to become new master
4) Bus master negates $\overline{BR}$

Terminate Arbitration

1) Negate $\overline{BG}$ (and wait for $\overline{BGACK}$ to be negated)

Operate as Bus Master

1) Perform Data Transfers (Read and Write cycles) according to the same rules the processor uses.

Release Bus Mastership

1) Negate $\overline{BGACK}$

Re-Arbitrate or Resume Processor Operation

Figure 24 Bus Arbitration Cycle Flow Chart

Requesting the Bus

External devices capable of becoming bus masters request the bus by asserting the bus request ($\overline{BR}$) signal. This is a wire ORed signal (although it need not be constructed from open collector devices) ·that indicates to the processor that some external device requires control of the external bus. The processor is effectively at a lower bus priority level that the external device and will relinquish the bus after it has completed the last bus cycle it has started.

When no acknowledge is received before the bus request signal goes inactive, the processor will continue processing when it detects that the bus request is inactive. This allows ordinary processing to continue if the arbitration circuitry responded to noise inadvertently.

Receiving the Bus Grant

The processor asserts bus grant ($\overline{BG}$) as soon as possible. Normally this is immediately after internal synchronization. The only exception to this occurs when the processor has made an internal decision to execute the next bus cycle but has not progressed far enough into the cycle to have asserted the address strobe ($\overline{AS}$) signal. In this case, bus grant will not be asserted until one clock after address strobe is asserted to indicate to external devices that a bus cycle is being executed.

The bus grant signal may be routed through a daisy-chained network or through a specific priority-encoded network. The processor is not affected by the external method of arbitration as long as the protocol is obeyed.

Acknowledgement of Mastership

Upon receiving a bus grant, the requesting device waits until address strobe, data transfer acknowledge, and bus grant acknowledge are negated before issuing its own $\overline{BGACK}$. The negation of the address strobe indicates that the previous master has completed its cycle, the negation of bus grant acknowledge indicates that the previous master has released the bus. (While address strobe is asserted no device is allowed to "break into" a cycle.) The negation of data transfer acknowledge indicates the previous slave has terminated its connection to the previous master. Note that in some applications data

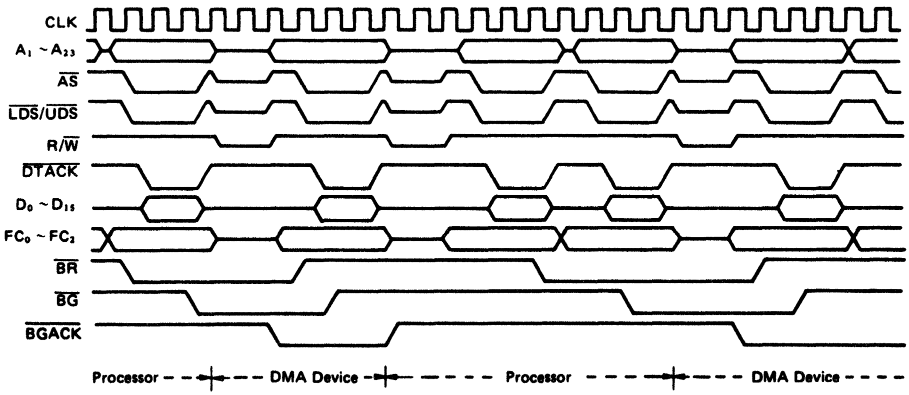


Figure 25 Bus Arbitration Cycle Timing Diagram

transfer acknowledge might not enter into this function. General purpose devices would then be connected such that they were only dependent on address strobe. When bus grant acknowledge is issued the device is bus master until it negates bus grant acknowledge. Bus grant acknowledge should not be negated until after the bus cycle(s) is (are) completed. Bus mastership is terminated at the negation of bus grant acknowledge.

The bus request from the granted device should be dropped after bus grant acknowledge is asserted. If a bus request is still pending, another bus grant will be asserted within a few clocks of the negation of bus grant Refer to Bus Arbitration Control section. Note that the processor does not perform any external bus cycles before it re-asserts bus grant.

BUS ARBITRATION CONTROL

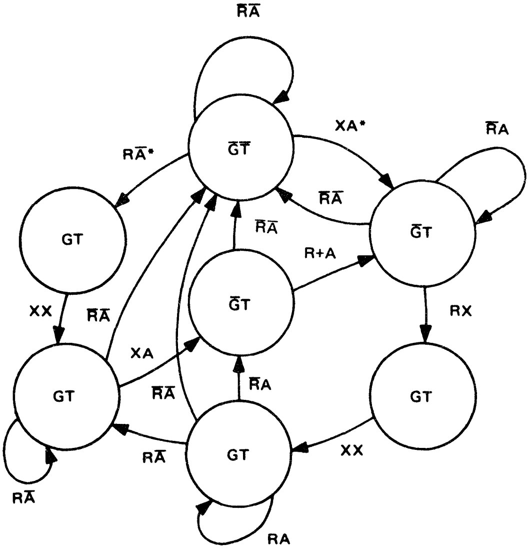
The bus arbitration control unit in the 68000 is implemented with a finite state machine. A state diagram of this machine is shown in Figure 26. All asynchronous signals to the 68000 are synchronized before being used internally. This synchronization is accomplished in a maximum of one cycle of the system clock, assuming that the asynchronous input setup time (#47) has been met (see Figure 27). The input signal is sampled on the falling edge of the clock and is valid internally after the next falling edge.

As shown in Figure 26, input signals labeled R and A are internally synchronized on the bus request and bus grant acknowledge pins respectively. The bus grant output is lebeled G and the internal three-state control signal T. If T is true, the address, data, function code line, and control buses are placed in a high-impedance state when $\overline{\mathrm{AS}}$ is negated. All signals are shown in positive logic (active high) regardless of their true active voltage level.

State changes (valid outputs) occur on the next rising edge after the internal signal is valid.

A timing diagram of the bus arbitration sequence during a processor bus cycle is shown in Figure 28. The bus arbitration sequence while the bus is inactive (i.e., executing internal operations such as a multiply instruction) is shown in Figure 29.

If a bus request is made at a time when the MPU has already begun a bus cycle but $\overline{\mathrm{AS}}$ has not been asserted (bus state S0), $\overline{\mathrm{BG}}$ will not be asserted on the next rising edge. Instead, $\overline{\mathrm{BG}}$ will be delayed until the second rising edge following it's internal assertion. This sequence is shown in Figure 30.


$\overline{R}\overline{A}$
$\overline{G}\overline{T}$
XA*
$\overline{R}A$
$R\overline{A}$*
GT
$\overline{R}\overline{A}$
$\overline{G}T$
$\overline{R}\overline{A}$
R+A
$\overline{G}T$
XX
$\overline{R}\overline{A}$
RX
XA
$\overline{R}\overline{A}$
$\overline{R}A$
GT
GT
XX
$R\overline{A}$
GT
$R\overline{A}$
RA

R = Bus Request Internal
A = Bus Grant Acknowledge Internal
G = Bus Grant
T = Three-State Control to Bus Control Logic**
X = Don't Care

* State machine will not change state if bus is in S0. Refer to BUS ARBITRATION CONTROL for additional information.

** The address bus will be placed in the high impedance state if T is asserted and $\overline{\mathrm{AS}}$ is negated.

Figure 26 State Diagram of 68000 Bus Arbitration Unit

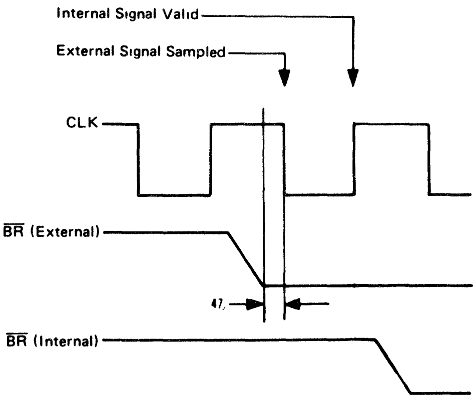

Internal Signal Valid
External Signal Sampled
CLK
$\overline{\mathrm{BR}}$ (External)
47
$\overline{\mathrm{BR}}$ (Internal)

Figure 27 Timing Relationship of External Asynchronous Inputs to Internal Signals

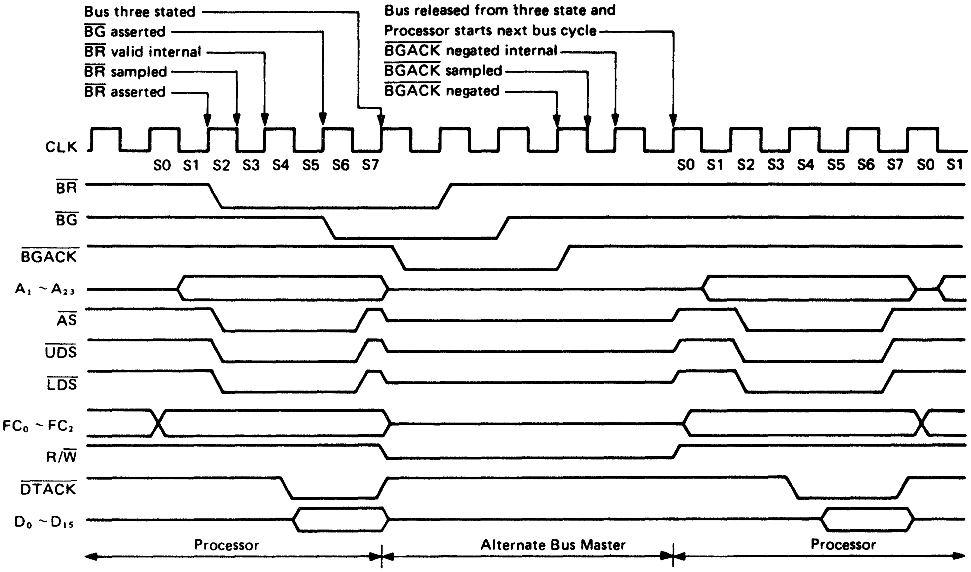


Figure 28 Bus Arbitration During Processor Bus Cycle

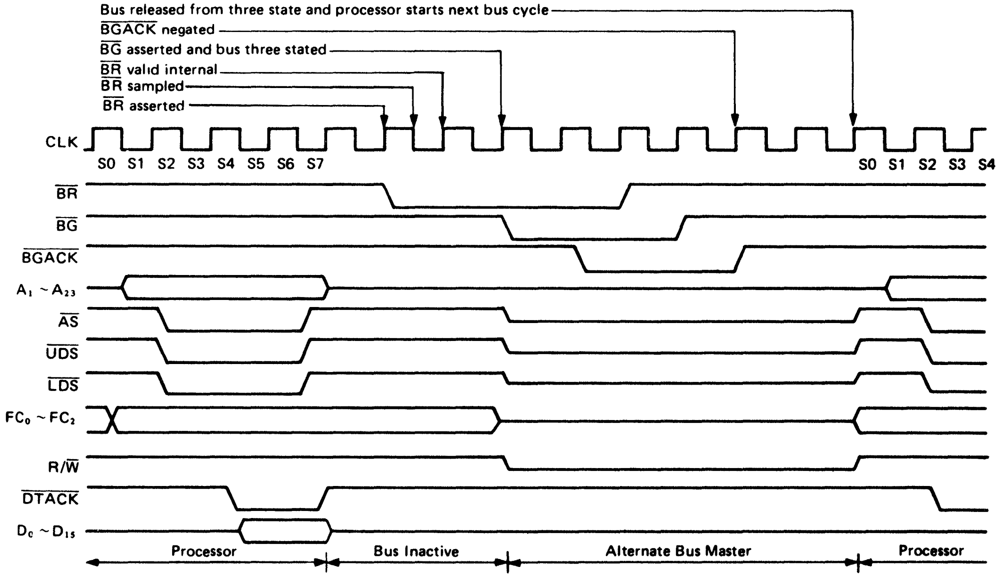


Figure 29 Bus Arbitration with Bus Inactive

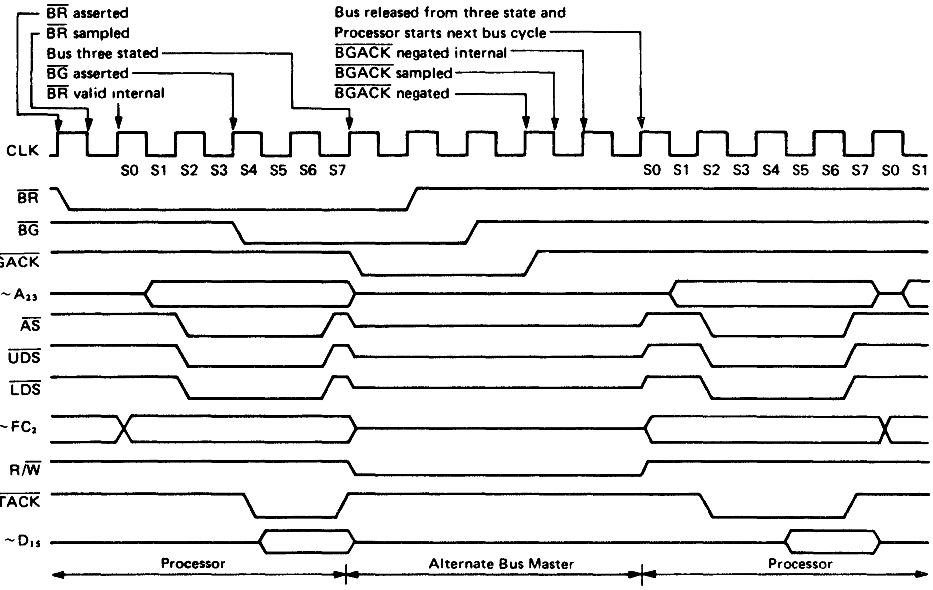


Figure 30 Bus Arbitration During Processor Bus Cycle Special Case

BUS ERROR AND HALT OPERATION

In a bus architecture that requires a handshake from an external device, the possibility exists that the handshake might not occur. Since different systems will require a different maximum response time, a bus error input is provided. External circuitry must be used to determine the duration between address strobe and data transfer acknowledge before issuing a bus error signal. When a bus error signal is received, the processor has two options initiate a bus error exception sequence or try running the bus cycle again.

Exception Sequence

When the bus error signal is asserted, the current bus cycle is terminated. If $\overline{\text{BERR}}$ is asserted before the falling edge of S2, $\overline{\text{AS}}$ will be negated in S7 in either a read or write cycle. As long as $\overline{\text{BERR}}$ remains asserted, the data and address buses will be in the high-impedance state. When $\overline{\text{BERR}}$ is negated, the processor will begin stacking for exception processing. Figure 31 is a timing diagram for the exception sequence. The sequence is composed of the following elements.

(1) Stacking the program counter and status register
(2) Stacking the error information
(3) Reading the bus error vector table entry
(4) Executing the bus error handler routine

The stacking of the program counter and the status register is the same as if an interrupt had occurred Several additional items are stacked when a bus error occurs. These items are used to determine the nature of the error and correct it, if possible. The bus error vector is vector number two located at address $000008. The processor loads the new program counter from this location. A software bus error handler routine is then executed by the processor. Refer to **EXCEPTION PROCESSING** for additional information.

Re-Running the Bus Cycle

When, during a bus cycle, the processor receives a bus error signal and the halt pin is being driven by an external device, the processor enters the re-run sequence. Figure 32 is a timing diagram for re-running the bus cycle.

The processor terminates the bus cycle, then puts the address and data output lines in the high-impedance state. The processor remains "halted," and will not run another bus cycle until the halt signal is removed by external logic. Then the processor will re-run the previous bus cycle using the same address, the same function codes, the same data (for a write operation), and the same controls. The bus error signal should be removed at least one clock cycle before the halt signal is removed.

(NOTE) The processor will not re-run a read-modify-write cycle. This restriction is made to guarantee that the entire cycle runs correctly and that the write operation of a Test-and-Set operation is performed without ever releasing $\overline{\text{AS}}$. If $\overline{\text{BERR}}$ and $\overline{\text{HALT}}$ are asserted during a read-modify-write bus cycle, a bus error operation results.

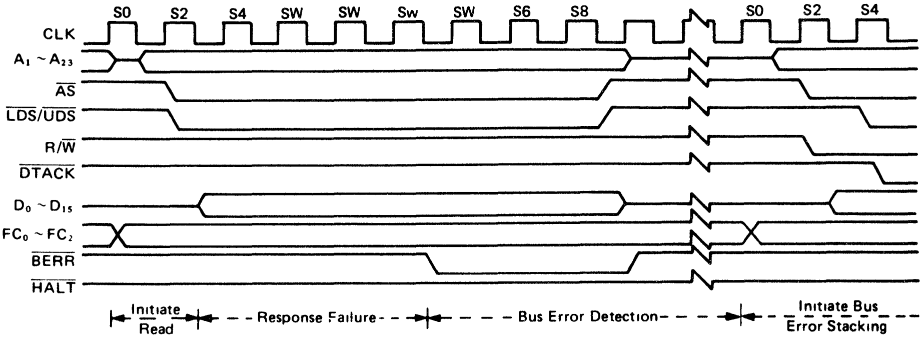


Figure 31 Bus Error Timing Diagram

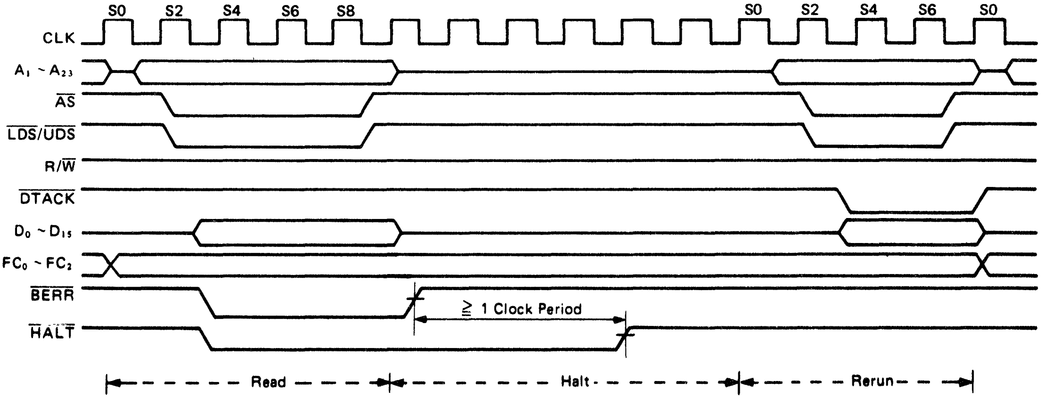


Figure 32 Re-Run Bus Cycle Timing Information

Halt Operation with No Bus Error

The halt input signal to the 68000 perform a Halt/Run/Single-Step function in a similar fashion to the HD6800 halt function. The halt and run modes are somewhat self explanatory in that when the halt signal is constantly active the processor "halts" (does nothing) and when the halt signal is constantly inactive the processor "runs" (does something).

The single-step mode is derived from correctly timed transitions on the halt signal input. It forces the processor to execute a single bus cycle by entering the "run" mode until the processor starts a bus cycle then changing to the "halt" mode. Thus, the single-step mode allows the user to proceed through (and therefore debug) processor operations one bus cycle at a time.

Figure 33 details the timing required for correct single-step operations and Figure 34 shows a simple circuit for providing the single-step function. Some care must be exercised to avoid harmful interactions between the bus error signal and the halt pin when using the single cycle mode as a debugging tool. This is also true of interactions between the halt and reset lines since these can reset the machine.

When the processor completes a bus cycle after recognizing that the halt signal is active, most three-state signals are put in the high-impedance state. These include:

(1) Address lines
(2) Data lines

This is required for correct performance of the re-run bus cycle operation.

While the processor is honoring the halt request, bus arbitration performs as usual. That is, halting has no effect on bus arbitration. It is the bus arbitration function that removes the control signals from the bus.

The halt function and the hardware trace capability allow the hardware debugger to trace single bus cycles or single instructions at a time. These processor capabilities, along with a software debugging package, give total debugging flexibility.

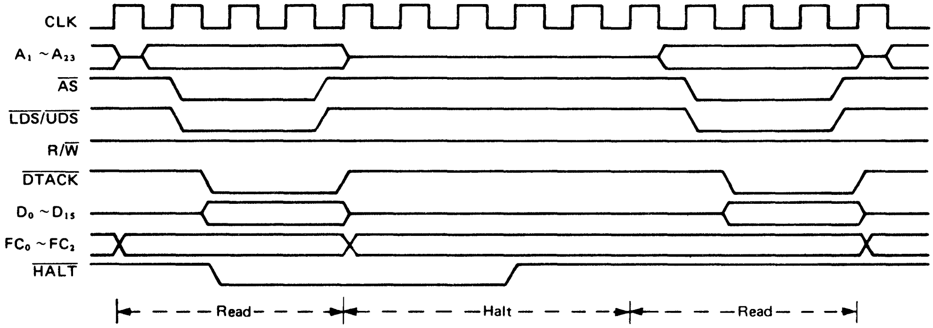


Figure 33 Halt Signal Timing Characteristics

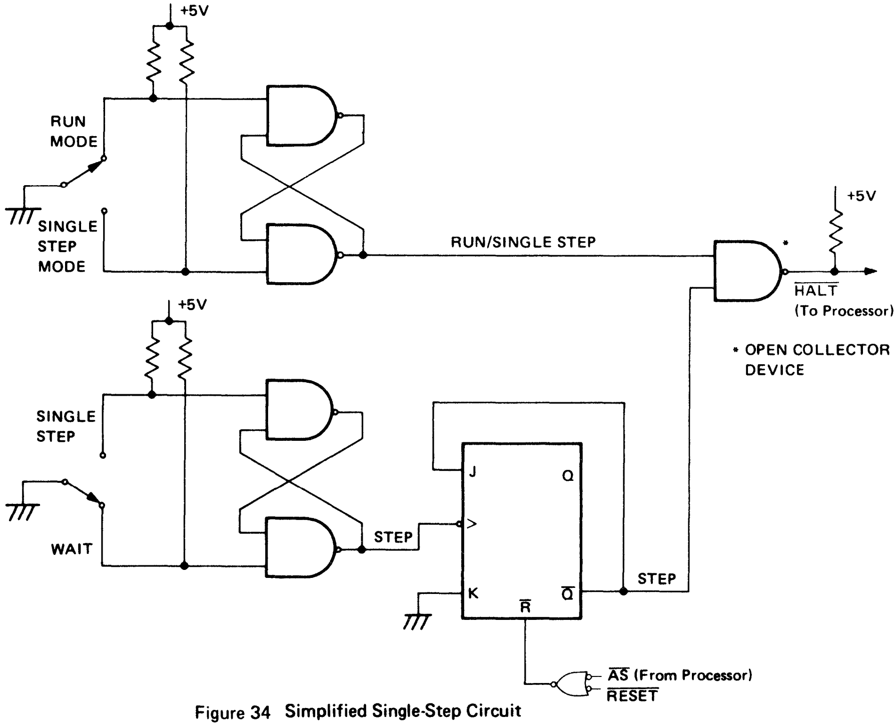


Figure 34 Simplified Single-Step Circuit

Double Bus Faults

When a bus error exception occurs, the processor will attempt to stack several words containing information about the state of the machine. If a bus error exception occurs during the stacking operation, there have been two bus errors in a row. This is commonly referred to as a double bus fault. When a double bus fault occurs, the processor will halt. Once a bus error exception has occurred, any bus error exception occurring before the execution of the next instruction constitutes a double bus fault.

Note that a bus cycle which is re-run does not constitute a bus error exception, and does not contribute to a double bus fault. Note also that this means that as long as the external hardware requests it, the processor will continue to re-run the same bus cycle.

The bus error pin also has an effect on processor operation after the processor receives an external reset input The processor reads the vector table after a reset to determine the address to start program execution. If a bus error occurs while reading the vector table (or at any time before the first instruction is executed), the processor reacts as if a double bus fault has occurred and it halts. Only an external reset will start a halted processor.

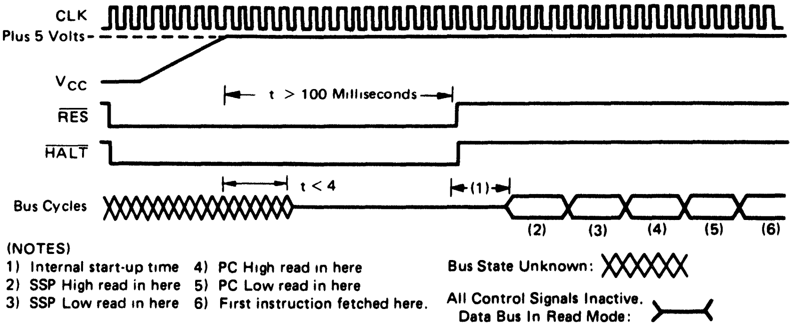
RESET OPERATION

The reset signal is a bidirectional signal that allows either the processor or an external signal to reset the system. Figure 35 is a timing diagram for the reset operations. Both the halt and reset lines must be asserted to ensure total reset of the processor.

When the reset and halt lines are driven by an external device, it is recognized as an entire system reset, including the processor. The processor responds by reading the reset vector table entry (vector unumber zero, address $000000) and loads it into the supervisor stack pointer (SSP). Vector table entry number one at address $000004 is read next and loaded into the program counter. The processor initializes the status register to an interrupt level of seven. No other registers are affected by the reset sequence.

When a RESET instruction is executed, the processor drives the reset pin for 124 clock periods. In this case, the processor is trying to reset the rest of the system. Therefore, there is no effect on the internal state of the processor. All of the processor's internal registers and the status register are unaffected by the execution of a RESET instruction. All external devices connected to the reset line should be reset at the completion of the RESET instruction.

Asserting the Reset and Halt pins for 10 clock cycles will cause a processor reset, except when V_{CC} is initially applied to the processor. In this case, an external reset must be applied for 100 milliseconds.



(NOTES)
1) Internal start-up time
2) SSP High read in here
3) SSP Low read in here
4) PC High read in here
5) PC Low read in here
6) First instruction fetched here.

Figure 35 Reset Operation Timing Diagram

THE RELATIONSHIP OF $\overline{\mathrm{DTACK}}$, $\overline{\mathrm{BERR}}$, AND $\overline{\mathrm{HALT}}$

In order to properly control termination of a bus cycle for a re-run or a bus error condition, $\overline{\mathrm{DTACK}}$, $\overline{\mathrm{BERR}}$, and $\overline{\mathrm{HALT}}$ should be asserted and negated on the rising edge of the 68000 clock. This will assure that when two signals are asserted simultaneously, the required setup time (#47) for both of them will be met during the same bus state.

This, or some equivalent precaution, should be designed external to the 68000. Parameter #48 is intended to ensure this operation in a totally asynchronous system, and may be ignored if the above conditions are met.

The preferred bus cycle terminations may be summarized as follows (case numbers refer to Table 16):

Normal Termination:	$\overline{\mathrm{DTACK}}$ occurs first (case 1).
Halt Termination:	$\overline{\mathrm{HALT}}$ is asserted at the same time or before $\overline{\mathrm{DTACK}}$ and $\overline{\mathrm{BERR}}$ remains negated (cases 2 and 3).
Bus Error Termination:	$\overline{\mathrm{BERR}}$ is asserted in lieu of, at the same time, or before $\overline{\mathrm{DTACK}}$ (case 4); $\overline{\mathrm{BERR}}$ is negated at the same time or after $\overline{\mathrm{DTACK}}$.
Re-Run Termination:	$\overline{\mathrm{HALT}}$ and $\overline{\mathrm{BERR}}$ are asserted in lieu of, at the same time, or before $\overline{\mathrm{DTACK}}$ (cases 6 and 7); $\overline{\mathrm{HALT}}$ must be held at least one cycle after $\overline{\mathrm{BERR}}$. Case 5 indicates $\overline{\mathrm{BERR}}$ may precede $\overline{\mathrm{HALT}}$ which allows fully asynchronous assertion.

Table 16 details the resulting bus cycle termination under various combinations of control signal sequences. The negation of these same control signals under several conditions is shown in Table 17 ($\overline{\mathrm{DTACK}}$ is assumed to be negated normally in all cases; for best results, both $\overline{\mathrm{DTACK}}$ and $\overline{\mathrm{BERR}}$ should be negated when address strobe is negated.)

Example A: A system uses a watch-dog timer to terminate accesses to un-populated address space. The timer asserts $\overline{\mathrm{DTACK}}$ and $\overline{\mathrm{BERR}}$ simultaneously after time-out. (case 4)

Example B: A system uses error detection on RAM contents. Designer may (a) delay $\overline{\mathrm{DTACK}}$ until data verified, and return $\overline{\mathrm{BERR}}$ and $\overline{\mathrm{HALT}}$ simultaneously to re-run error cycle (case 6), or if valid, return $\overline{\mathrm{DTACK}}$; (b) delay $\overline{\mathrm{DTACK}}$ until data verified, and return $\overline{\mathrm{BERR}}$ at same time as $\overline{\mathrm{DTACK}}$ if data in error (case 4); (c) return $\overline{\mathrm{DTACK}}$ prior to data verification, as described in previous section. If data invalid, $\overline{\mathrm{BERR}}$ is asserted (case 1) in next cycle. Error-handling software must know how to recover error cycle.

Table 16 $\overline{\text{DTACK}}$, $\overline{\text{BERR}}$, $\overline{\text{HALT}}$ Assertion Results

Case No.	Control Signal	Asserted on Rising Edge of State		Result
		N	N + 2	
1	$\overline{\text{DTACK}}$ $\overline{\text{BERR}}$ $\overline{\text{HALT}}$	A NA NA	S X X	Normal cycle terminate and continue.
2	$\overline{\text{DTACK}}$ $\overline{\text{BERR}}$ $\overline{\text{HALT}}$	A NA A	S X S	Normal cycle terminate and halt. Continue when $\overline{\text{HALT}}$ removed.
3	$\overline{\text{DTACK}}$ $\overline{\text{BERR}}$ $\overline{\text{HALT}}$	NA NA A	A NA S	Normal cycle terminate and halt. Continue when $\overline{\text{HALT}}$ removed.
4	$\overline{\text{DTACK}}$ $\overline{\text{BERR}}$ $\overline{\text{HALT}}$	X A NA	X S NA	Terminate and take bus error trap.
5	$\overline{\text{DTACK}}$ $\overline{\text{BERR}}$ $\overline{\text{HALT}}$	NA A NA	X S A	Terminate and re-run.
6	$\overline{\text{DTACK}}$ $\overline{\text{BERR}}$ $\overline{\text{HALT}}$	X A A	X S S	Terminate and re-run when $\overline{\text{HALT}}$ removed.
7	$\overline{\text{DTACK}}$ $\overline{\text{BERR}}$ $\overline{\text{HALT}}$	NA NA A	X A S	Terminate and re-run when $\overline{\text{HALT}}$ removed.

gend.

N	— The number of the current even bus state (e g , S4, S6, etc)
A	— Signal is asserted in this bus state
NA	— Signal is not asserted in this state
X	— Don't care
S	— Signal was asserted in previous state and remains asserted in this state

Table 17 $\overline{\text{BERR}}$ and $\overline{\text{HALT}}$ Negation Results

Conditions of Termination in Table A	Control Signal	Negated on Rising Edge of State		Results — Next Cycle
		N	N + 2	
Bus Error	$\overline{\text{BERR}}$ $\overline{\text{HALT}}$	● or ● or	● ●	Takes bus error trap.
Re-run	$\overline{\text{BERR}}$ $\overline{\text{HALT}}$	● or ●	●	Illegal sequence; usually traps to vector number 0.
Re-run	$\overline{\text{BERR}}$ $\overline{\text{HALT}}$	●	 ●	Re-runs the bus cycle.
Normal	$\overline{\text{BERR}}$ $\overline{\text{HALT}}$	● ● or	 ●	May lengthen next cycle.
Normal	$\overline{\text{BERR}}$ $\overline{\text{HALT}}$	 ● or	● none	If next cycle is started it will be terminated as a bus error.

ASYNCHRONOUS VERSUS SYNCHRONOUS OPERATION

Asynchronous Operation

To achieve clock frequency independence at a system level, the 68000 can be used in an asynchronous manner. This entails using only the bus handshake lines ($\overline{\text{AS}}$, $\overline{\text{UDS}}$, $\overline{\text{LDS}}$, $\overline{\text{DTACK}}$, $\overline{\text{BERR}}$, $\overline{\text{HALT}}$, and $\overline{\text{VPA}}$) to control the data transfer. Using this method, $\overline{\text{AS}}$ signals the start of a bus cycle and the data strobes are used as a condition for valid data on a write cycle. The slave device (memory or peripheral) then responds by placing the requested data on the data bus for a read cycle or latching data on a write cycle and asserting the data transfer acknowledge signal ($\overline{\text{DTACK}}$) to terminate the bus cycle. If no slave responds or the access is invalid, external control logic asserts the $\overline{\text{BERR}}$, or $\overline{\text{BERR}}$ and $\overline{\text{HALT}}$, signal to abort or re-run the bus cycle.

The $\overline{\text{DTACK}}$ signal is allowed to be asserted before the data from a slave device is valid on a read cycle. The length of time that $\overline{\text{DTACK}}$ may precede data is given as parameter #31 and it must be met in any asynchronous system to insure that valid data is latched into the processor. Notice that there is no maximum time specified from the assertion of $\overline{\text{AS}}$ to the assertion of $\overline{\text{DTACK}}$. This is because the MPU will insert wait cycles of one clock period each until $\overline{\text{DTACK}}$ is recognized.

The $\overline{\text{BERR}}$ signal is allowed to be asserted after the $\overline{\text{DTACK}}$ signal is asserted. $\overline{\text{BERR}}$ must be asserted within the time given as parameter #48 after $\overline{\text{DTACK}}$ is asserted in any asynchronous system to insure proper operation. If this maximum delay time is violated, the processor may exhibit erratic behavior.

Synchronous Operation

To allow for those systems which use the system clock as a signal to generate $\overline{\text{DTACK}}$ and other asynchronous inputs, the asynchronous input setup time is given as parameter #47. If this setup is met on an input, such as $\overline{\text{DTACK}}$, the processor is guaranteed to recognize that signal on the next falling edge of the system clock. However, the converse is not true – if the input signal does not meet the setup time it is not guaranteed not to be recognized In addition, if $\overline{\text{DTACK}}$ is recognized on a falling edge, valid data will be latched into the processor (on a read cycle) on the next falling edge provided that the data meets the setup time given as parameter #27. Given this, parameter #31 may be ignored. Note that if $\overline{\text{DTACK}}$ is asserted, with the required setup time, before the falling edge of S4, no wait status will be incurred and the bus cycle will run at its maximum speed of four clock periods.

In order to assure proper operation in a synchronous system when $\overline{\text{BERR}}$ is asserted after $\overline{\text{DTACK}}$, $\overline{\text{BERR}}$ must meet the setup time parameter #27A prior to the falling edge of the clock one clock cycle after $\overline{\text{DTACK}}$ was recognized This setup time is critical to proper operation, and the HD68000 may exhibit erratic behavior if it is violated

(NOTE)

During an active bus cycle, $\overline{\text{VPA}}$ and $\overline{\text{BERR}}$ are sampled on every falling edge of the clock starting with S0 $\overline{\text{DTACK}}$ is sampled on every falling edge of the clock starting with S4 and data is latched on the falling edge of S6 during a read The bus cycle will then be terminated in S7 except when $\overline{\text{BERR}}$ is asserted in the absence of $\overline{\text{DTACK}}$, in which case it will terminate one clock cycle later in S9

■ PROCESSING STATES

This section describes the actions the 68000 which are outside the normal processing associated with the execution of instructions. The functions of the bits in the supervisor portion of the status register are covered: the supervisor/user bit, the trace enable bit, and the processor interrupt priority mask. Finally, the sequence of memory references and actions taken by the processor on exception conditions is detailed.

The 68000 is always in one of three processing states: normal, exception, or halted The normal processing state is that associated with instruction execution, the memory references are to fetch instructions and operands, and to store results A special case of the normal state is the stopped state which the processor enters when a STOP instruction is executed In this state, no further memory references are made

The exception processing state is associated with interrupts, trap instructions, tracing and other exceptional conditions The exception may be internally generated by an instruction or by an unusual condition arising during the execution of an instruction Externally, exception processing can be forced by an interrupt, by a bus error, or by a reset Exception processing is designed to provide an efficient context switch so that the processor may handle unusual conditions

The halted processing state is an indication of catastrophic hardware failure For example, if during the exception processing of a bus error another bus error occurs, the processor assumes that the system is unusable and halts Only an external reset can restart a halted processor. Note that a processor in the stopped state is not in the halted state, nor vice versa.

PROCESSING STATES

NORMAL	INSTRUCTION EXECUTION (INCLUDING STOP)
EXCEPTION	INTERRUPTS TRAPS TRACING ETC.
HALTED	HARDWARE HALT DOUBLE BUS FAULT

● PRIVILEGE STATES

The processor operates in one of two states of privilege the "user" state or the "supervisor" state. The privilege state determines which operations are legal, are used to choose between the supervisor stack pointer and the user stack pointer in instruction references, and may be used by an external memory management device to control and translate accesses.

The privileges state is a mechanism for providing security in a computer system Programs should access only their own code and data areas, and ought to be restricted from accessing information which they do not need and must not modify.

The privilege mechanism provides security by allowing most programs to execute in user state. In this state, the accesses are controlled, and the effects on other parts of the system are limited. The operating system executes in the supervisor state, has access to all resources, and performs the overhead tasks for the user state programs

SUPERVISOR STATE

The supervisor state is the higher state of privilege. For instruction execution, the supervisor state is determined by the S-bit of the status register, if the S-bit is asserted (high), the processor is in the supervisor state. All instructions can be executed in the supervisor state. The bus cycles generated by instructions executed in the supervisor state are classified as supervisor references. While the processor is in the supervisor privilege state, those instructions which use either the system stack pointer implicitly or address register seven explicitly access the supervisor stack pointer.

All exception processing is done in the supervisor state, regardless of the setting of the S-bit. The bus cycles generated during exception processing are classified as supervisor references. All stacking operations during exception processing use the supervisor stack pointer.

USER STATE

The user state is the lower state of privilege. For instruction execution, the user state is determined by the S-bit of the status register; if the S-bit is negated (low), the processor is executing instructions in the user state.

Most instructions execute the same in user state as in the supervisor state. However, some instructions which have important system effects are made privileged. User programs are not permitted to execute the STOP instruction, or the

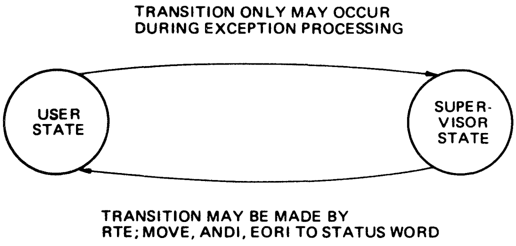
RESET instruction. To ensure that a user program cannot enter the supervisor state except in a controlled manner, the instructions which modify the whole status register are privileged. To aid in debugging programs which are to be used as operating systems, the move to user stack pointer (MOVE to USP) and move from user stack pointer (MOVE from USP) instructions are also privileged.

The bus cycles generated by an instruction executed in user state are classified as user state references. This allows an external memory management device to translate the address and to control access to protected portions of the address space. While the processor is in the user privilege state, those instructions which use either the system stack pointer implicitly, or address register seven explicitly, access the use stack pointer.

PRIVILEGE STATE CHANGES

Once the processor is in the user state and executing instructions, only exception processing can change the privilege state. During exception processing, the current setting of the S-bit of the status register is saved and the S-bit is asserted, putting the processing in the supervisor state. Therefore, when instruction execution resumes at the address specified to process the exception, the processor is in the supervisor privilege state.

USER/SUPERVISOR MODES


TRANSITION ONLY MAY OCCUR
DURING EXCEPTION PROCESSING
USER STATE
SUPER-VISOR STATE
TRANSITION MAY BE MADE BY
RTE; MOVE, ANDI, EORI TO STATUS WORD

REFERENCE CLASSIFICATION

When the processor makes a reference, it classifies the kind of reference being made, using the encoding on the three function code output lines. This allows external translation of addresses, control of access, and differentiation of special processor states, such as interrupt acknowledge. Table 18 lists the classification of references.

Table 18 Reference Classification

Function Code Output			Reference Class
FC_2	FC_1	FC_0	
0	0	0	(Unassigned)
0	0	1	User Data
0	1	0	User Program
0	1	1	(Unassigned)
1	0	0	(Unassigned)
1	0	1	Supervisor Data
1	1	0	Supervisor Program
1	1	1	Interrupt Acknowledge

• EXCEPTION PROCESSING

Before discussing the details of interrupts, traps, and tracing, a general description of exception processing is in order. The processing of an exception occurs in four steps, with variations for different exception causes. During the first step, a temporary copy of the status register is made, and the status register is set for exception processing. In the second step the exception vector is determined, and the third step is the saving of the current processor context. In the fourth step a new context is obtained, and the processor switches to instruction processing.

EXCEPTION VECTORS

Exception vectors are memory locations from which the processor fetches the address of a routine which will handle that exception. All exception vectors are two words in length (Figure 36), except for the reset vector, which is four words. All exception vectors lie in the supervisor data space, except for the reset vector which is in the supervisor program space. A vector number is an eight-bit number which, when multiplied by four, gives the address of an exception vector. Vector numbers are generated internally or externally depending on the cause of the exception. In the case of interrupts, during the interrupt acknowledge bus cycle, a peripheral provides an 8-bit vector number (Figure 37) to the processor on data bus lines D_0 through D_7. The processor translates the vector number into a full 24-bit address, as shown in Figure 38. The memory layout for exception vectors is given in Table 19.

As shown in Table 19, the memory layout is 512 words long (1024 bytes). It starts at address 0 and proceeds through address 1023. This provides 255 unique vectors; some of these are reserved for TRAPS and other system functions Of the 255, there are 192 reserved for user interrupt vectors However, there is no protection on the first 64 entries, so user interrupt vectors may overlap at the discretion of the systems designer.

KINDS OF EXCEPTIONS

Exceptions can be generated by either internal or external causes. The externally generated exceptions are the interrupts and the bus error and reset requests. The interrupts are requests from peripheral devices for processor action while the bus error and reset inputs are used for access control and processor restart. The internally generated exceptions come from instructions, or from address error or tracing. The trap (TRAP), trap on overflow (TRAPV), check register against bounds (CHK) and divide (DIV) instructions all can generate exceptions as part of their instruction execution. In addition, illegal instructions, word fetches from odd addresses and privilege violations cause exceptions. Tracing behaves like a very high priority, internally generated interrupt after each instruction execution.

EXCEPTION PROCESSING SEQUENCE

Exception processing occurs in four identifiable steps. In the first step, an internal copy is made of the status register. After the copy is made, the S-bit is asserted, putting the processor into the supervisor privilege state. Also, the T-bit is negated which will allow the exception handler to execute unhindered by tracing. For the reset and interrupt exceptions, the interrupt priority mask is also updated.

In the second step, the vector number of the exception is determined. For interrupts, the vector number is obtained by a processor fetch, classified as an interrupt acknowledge. For all other exceptions, internal logic provides the vector number. This vector number is then used to generate the address of the exception vector.

Word 0	New Program Counter (High)	A0=0, A1=0
Word 1	New Program Counter (Low)	A0=0, A1=1

Figure 36 Exception Vector Format

D15 D8	D7							D0
Ignored	v7	v6	v5	v4	v3	v2	v1	v0

Where
v7 is the MSB of the Vector Number
v0 is the LSB of the Vector Number

Figure 37 Peripheral Vector Number Format

A23 A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
All Zeroes	v7	v6	v5	v4	v3	v2	v1	v0	0	0

Figure 38 Address Translated From 8-Bit Vector Number

Table 19 Exception Vector Assignment

Vector Number(s)	Address			Assignment
	Dec	Hex	Space	
0	0	000	SP	Reset Initial SSP
–	4	004	SP	Reset Initial PC
2	8	008	SD	Bus Error
3	12	00C	SD	Address Error
4	16	010	SD	Illegal Instruction
5	20	014	SD	Zero Divide
6	24	018	SD	CHK Instruction
7	28	01C	SD	TRAPV Instruction
8	32	020	SD	Privilege Violation
9	36	024	SD	Trace
10	40	028	SD	Line 1010 Emulator
11	44	02C	SD	Line 1111 Emulator
12*	48	030	SD	(Unassigned, reserved)
13*	52	034	SD	(Unassigned, reserved)
14*	56	038	SD	(Unassigned, reserved)
15	60	03C	SD	Uninitialized Interrupt Vector
16 ~ 23*	64	040	SD	(Unassigned, reserved)
	95	05F		
24	96	060	SD	Spurious Interrupt
25	100	064	SD	Level 1 Interrupt Autovector
26	104	068	SD	Level 2 Interrupt Autovector
27	108	06C	SD	Level 3 Interrupt Autovector
28	112	070	SD	Level 4 Interrupt Autovector
29	116	074	SD	Level 5 Interrupt Autovector
30	120	078	SD	Level 6 Interrupt Autovector
31	124	07C	SD	Level 7 Interrupt Autovector
32 ~ 47	128	080	SD	TRAP Instruction Vectors
	191	0BF		
48 ~ 63*	192	0C0	SD	(Unassigned, reserved)
	255	0FF		
64 ~ 255	256	100	SD	User Interrupt Vectors
	1023	3FF		

SP Supervisor program, SD Supervisor data

* Vector numbers 12, 13, 14, 16 through 23 and 48 through 63 are reserved for future enhancements by Hitachi No user peripheral devices should be assigned these numbers

The third step is to save the current processor status, except for the reset exception. The current program counter value and the saved copy of the status register are stacked using the supervisor stack pointer as shown in Figure 39. The program counter value stacked usually points to the next unexecuted instruction, however for bus error and address error, the value stacked for the program counter is unpredictable, and may be incremented from the address of the instruction which caused the error. Additional information defining the current context is stacked for the bus error and address error exceptions.

The last step is the same for all exceptions. The new program counter value is fetched from the exception vector. The processor then resumes instruction execution. Then instruction at the address given in the exception vector is fetched, and normal instruction decoding and execution is started.

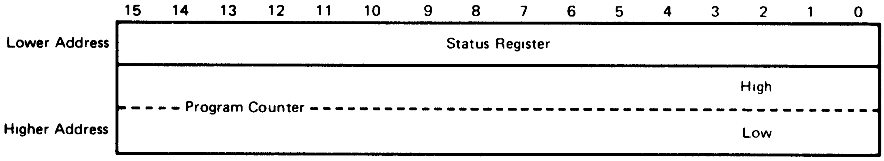


Figure 39 Exception Stack Order (Group 1, 2)

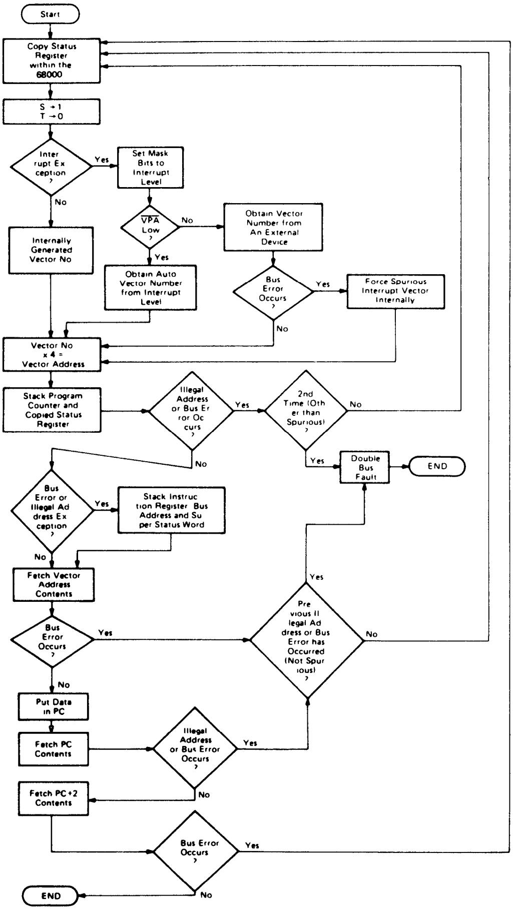

Start
Copy Status Register within the 68000
S → 1
T → 0
Inter rupt Ex ception ?
Yes
Set Mask Bits to Interrupt Level
No
VPA Low ?
No
Obtain Vector Number from An External Device
Internally Generated Vector No
Yes
Obtain Auto Vector Number from Interrupt Level
Bus Error Occurs ?
Yes
Force Spurious Interrupt Vector Internally
No
Vector No x 4 = Vector Address
Stack Program Counter and Copied Status Register
Illegal Address or Bus Er ror Oc curs ?
Yes
2nd Time (Oth er than Spurious) ?
No
Yes
Double Bus Fault
END
No
Bus Error or Illegal Ad dress Ex ception ?
Yes
Stack Instruc tion Register Bus Address and Su per Status Word
No
Fetch Vector Address Contents
Yes
Bus Error Occurs ?
Yes
Pre vious Il legal Ad dress or Bus Error has Occurred (Not Spur ious) ?
No
No
Put Data in PC
Fetch PC Contents
Illegal Address or Bus Error Occurs ?
Yes
Fetch PC+2 Contents
No
Bus Error Occurs ?
Yes
END
No

Figure 40 Exception Processing Sequence (Not Reset)

MULTIPLE EXCEPTIONS

These paragraphs describe the processing which occurs when multiple exceptions arise simultaneously. Exceptions can be grouped according to their occurrence and priority. The Group 0 exceptions are reset, bus error, and address error. These exceptions cause the instruction currently being executed to be aborted, and the exeception processing to commence within two clock cycles. The Group 1 exceptions are trace and interrupt, as well as the privilege violations and illegal instructions. These exceptions allow the current instruction to execute to completion, but preempt the execution of the next instruction by forcing exception processing to occur (privilege violations and illegal instructions are detected when they are the next instruction to be executed). The Group 2 exceptions occur as part of the normal processing of instructions. The TRAP, TRAPV, CHK, and zero divide exceptions are in this group. For these exceptions, the normal execution of an instruction may lead to exception processing

Group 0 exceptions have highest priority, while Group 2 exceptions have lowest priority. Within Group 0, reset has highest priority, followed by address error and then bus error. Within Group 1, trace has priority over external interrupts, which in turn takes priority over illegal instruction and privilege violation. Since only one instruction can be executed at a time, there is no priority relation within Group 2.

The priority relation between two exceptions determines which is taken, or taken first, if the conditions for both arise simultaneously. Therefore, if a bus error occurs during a TRAP instruction, the bus error takes precedence, and the TRAP instruction processing is aborted. In another example, if an interrupt request occurs during the execution of an instruction while the T-bit is asserted, the trace exception has priority, and is processed first. Before instruction processing resumes, however, the interrupt exception is also processed, and instruction processing commences finally in the interrupt handler routine. A summary of exception grouping and priority is given in Table 20.

Table 20 Exception Grouping and Priority

Group	Exception	Processing
0	Reset Address Error Bus Error	Exception processing begins within two clock cycles.
1	Trace Interrupt Illegal Privilege	Exception processing begins before the next instruction
2	TRAP, TRAPV CHK, Zero Divide	Exception processing is started by normal instruction execution

RECOGNITION TIMES OF EXCEPTIONS, HALT, AND BUS ARBITRATION

- END OF A CLOCK CYCLE
  - RESET
- END OF A BUS CYCLE
  - ADDRESS ERROR
  - BUS ERROR
  - HALT
  - BUS ARBITRATION
- END OF AN INSTRUCTION CYCLE
  - TRACE EXCEPTION
  - INTERRUPT EXCEPTIONS
  - ILLEGAL INSTRUCTION
  - UNIMPLEMENTED INSTRUCTION
  - PRIVILEGE VIOLATION
- WITHIN AN INSTRUCTION CYCLE
  - TRAP, TRAPV
  - CHK
  - ZERO DIVIDE

• EXCEPTION PROCESSING DETAILED DISCUSSION

Exceptions have a number of sources, and each exception has processing which is peculiar to it. The following paragraphs detail the sources of exceptions, how each arises, and how each is processed.

RESET

The reset input provides the highest exception level. The processing of the reset signal is designed for system initiation, and recovery from catastrophic failure. Any processing in progress at the time of the reset is aborted and cannot be recovered. The processor is forced into the supervisor state, and the trace state is forced off. The processor interrupt priority mask is set at level seven. The vector number is internally generated to reference the reset exception vector at location 0 in the supervisor program space. Because no assumptions can be made about the validity of register contents, in particular the supervisor stack pointer, neither the program counter nor the status register is saved. The address contained in the first two words of the reset exception vector is fetched as the initial supervisor stack pointer, and the address in the last two words of the reset exception vector is fetched as the initial program counter. Finally, instruction execution is started at the address in the program counter. The power-up/restart code should be pointed to by the initial program counter.

The RESET instruction does not cause loading of the reset vector, but does assert the reset line to reset external devices. This allows the software to reset the system to a known state and then continue processing with the next instruction.

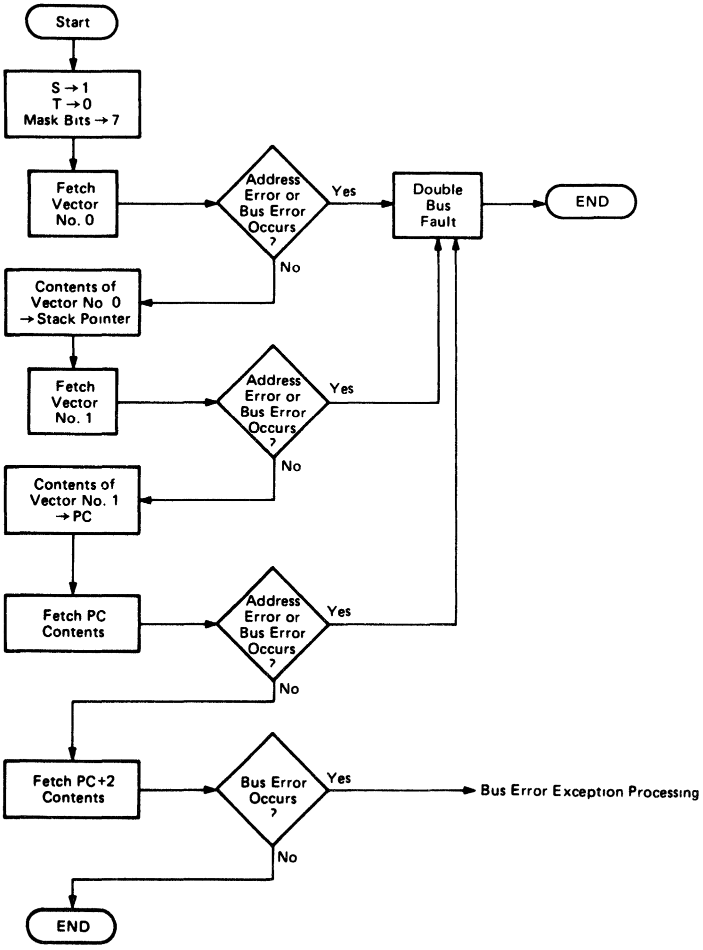

Start
S → 1
T → 0
Mask Bits → 7
Fetch Vector No. 0
Address Error or Bus Error Occurs ?
Yes
No
Double Bus Fault
END
Contents of Vector No 0 → Stack Pointer
Fetch Vector No. 1
Address Error or Bus Error Occurs ?
Yes
No
Contents of Vector No. 1 → PC
Fetch PC Contents
Address Error or Bus Error Occurs ?
Yes
No
Fetch PC+2 Contents
Bus Error Occurs ?
Yes
Bus Error Exception Processing
No
END

Figure 41 Reset Exception Processing

INTERRUPTS

Seven levels of interrupt priorities are provided. Devices may be chained externally within interrupt priority levels, allowing an unlimited number of peripheral devices to interrupt the processor. Interrupt priority levels are numbered from one to seven, with level seven being the highest priority. The status register contains a three-bit mask which indicates the current processor priority, and interrupts are inhibited for all priority levels less than or equal to the current processor priority.

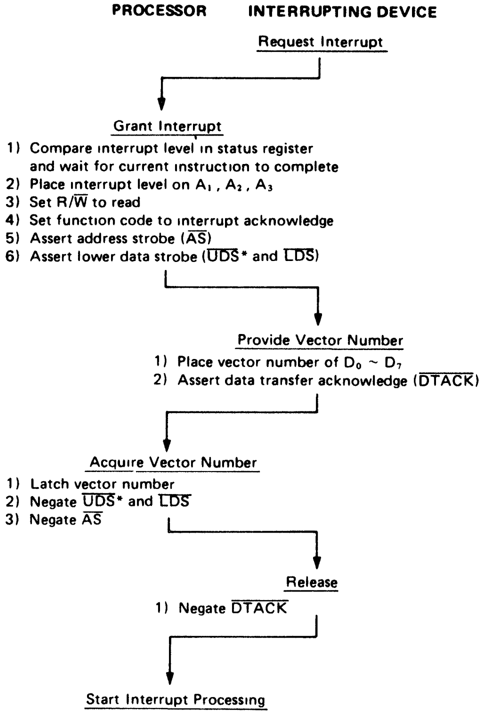
An interrupt request is made to the processor by encoding the interrupt request level on the interrupt request lines; a zero indicates no interrupt request. Interrupt requests arriving at the processor do not force immediate exception processing, but are made pending. Pending interrupts are detected between instruction executions. If the priority of the pending interrupt is lower than or equal to the current processor priority, execution continues with the next instruction and the interrupt exception processing is postponed. (The recognition of level seven is slightly different, as explained in a following paragraph.)

If the priority of the pending interrupt is greater than the current processor priority, the exception processing sequence is started. First a copy of the status register is saved, and the privilege state is set to supervisor, tracing is suppressed, and the processor priority level is set to the level of the interrupt being acknowledged. The processor fetches the vector number from the interrupting device, classifying the reference as an interrupt acknowledge and displaying the level number of the interrupt being acknowledged on the address bus. If external logic requests an automatic vectoring, the processor internally generates a vector number which is determined by the interrupt level number. If external logic indicates a bus error, the 'interrupt is taken to be spurious, and the generated vector number references the spurious interrupt vector. The processor then proceeds with the usual exception processing, saving the program counter and status register on the supervisor stack. The saved value of the program counter is the address of the instruction which would have been executed had the interrupt not been present. The content of the interrupt vector whose vector number was previously obtained is fetched and loaded into the program counter, and normal instruction execution commences in the interrupt handling routine. A flow chart for the interrupt acknowledge sequence is given in Figure 42, a timing diagram is given in Figure 43, and the interrupt exception timing sequence is shown in Figure 44.

Table 21 Internal Interrupt Level

Level	I2	I1	I0	Interrupt
7	1	1	1	Non-Maskable Interrupt
6	1	1	0	Maskable Interrupt
5	1	0	1	
4	1	0	0	
3	0	1	1	
2	0	1	0	
1	0	0	1	
0	0	0	0	No Interrupt

(NOTE) The internal interrupt mask level (I2, I1, I0) are inverted to the logic level applied to the pins ($\overline{IPL_2}$, $\overline{IPL_1}$, $\overline{IPL_0}$)



PROCESSOR **INTERRUPTING DEVICE**

Request Interrupt

Grant Interrupt

1) Compare interrupt level in status register and wait for current instruction to complete
2) Place interrupt level on A_1, A_2, A_3
3) Set R/$\overline{W}$ to read
4) Set function code to interrupt acknowledge
5) Assert address strobe ($\overline{AS}$)
6) Assert lower data strobe ($\overline{UDS}$* and $\overline{LDS}$)

Provide Vector Number

1) Place vector number of D_0 ~ D_7
2) Assert data transfer acknowledge ($\overline{DTACK}$)

Acquire Vector Number

1) Latch vector number
2) Negate $\overline{UDS}$* and $\overline{LDS}$
3) Negate $\overline{AS}$

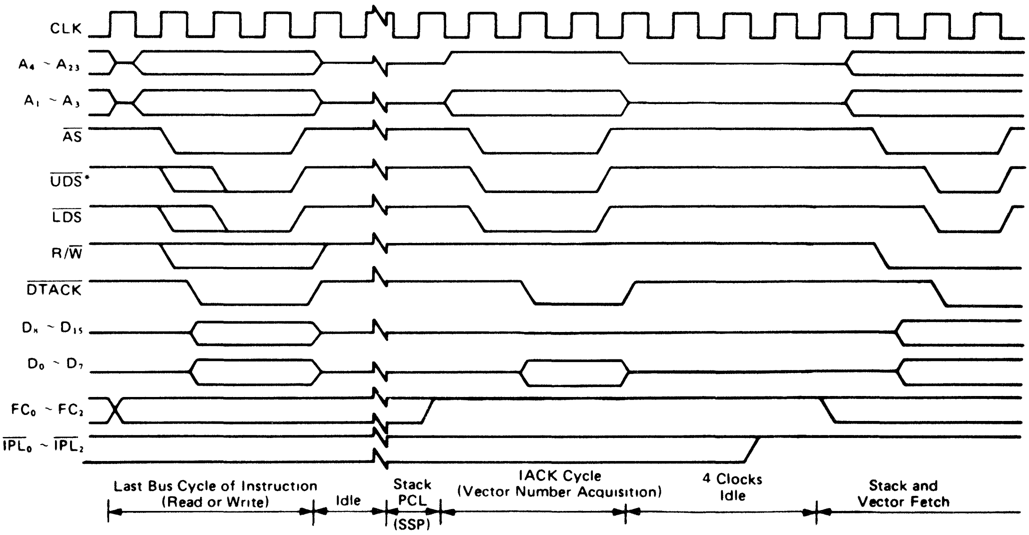
Release

1) Negate $\overline{DTACK}$

Start Interrupt Processing

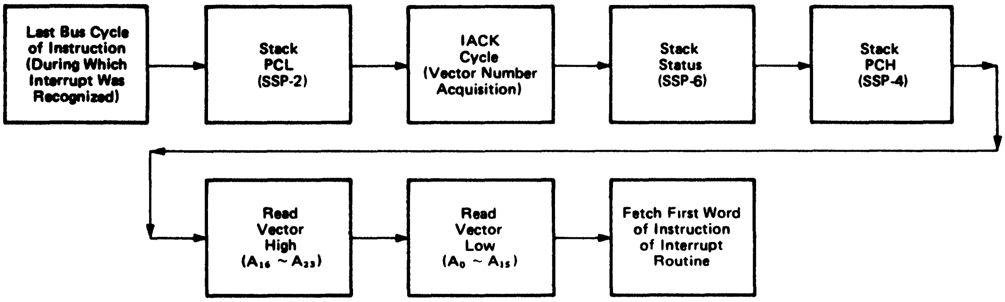
* Although a vector number is one byte, both data strobes are asserted due to the microcode used for exception processing. The processor does not recognize anything on data lines D_8 through D_{15} at this time.

Figure 42 Interrupt Acknowledge Sequence Flow Chart



* Although a vector number is one byte, both data strobes are asserted due to the microcode used for exception processing. The processor does not recognize anything on data lines D_8 through D_{15} at this time.

Figure 43 Interrupt Acknowledge Sequence Timing Diagram


Last Bus Cycle of Instruction (During Which Interrupt Was Recognized)
Stack PCL (SSP-2)
IACK Cycle (Vector Number Acquisition)
Stack Status (SSP-6)
Stack PCH (SSP-4)
Read Vector High (A_{16} ~ A_{23})
Read Vector Low (A_0 ~ A_{15})
Fetch First Word of Instruction of Interrupt Routine

Note: SSP refers to the value of the supervisor stack pointer before the interrupt occurs.

Figure 44 Interrupt Exception Timing Sequence

Priority level seven is a special case. Level seven interrupts cannot be inhibited by the interrupt priority mask, thus providing a "non-maskable interrupt" capability. An interrupt is generated each time the interrupt request level changes from some lower level to level seven. Note that a level seven interrupt may still be caused by the level comparison if the request level is a seven and the processor priority is set to a lower level by an instruction.

UNINITIALIZED INTERRUPT

An interrupting device asserts $\overline{\text{VPA}}$ or provides an interrupt vector during an interrupt acknowledge cycle to the 68000.
If the vector register has not been initialized, the responding HD68000 Family peripheral will provide vector 15, the unitialized interrupt vector. This provides a uniform way to recover from a programming error.

SPURIOUS INTERRUPT

If during the interrupt acknowledge cycle no device responds by asserting $\overline{\text{DTACK}}$ or $\overline{\text{VPA}}$, the bus error line should be asserted to terminate the vector acquisition. The processor separates the processing of this error from bus error by fetching the spurious interrupt vector instead of the bus error vector. The processor then proceeds with the usual exception processing.

INSTRUCTION TRAPS

Traps are exceptions caused by instructions. They arise either from processor recognition of abnormal conditions during instruction execution, or from use of instructions whose normal behavior is trapping.

Some instructions are used specifically to generate traps. The TRAP instruction always forces an exception, and is useful for implementing system calls for user programs. The TRAPV and CHK instructions force an exception if the user program detects a runtime error, which may be an arithmetic overflow or a subscript out of bounds.

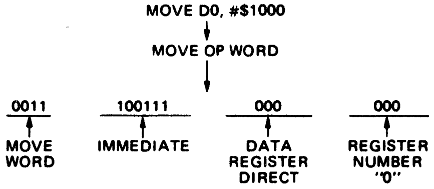
The signed divide (DIVS) and unsigned divide (DIVU) instructions will force an exception if a division operation is attempted with a divisor of zero.

ILLEGAL AND UNIMPLEMENTED INSTRUCTIONS

Illegal instruction is the term used to refer to any of the word bit patterns which are not the bit pattern of the first word of a legal instruction. During instruction execution, if such an instruction is fetched, an illegal instruction exception occurs.

Word patterns with bits 15 through 12 equaling 1010 or 1111 are distinguished as unimplemented instructions and separate exception vectors are given to these patterns to permit efficient emulation. This facility allows the operating system to detect program errors, or to emulate unimplemented instructions in software.

ILLEGAL INSTRUCTION EXAMPLE



PRIVILEGE VIOLATIONS

In order to provide system security, various instructions are privileged. An attempt to execute one of the privileged instructions while in the user state will cause an exception. The privileged instruction are:

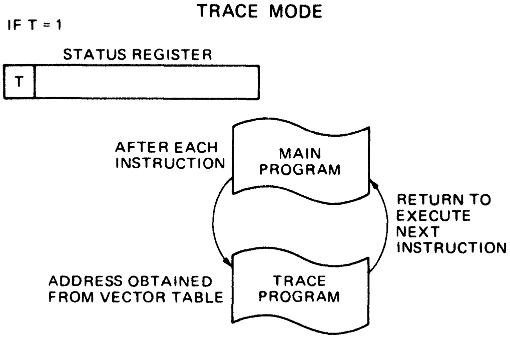
STOP	AND (word) Immediate to SR
RESET	EOR (word) Immediate to SR
RTE	OR (word) Immediate to SR
MOVE to SR	MOVE USP

TRACING

To aid in program development, the 68000 includes a facility to allow instruction by instruction tracing. In the trace state, after each instruction is executed an exceptions is forced, allowing a debugging program to monitor the execution of the program under test.

The trace facility uses the T-bit in the supervisor portion of the status register. If the T-bit is negated (off), tracing is disabled, and instruction execution proceeds from instruction to instruction as normal. If the T-bit is asserted (on) at the beginning of the execution of an instruction, a trace exception will be generated after the execution of that instruction is completed. If the instruction is not executed. either because an interrupt is taken, or the instruction is illegal or privileged, the trace exception does not occur. The trace exception also does not occur if the instruction is aborted by a reset, bus error, or address error exception. If the instruction is indeed executed and an interrupt is pending on completion, the trace exception is processed before the interrupt exception. If, during the execution of the instruction, an exception is forced by that instruction, the forced exception is processed before the trace exception.

As an extreme illustration of the above rules, consider the arrival of an interrupt during the execution of a TRAP instruction while tracing is enabled. First the trap exception is processed, then the trace exception, and finally the interrupt exception. Instruction execution resumes in the interrupt handler routine.



1 If, upon completion of an instruction, T = 1, go to trace exception processing
2 Execute trace exception sequence
3 Execute trace service routine
4. At the end of the service routine, execute return from exception (RTE)

BUS ERROR

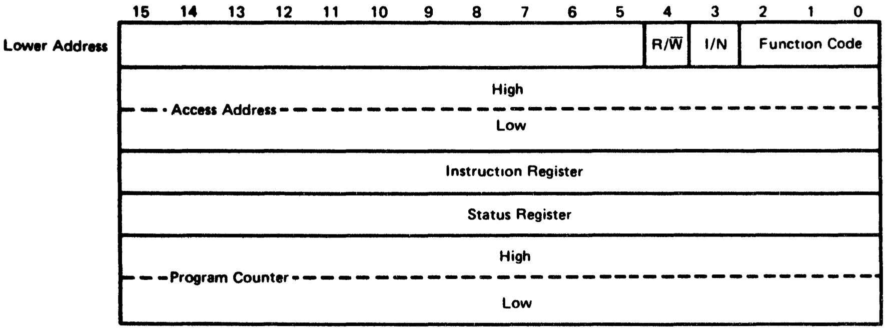
Bus error exceptions occur when the external logic requests that a bus error be processed by an exception. The current bus cycle which the processor is making is then aborted. Whether the processor was doing instruction or exception processing, that processing is terminated, and the processor immediately begins exception processing.

Exception processing for bus error follows the usual sequence of steps. The status register is copied, the supervisor state is entered, and the trace state is turned off. The vector number is generated to refer to the bus error vector. Since the processor was not between instructions when the bus error exception request was made, the context of the processor is more detailed. To save more of this context, additional information is saved on the supervisor stack. The program counter and the copy of the status register are of course saved. The value saved for the program counter is advanced by some amount, one to five words beyond the address of the first word of the instruction which made the reference causing the bus error. If the bus error occurred during the fetch of the next instruction, the saved program counter has a value in the vicinity of the current instruction, even if the current instruction is a branch, a jump, or a return instruction. Besides the usual information, the processor saves its internal copy of the first word of the instruction being processed, and the address which was being accessed by the aborted bus cycle. Specific information about the access is also saved: whether it was a read or a write, whether the processor was processing an instruction or not, and the classification displayed on the function code outputs when the bus error occurred. The processor is processing an instruction if it is in the normal state or processing a Group 2 exception; the processor is not processing an instruction if it is processing a Group 0 or a Group 1 exception. Figure 45 illustrates how this information is organized on the supervisor stack. Although this information is not sufficient in general to effect full recovery from the bus error, it does allow software diagnosis. Finally, the processor commences instruction processing at the address contained in the vector. It is the responsibility of the error handler routine to clean up the stack and determine where to continue execution.

If a bus error occurs during the exception processing for a bus error, address error, or reset, the processor is halted, and all processing cases. This simplifies the detection of catastrophic system failure, since the processor removes itself from the system rather than destroy all memory contents. Only the $\overline{\text{RES}}$ pin can restart a halted processor.

ADDRESS ERROR

Address error exceptions occur when the processor attempts to access a word or a long word operand or an instruction at an odd address. The effect is much like an internally generated bus error, so that the bus cycle is aborted, and the processor ceases whatever processing it is currently doing and begins exception processing. After exception processing commences, the sequence is the same as that for bus error including the information that is stacked, except that the vector number refers to the address error vector instead. Likewise, if an address error occurs during the exception processing for a bus error, address error, or reset, the processor is halted. As shown in Figure 46, an address error will execute a short bus cycle followed by exception processing.



R/W̄ (read/write): write = 0, read = 1. I/N (instruction/not)· instruction = 0, not = 1

Figure 45 Exception Stack Order (Group 0)

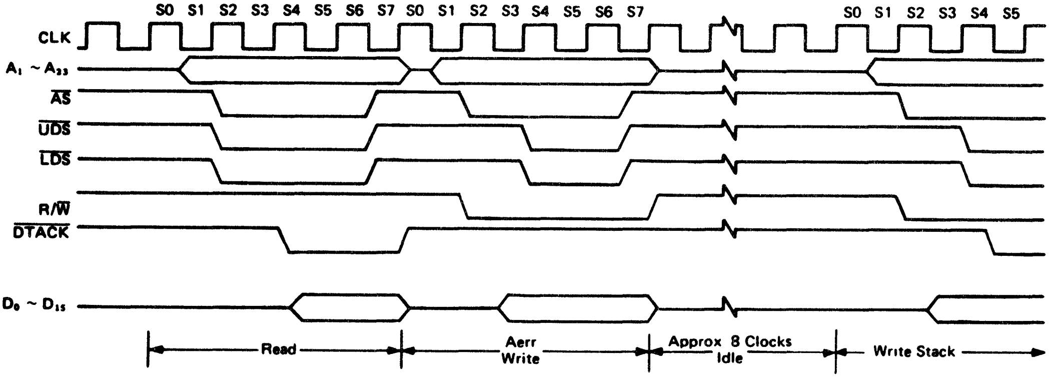


Figure 46 Address Error Timing

■ INTERFACE WITH HD6800 PERIPHERALS

Hitachi's extensive line of HD6800 peripherals are directly compatible with the 68000. Some of these devices that are particularly useful are:

HD6821	Peripheral Interface Adapter
HD6840	Programmable Timer Module
HD6843	Floppy Disk Controller
HD6845S	CRT Controller
HD46508	Analog Data Acquisition Unit
HD6850	Asynchronous Communication Interface Adapter
HD6852	Synchronous Serial Data Adapter

To interface the synchronous HD6800 peripherals with the asynchronous 68000, the processor modifies its bus cycle to meet the HD6800 cycle requirements whenever an HD6800 device address is detected. This is possible since both processors use memory mapped I/O. Figure 48 is a flow chart of the interference operation between the processor and HD6800 devices.

• DATA TRANSFER OPERATION

Three signals on the processor provide the HD6800 interface. They are enable (E), valid memory address (VMA̅), and valid peripheral address (VPA̅). Enable corresponds to the E or ϕ_2 signal in existing HD6800 systems. The bus frequency is one tenth of the incoming 68000 clock frequency. The timing of E allows 1 MHz peripherals to be used with an 8 MHz 68000. Enable has a 60/40 duty cycle; that is, it is low for six input clocks and high for four input clocks. This duty cycle allows the processor to do successive VPA̅ accesses on successive E pulses.

HD6800 cycle timing is given in Figures 49 and 50. At state zero (S0) in the cycle, the address bus is in the high-impedance state. A function code is asserted on the function code output lines. One-half clock later, in state 1 the address bus is released from the high-impedance state.

During state 2, the address strobe (AS̅) is asserted to indicate tnat there is a valid address on the address bus. If the bus cycle is a read cycle, the upper and/or lower data strobes are also asserted in state 2. If the bus cycle is a write cycle,

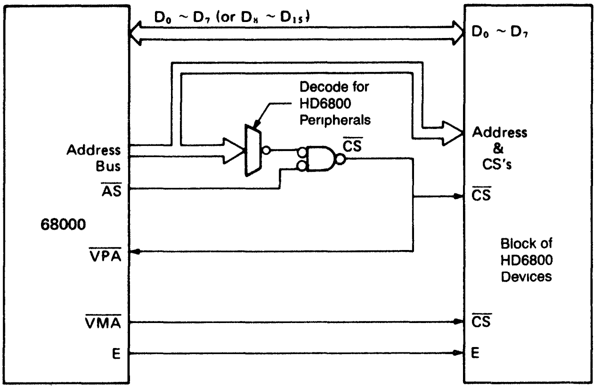


Figure 47 Connection of HD6800 Peripherals

the read/write ($R/\overline{W}$) signal is switched to low (write) during state 2. One half clock later, in state 3, the write data is placed on the data bus, and in state 4 the data strobes are issued to indicate valid data on the data bus. The processor now inserts wait states until it recognizes the assertion of $\overline{VPA}$.

The $\overline{VPA}$ input signals the processor that the address on the bus is the address of an HD6800 device (or an area reserved for HD6800 devices) and that the bus should conform to the ϕ_2 transfer characteristics of the HD6800 bus. Valid peripheral address is derived by decoding the address bus, conditioned by address strobe. Chip select for the HD6800 peripherals should be derived by decoding the address bus conditioned by $\overline{VMA}$.

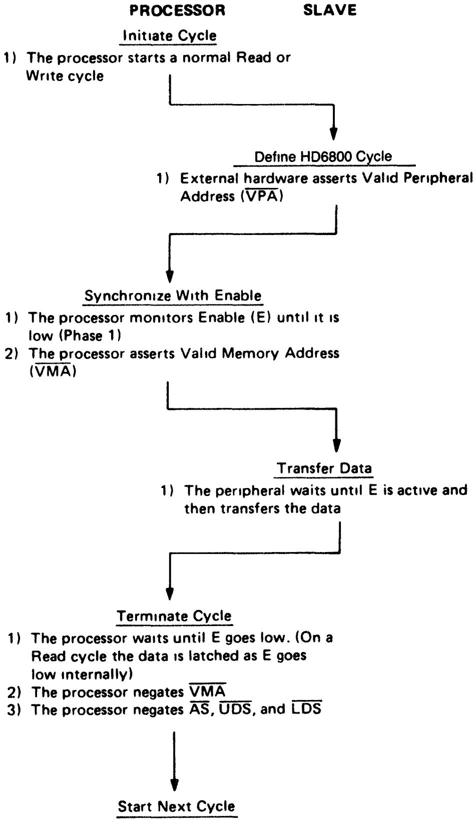
After the recognition of $\overline{VPA}$, the processor assures that the Enable (E) is low, by waiting if necessary, and subsequently asserts $\overline{VMA}$. Valid memory address is then used as part of the chip select equation of the peripheral. This ensures that the HD6800 peripherals are selected and deselected at the correct time. The peripheral now runs in cycle during the high portion of the E signal. Figures 49 and 50 depict the best and worst case HD6800 cycle timing. This cycle length is dependent strictly upon when $\overline{VPA}$ is asserted in relationship to the E clock.

dependent strictly upon when $\overline{VPA}$ is asserted in relationship to the E clock.

If we assume that external circuitry asserts $\overline{VPA}$ as soon as possible after the assertion of $\overline{AS}$, then $\overline{VPA}$ will be recognized as being asserted on the falling edge of S4. In this case, no "extra" wait cycles will be inserted prior to the recognition of $\overline{VPA}$ asserted and only the wait cycles inserted to synchronize with the E clock will determine the total length of the cycle. In any case, the synchronization delay will be some integral number of clock cycles within the following two extremes:

1. Best Case – $\overline{VPA}$ is recognized as being asserted on the falling edge three clock cycles before E rises (or three clock cycles after E falls).
2. Worst Case – $\overline{VPA}$ is recognized as being asserted on the falling edge two clock cycles before E rises(or four clock cycles after E falls).

During a read cycle, the processor latches the peripheral data in state 6. For all cycles, the processor negates the address and data strobes one half clock cycle later in state 7, and the Enable signal goes low at this time. Another half clock later, the address bus is put in the high-impedance state. During a write cycle, the data bus is put in the high-impedance state



PROCESSOR SLAVE

Initiate Cycle

1) The processor starts a normal Read or Write cycle

Define HD6800 Cycle

1) External hardware asserts Valid Peripheral Address ($\overline{VPA}$)

Synchronize With Enable

1) The processor monitors Enable (E) until it is low (Phase 1)
2) The processor asserts Valid Memory Address ($\overline{VMA}$)

Transfer Data

1) The peripheral waits until E is active and then transfers the data

Terminate Cycle

1) The processor waits until E goes low. (On a Read cycle the data is latched as E goes low internally)
2) The processor negates $\overline{VMA}$
3) The processor negates $\overline{AS}$, $\overline{UDS}$, and $\overline{LDS}$

Start Next Cycle

Figure 48 HD6800 Interface Flow Chart

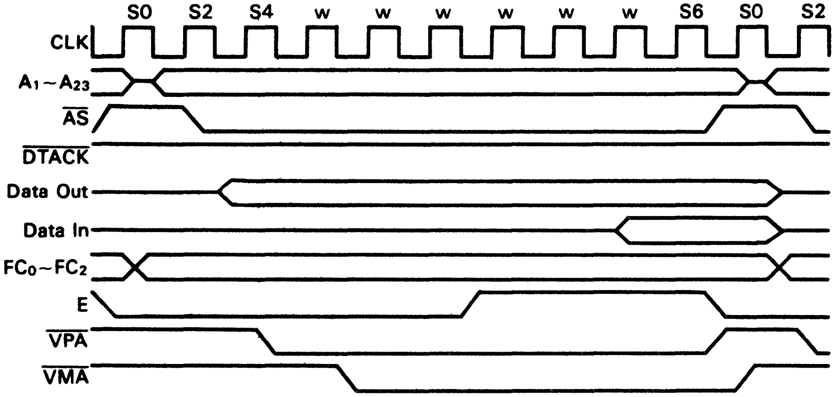


Figure 49 68000 to HD6800 Peripheral Timing—Best Case

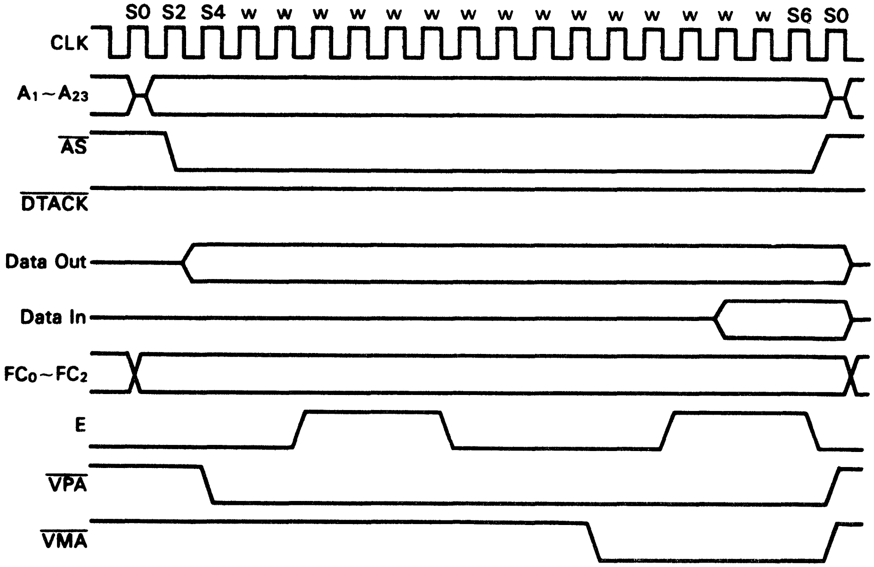
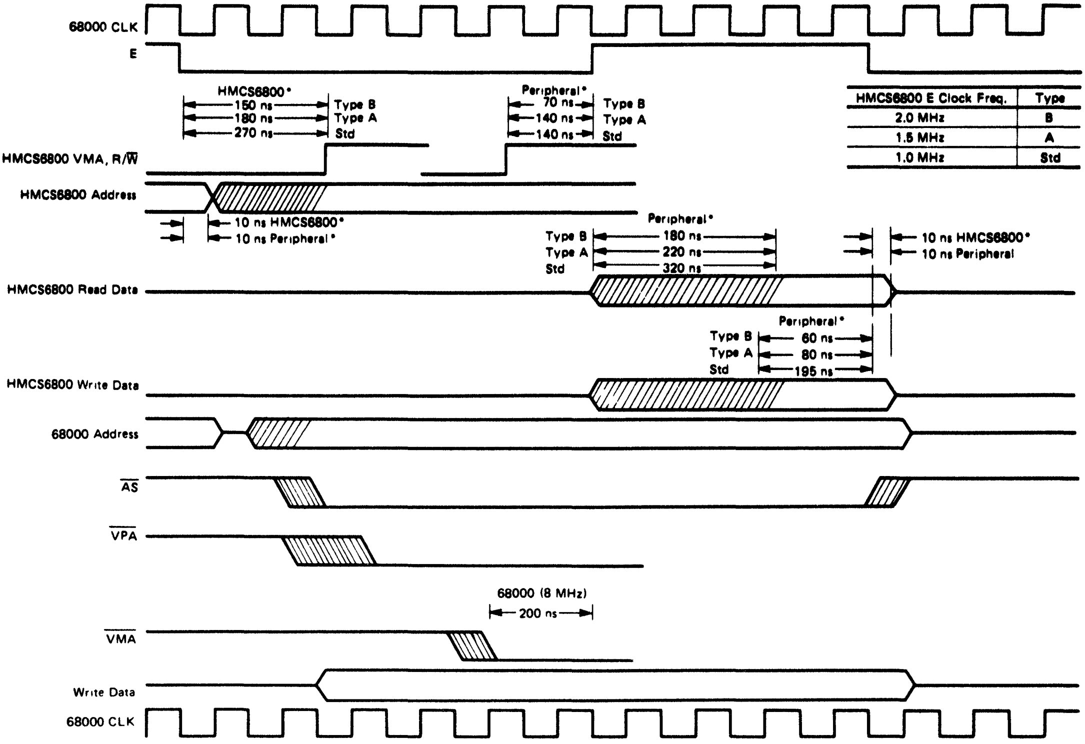


Figure 50 68000 to HD6800 Peripheral Timing—Worst Case

4



HMCS6800 E Clock Freq.	Type
2.0 MHz	B
1.5 MHz	A
1.0 MHz	Std

* Times are expressed for different device clock frequencies

Figure 51 68000 to HD6800 Peripheral Timing Diagram

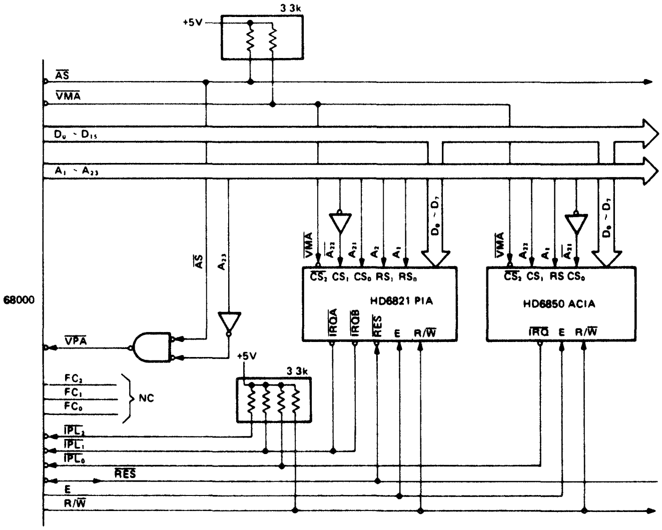


Figure 52 HD6800 Interface—Example 1

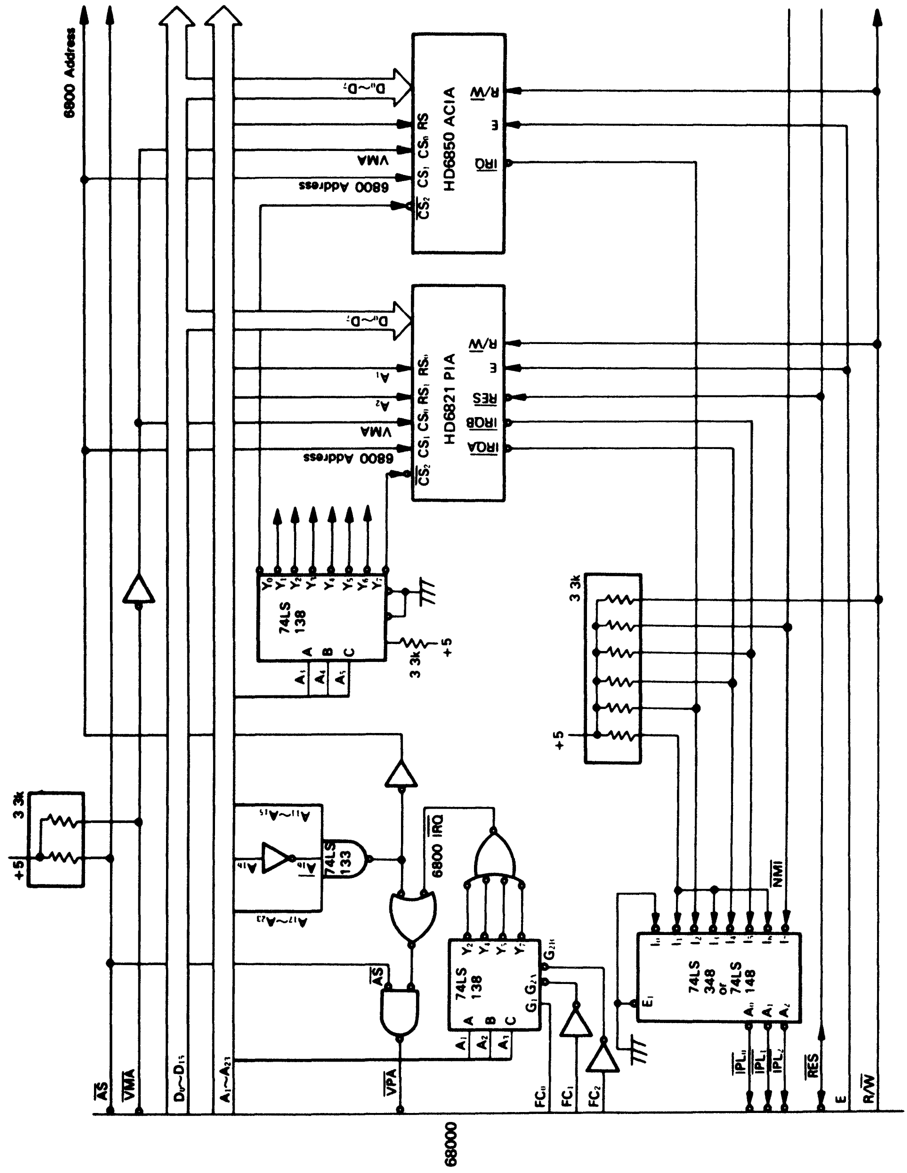


Figure 53 HD6800 Interface—Example 2

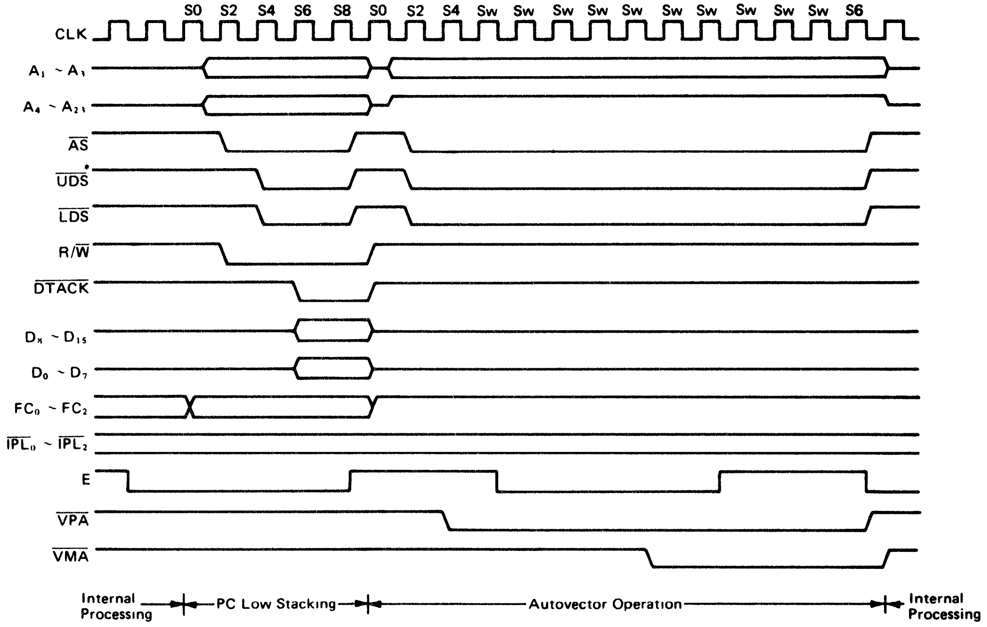
and the read/write signal is switched high. The peripheral logic must remove $\overline{\mathrm{VPA}}$ within one clock after address strobe is negated.

Figure 51 shows the timing required by HD6800 peripherals, the timing specified for HD6800, and the corresponding timing for the 68000. Two example systems with HD6800 peripherals are shown in Figures 52 and 53. The system in Figure 52 reserves the upper eight megabytes of memory for HD6800 peripherals. The system in Figure 53 is more efficient with memory and easily expandable, but more complex.

$\overline{\mathrm{DTACK}}$ should not be asserted while $\overline{\mathrm{VPA}}$ is asserted. Notice that the 68000 $\overline{\mathrm{VMA}}$ is active low, contrasted with the active high HD6800 VMA. This allows the processor to put its buses in the high-impedance state on DMA requests without inadvertently selecting peripherals.

• INTERRUPT OPERATION

During an interrupt acknowledge cycle while the processor is fetching the vector, if $\overline{\mathrm{VPA}}$ is asserted, the 68000 will assert $\overline{\mathrm{VMA}}$ and complete a normal HD6800 read cycle as shown in Figure 54. The processor will then use an internally generated vector that is a function of the interrupt being serviced. This process is known as autovectoring. The seven autovectors are vector numbers 25 through 31 (decimal).



* Although a vector number is one byte, both data strobes are asserted due to the microcode used for exception processing. The processor does not recognize anything on data lines D8 through D_{15} at this time

Figure 54 Autovector Operation Timing Diagram

This operates in the same fashion (but is not restricted to) HD6800 interrupt sequence. The basic difference is that there are six normal interrupt vectors and one NMI type vector. As with both the HD6800 and the 68000's normal vectored interrupt, the interrupt service routine can be located anywhere in the address space. This is due to the fact that while the vector numbers are fixed, the contents of the vector table entries are assigned by the user.

Since $\overline{\mathrm{VMA}}$ is asserted autovectoring, the HD6800 peripheral address decoding should prevent unintended accesses.

■ CONDITION CODES COMPUTATION

This provides a discussion of how the condition codes were developed, the meanings of each bit, how they are computed, and how they are represented in the instruction set details.

• CONDITION CODE REGISTER

The condition code register portion of the status register contains five bits:

N – Negative
Z – Zero
V – Overflow
C – Carry
X – Extend

The first four bits are true condition code bits in that they reflect the condition of the result of a processor operation. The X-bit is an operand for multiprecision computations. The carry bit (C) and the multiprecision operand extend bit (X) are separate in the 68000 to simplify the programming model.

• CONDITION CODE REGISTER NOTATION

In the instruction set details, the description of the effect on the condition codes is given in the following form:

Condition Codes:

X	N	Z	V	C

Where

N (negative) set if the most significant bit of the result is set. Cleared otherwise.

Z (zero) set if the result equals zero. Cleared otherwise.

V (overflow) set if there was an arithmetic overflow. This implies that the result is not representable in the operand size. Cleared otherwise.

C (carry) set if a carry is generated out of the most significant bit of the operands for an addition. Also set if a borrow is generated in a subtraction. Cleared otherwise.

X (extend) transparent to data movement. When affected, it is set the same as the C-bit.

The notational convention that appears in the representation of the condition code registers is:

*	set according to the result of the operation
–	not affected by the operation
0	cleared
1	set
U	undefined after the operation

• CONDITION CODE COMPUTATION

Most operations take a source operand and a destination operand, compute, and store the result in the destination location. Unary operations take a destination operand, compute, and store the result in the destination location. Table 22 details how each instruction sets the condition codes.

Table 22 Condition Code Computations

Operations	X	N	Z	V	C	Special Definition
ABCD	*	U	?	U	?	C = Decimal Carry $Z = Z \cdot \overline{Rm} \cdot .. \cdot \overline{R0}$
ADD, ADDI, ADDQ	*	*	*	?	?	$V = Sm \cdot Dm \cdot \overline{Rm} + \overline{Sm} \cdot \overline{Dm} \cdot Rm$ $C = Sm \cdot Dm + \overline{Rm} \cdot Dm + Sm \cdot \overline{Rm}$
ADDX	*	*	?	?	?	$V = Sm \cdot Dm \cdot \overline{Rm} + \overline{Sm} \cdot \overline{Dm} \cdot Rm$ $C = Sm \cdot Dm + \overline{Rm} \cdot Dm + Sm \cdot \overline{Rm}$ $Z = Z \cdot \overline{Rm} \cdot .. \cdot \overline{R0}$
AND, ANDI, EOR, EORI, MOVEQ, MOVE, OR, ORI, CLR, EXT, NOT, TAS, TST	–	*	*	0	0	
CHK	–	*	U	U	U	
SUB, SUBI SUBQ	*	*	*	?	?	$V = \overline{Sm} \cdot Dm \cdot \overline{Rm} + Sm \cdot \overline{Dm} \cdot Rm$ $C = Sm \cdot \overline{Dm} + Rm \cdot \overline{Dm} + Sm \cdot Rm$
SUBX	*	*	?	?	?	$V = \overline{Sm} \cdot Dm \cdot \overline{Rm} + Sm \cdot \overline{Dm} \cdot Rm$ $C = Sm \cdot \overline{Dm} + Rm \cdot \overline{Dm} + Sm \cdot Rm$ $Z = Z \cdot \overline{Rm} \cdot . \cdot \overline{R0}$
CMP, CMPI, CMPM	–	*	*	?	?	$V = \overline{Sm} \cdot Dm \cdot \overline{Rm} + Sm \cdot \overline{Dm} \cdot Rm$ $C = Sm \cdot \overline{Dm} + Rm \cdot \overline{Dm} + Sm \cdot Rm$
DIVS, DIVU	–	*	*	?	0	V = Division Overflow
MULS, MULU	–	*	*	0	0	
SBCD, NBCD	*	U	?	U	?	C = Decimal Borrow $Z = Z \cdot \overline{Rm} \cdot ... \cdot \overline{R0}$
NEG NEGX	* *	* *	* ?	? ?	? ?	$V = Dm \cdot Rm, C = Dm + Rm$ $V = Dm \cdot Rm, C = Dm + Rm$ $Z = Z \cdot \overline{Rm} \cdot .. \cdot \overline{R0}$
BTST, BCHG, BSET, BCLR	–	–	?	–	–	$Z = \overline{Dn}$
ASL	*	*	*	?	?	$V = Dm \cdot (\overline{D_{m-1}} + . . + \overline{D_{m-r}})$ $+ \overline{Dm} \cdot (D_{m-1} + ... + D_{m-r})$ $C = D_{m-r+1}$
ASL (r = 0)	–	*	*	0	0	
LSL, ROXL	*	*	*	0	?	$C = D_{m-r+1}$
LSR (r = 0)	–	*	*	0	0	
ROXL (r = 0)	–	*	*	0	?	C = X
ROL	–	*	*	0	?	$C = D_{m-r+1}$
ROL (r = 0)	–	*	*	0	0	
ASR, LSR, ROXR	*	*	*	0	?	$C = D_{r-1}$
ASR, LSR (r = 0)	–	*	*	0	0	
ROXR (r = 0)	–	*	*	0	?	C = X
ROR	–	*	*	0	?	$C = D_{r-1}$
ROR (r = 0)	–	*	*	0	0	

– Not affected
U Undefined
? Other— see Special Definition

* General Case
X = C
N = Rm
$Z = \overline{Rm} \cdot \quad \cdot \overline{R0}$

Sm – Source operand most significant bit
Dm – Destination operand most significant bit
Rm – Result bit most significant bit
n – bit number
r – shift amount

4

• CONDITIONAL TESTS

Table 23 lists the condition names, encodings, and tests for the conditional branch and set instructions. The test associated with each condition is a logical formula based on the current state of the condition codes. If this formula evaluates to 1, the condition succeeds, or is true. If the formula evaluates to 0, the condition is unsuccessful, or false. For example, the T condition always succeeds, while the EQ condition succeeds only if the Z bit is currently set in the condition codes.

Table 23 Conditional Tests

Mnemonic	Condition	Encoding	Test
T	true	0000	1
F	false	0001	0
HI	high	0010	$\overline{C} \cdot \overline{Z}$
LS	low or same	0011	$C + Z$
CC	carry clear	0100	$\overline{C}$
CS	carry set	0101	C
NE	not equal	0110	$\overline{Z}$
EQ	equal	0111	Z
VC	overflow clear	1000	$\overline{V}$
VS	overflow set	1001	V
PL	plus	1010	$\overline{N}$
MI	minus	1011	N
GE	greater or equal	1100	$N \cdot V + \overline{N} \cdot \overline{V}$
LT	less than	1101	$N \cdot \overline{V} + \overline{N} \cdot V$
GT	greater than	1110	$N \cdot V \cdot \overline{Z} + \overline{N} \cdot \overline{V} \cdot \overline{Z}$
LE	less or equal	1111	$Z + N \cdot \overline{V} + \overline{N} \cdot V$

■ INSTRUCTION SET

The following paragraphs provide information about the addressing categories and instruction set of the 68000.

• ADDRESSING CATEGORIES

Effective address modes may be categorized by the ways in which they may used. The following classifications will be used in the instruction definitions.

Data If an effective address mode may be used to refer to data operands, it is considered a data addressing effective address mode.

Memory If an effective address mode may be used to refer to memory operands, it is considered a memory addressing effective address mode.

Alterable If an effective address mode may be used to refer to alterable (writeable) operands, it is considered an alterable addressing effective address mode.

Control If an effective address mode may be used to refer to memory operands without an associated size, it is considered a control addressing effective address mode.

Table 24 shows the various categories to which each of the effective address modes belong. Table 25 is the instruction set summary.

The status register addressing mode is not permitted unless it is explicitly mentioned as a legal addressing mode.

These categories may be combined so that additional, more restrictive, classifications may be defined. For example, the instruction descriptions use such classifications as alterable memory or data alterable. The former refers to those addressing modes which are both alterable and memory addresses, and the latter refers to addressing modes which are both data and alterable.

• INSTRUCTION PRE-FETCH

The 68000 uses a 2-word tightly-coupled instruction prefetch mechanism to enhance performance. This mechanism is described in terms of the microcode operations involved. If the execution of an instruction is defined to begin when the microroutine for that instruction is entered, some features of the prefetch mechanism can be described.

1) When execution of an instruction begins, the operation word and the word following have already been fetched. The operation word is in the instruction decoder.
2) In the case of multi-word instructions, as each additional word of the instruction is used internally, a fetch is made to the instruction stream to replace it.
3) The last fetch from the instruction stream is made when the operation word is discarded and decoding is started on the next instruction.
4) If the instruction is a single-word instruction causing a branch, the second word is not used. But because this word is fetched by the preceding instruction, it is impossible to avoid this superfluous fetch. In the case of an interrupt or trace exception, both words are not used.
5) The program counter usually points to the last word fetched from the instruction stream.

Table 24 Effective Addressing Mode Categories

Effective Address Modes	Mode	Register	Data	Addressing Categories		
				Memory	Control	Alterable
Dn	000	register number	X	–	–	X
An	001	register number	–	–	–	X
An@	010	register number	X	X	X	X
An@+	011	register number	X	X	–	X
An@-	100	register number	X	X	–	X
An@(d)	101	register number	X	X	X	X
An@(d, ix)	110	register number	X	X	X	X
xxx.W	111	000	X	X	X	X
xxx.L	111	001	X	X	X	X
PC@(d)	111	010	X	X	X	–
PC@(d, ix)	111	011	X	X	X	–
#xxx	111	100	X	X	–	–

The following example illustrates many of the features of instruction prefetch. The contents of memory are assumed to be as illustrated in Figure 55.

```
            ORG      0                          DEFINE RESTART VECTOR
            DC.L     INISSP                     INITIAL SYSTEM STACK POINTER
            DC.L     RESTART                    RESTART SYSTEM ENTRY POINT
            ORG      INTVECTOR                  DEFINE AN INTERRUPT VECTOR
            DC.L     INTHANDLER                 HANDLER ADDRESS FOR THIS VECTOR
            ORG                                 SYSTEM RESTART CODE
RESTART:
            NOP                                 NO OPERATION EXAMPLE
            BRA.S    LABEL                      SHORT BRANCH
            ADD.W    D0, D1                     ADD REGISTER TO REGISTER
LABEL·
            SUB.W    DISP(A0), A1               SUBTRACT REGISTER INDIRECT WITH OFFSET
            CMP.W    D2, D3                     COMPARE REGISTER TO REGISTER
            SGE.B    D7                         Scc TO REGISTER
            ..
            ...
INTHANDLER:
            MOVE.W   LONGADR1, LONGADR2         MOVE WORD FROM AND TO LONG ADDRESS
            NOP                                 NO OPERATION
            SWAP.W                              REGISTER SWAP
```

Figure 55 Instruction Prefetch Example, Memory Contents

The sequence we shall illustrate consists of the power-up reset, the execution of NOP, BRA, SUB, the taking of an interrupt, and the execution of the MOVE.W xxx. L to yyy.L. The order of operations described within each microroutine is not exact, but is intended for illustrative purpose only.

Microroutine	Operation	Location	Operand
Reset	Read	0	SSP High
	Read	2	SSP Low
	Read	4	PC High
	Read	6	PC Low
	Read	(PC)	NOP
	Read	+(PC)	BRA
	<begin NOP>		
NOP	Read	+(PC)	ADD
	<begin BRA>		
BRA	PC=PC+d		
	Read	(PC)	SUB
	Read	+(PC)	DISP
	<begin SUB>		
SUB	Read	+(PC)	CMP
	Read	DISP(A0)	<src>
	Read	+(PC)	SGE
	<begin CMP>	<take INT>	
INTERRUPT	Write	-(SSP)	PC Low
	Read	<INT ACK>	Vector #
	Write	-(SSP)	SR
	Write	-(SSP)	PC High
	Read	(VR)	PC High
	Read	+(VR)	PC Low
	Read	(PC)	MOVE
	Read	+(PC)	xxx High
	<begin MOVE>		
MOVE	Read	+(PC)	xxx Low
	Read	+(PC)	yyy High
	Read	xxx	<src>
	Read	+(PC)	yyy Low
	Write	yyy	<dest>
	Read	+(PC)	NOP
	Read	+(PC)	SWAP
	<begin NOP>		

Figure 56 Instruction Prefetch Example

- **DATA PREFETCH**

Normally the 68000 prefetches only instructions and not data. However, when the MOVEM instruction is used to move data from memory to registers, the data stream is prefetched in order to optimize performance. As a result, the processor reads one extra word beyond the higher end of the source area. For example, the instruction sequence in Figure 57 will operate as shown in Figure 58.

	...		
	MOVEM.L	A, D0/D1	MOVE TWO LONGWORDS INTO REGISTERS
	...		
A	DC.W	1	WORD 1
B	DC.W	2	WORD 2
C	DC.W	3	WORD 3
D	DC.W	4	WORD 4
E	DC.W	5	WORD 5
F	DC.W	6	WORD 6

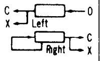
Figure 57 MOVEM Example, Memory Contents

Assume Effective Address Evaluation is Already Done

Microroutine	Operation	Location	Other Operations
	...		
MOVEM	Read	A	
			Prepare to Fill D0
	Read	B	A → D0H
	Read	C	B → D0L
			Prepare to Fill D1
	Read	D	C → D1H
	Read	E	D → D1L
			Detect Register List Complete

Figure 58 MOVEM Example, Operation Sequence

Table 25 Instruction Set

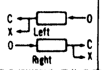
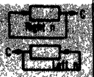
Mnemonic Operation	Size	Addr Mode	Dn		An		(An)		(An) +		- (An)		d(An)		d(An, Xi)		Abs W		Abs L		d(PC)		d(PC, Xi)		s = Immed d = SR/CC		Opcode Bit Pattern 1111 11		Boolean	Condition Codes
			#	~	#	~	#	~	#	~	#	~	#	~	#	~	#	~	#	~	#	~	#	~	#	~	5432 1098	7654 3210		XNZVC
ABCD Add Digits	B	s Dn d	2	6																							1100 RRR1	0000 0rrr	d10 + s10 + X →d	*U*U*
		s (An) d									2	18															1100 RRR1	0000 1rrr		
ADD Add Binary	B W	s Dn d			ADDA		2	12	2	12	2	14	4	16	4	18	4	16	6	20							1101 DDD1	SSEE EEEE	d + Dn →d	*****
		d Dn s	2	4	2•	4	2	8	2	8	2	10	4	12	4	14	4	12	6	16	4	12	4	14	4	8	1101 DDD0	SSee eeee	Dn + s →Dn	
	L	s Dn d			ADDA		2	20	2	20	2	22	4	24	4	26	4	24	6	28							1101 DDD1	10EE EEEE	d + Dn →d	
		d Dn s	2	8	2	8	2	14	2	14	2	16	4	18	4	20	4	18	6	22	4	18	4	20	6	14	1101 DDD0	10ee eeee	Dn + s →Dn	
ADDA Add Address	W	d An s	2	8	2	8	2	12	2	12	2	14	4	16	4	18	4	16	6	20	4	16	4	18	4	12	1101 AAA0	11ee eeee	An + s →An	-----
	L	d An s	2	8	2	8	2	14	2	14	2	16	4	18	4	20	4	18	6	22	4	18	4	20	6	14	1101 AAA1	11ee eeee		
ADDI Add Immed	B W	s Imm d	4	8	ADDA		4	16	4	16	4	18	6	20	6	22	6	20	8	24							0000 0110	SSEE EEEE	d + # →d	*****
	L	s Imm d	6	16	ADDA		6	28	6	28	6	30	8	32	8	34	8	32	10	36										
ADDQ Add Quick	B W	s·Imm3 d	2	4	2•	4	2	12	2	12	2	14	4	16	4	18	4	16	6	20							0101 QQQ0	SSEE EEFE	d + # →d	*****
	L	s Imm3 d	2	8	2	8	2	20	2	20	2	22	4	24	4	26	4	24	6	28										
ADDX Add Multi-precision	B W	s Dn d	2	4																							1101 RRR1	SS00 0rrr	d + s + X →d	*****
		s -(An) d									2	18															1101 RRR1	SS00 1rrr		
	L	s Dn d	2	8																							1101 RRR1	1000 0rrr		
		s -(An) d									2	30															1101 RRR1	1000 1rrr		
AND Logic And	B W	s Dn d					2	12	2	12	2	14	4	16	4	18	4	16	6	20							1100 DDD1	SSEE FEEE	d < and > Dn →d	-**00
		d Dn s	2	4			2	8	2	8	2	10	4	12	4	14	4	12	6	16	4	12	4	14	4	8	1100 DDD0	SSee eeee	Dn < and > s →Dn	
	L	s Dn d					2	20	2	20	2	22	4	24	4	26	4	24	6	28							1100 DDD1	10EE EEEE	d < and > Dn →d	
		d:Dn s	2	8			2	14	2	14	2	16	4	18	4	20	4	18	6	22	4	18	4	20	6	14	1100 DDD0	10ee eeee	Dn < and > s →Dn	
ANDI And Immed	B W	s Imm d	4	8			4	16	4	16	4	18	6	20	6	22	6	20	8	24					4	20	0000 0010	SSEE EEEE	d < and > # →d	-**00
	L	s Imm d	6	16			6	28	6	28	6	30	8	32	8	34	8	32	10	36										
ASL, ASR Arithmetic Shift	B W	count:Dn d	2	6 + 2n																							1110 rrrf	SS10 0DDD		*****
		count·#1~8 d	2	6 + 2n																							1110 QQQf	SS00 0DDD		
	L	count:Dn d	2	8 + 2n																							1110 rrrf	1010 0DDD		
		count·#1~8 d	2	8 + 2n																							1110 QQQf	1000 0DDD		
Memory	W	count:1 d					2•	12	2•	12	2•	14	4•	16	4•	18	4•	16	6•	20							1110 000f	11EE EEEE		
BCHG Test and Change	B	bit #:Dn d					2	12	2	12	2	14	4	16	4	18	4	16	6	20							0000 rrr1	01EE EEEE	~(bit) # of d →Z	--*--
		bit #:Imm d					4	16	4	16	4	18	6	20	6	22	6	20	8	24							0000 1000	01EE EEEE	~(bit) # of d →	
	L	bit #:Dn d	2	< 8																							0000 rrr1	01EE EEEE	(bit) # of d	
		bit #:Imm d	4	< 12																							0000 1000	01EE EEEE		
BCLR Test and Clear	B	bit #:Dn d					2	12	2	12	2	14	4	16	4	18	4	17	6	20							0000 rrr1	10EE EEEE		
		bit #·Imm d					4	16	4	16	4	18	6	20	6	22	6	20	8	24							0000 1000	10EE EEEE	~(bit) # of d →Z	--*--
	L	bit #:Dn d	2	< 10																							0000 rrr1	10EE EEEE	0→(bit) # of d	
		bit #:Imm d	4	< 14																							0000 1000	10EE EEEE		
BSET Test and Set	B	bit #:Dn d					2	12	2	12	2	14	4	16	4	18	4	16	6	20							0000 rrr1	11EE EEEE	~(bit) # of d →Z	--*--
		bit #:Imm d					4	16	4	16	4	18	6	20	6	22	6	20	8	24							0000 1000	11EE EEEE	1→(bit) # of d	
	L	bit #:Dn d	2	< 8																							0000 rrr1	11EE EEEE		
		bit #:Imm d	4	< 12																							0000 1000	11EE EEEE		
BTST Bit Test	B	bit #·Dn d					2	8	2	8	2	10	4	12	4	14	4	12	6	16	4	12	4	14	4	10	0000 rrr1	00EE EEEE	~(bit) # of d →Z	--*--
		bit # Imm d					4	12	4	12	4	14	6	16	6	18	6	16	8	20	6	16	6	18			0000 1000	00EE EEEE		
	L	bit #:Dn d	2	6																							0000 rrr1	00EE EEEE		
		bit # Imm d	4	10																							0000 1000	00EE EEEE		
CHK Check Register Against Bounds	W	d:Dn s		< 40	←trap→			< 44		< 44		< 46		< 48		< 50		< 48		< 52		< 48		< 50		< 44	0100 DDD1	10ee eeee	If Dn < 0 or Dn > (bound), then trap	-*UUU
		(bound)	2	10	←no→ trap		2	14	2	14	2	16	4	18	4	20	4	18	6	22	4	18	4	20	4	14				
CLR Clear Operand	B W	d	2	4			2	12	2	12	2	14	4	16	4	18	4	16	6	20							0100 0010	SSEE EEEE	d →MPU 0→d	-0100
	L	d	2	6			2	20	2	20	2	22	4	24	4	26	4	24	6	28										
CMP Compare Binary	B W	d:Dn s	2	4	2•	4	2	8	2	8	2	10	4	12	4	14	4	12	6	16	4	12	4	14	4	8	1011 DDD0	SSee eeee	Dn−s	-****
	L	d:Dn s	2	6	2	6	2	14	2	14	2	16	4	18	4	20	4	18	6	22	4	18	4	20	6	14				
CMPA Compare Address	W	d·An s	2	6	2	6	2	10	2	10	2	12	4	14	4	16	4	12	6	18	4	14	4	16	4	10	1011 AAA0	11ee eeee	An−s	-****
	L	d:An s	2	6	2	6	2	14	2	14	2	16	4	18	4	20	4	18	6	22	4	18	4	20	6	14	1011 AAA1	11ee eeee		
CMPI Compare Imm	B W	s:Imm d	4	8	CMPA		4	12	4	12	4	14	6	16	6	18	6	16	8	20							0000 1100	SSEE EEEE	d−#	-****
	L	s:Imm d	6	14	CMPA		6	20	6	20	6	22	8	24	8	26	8	24	10	28										
CMPM Compare Memory	B/W	s (An) + d							2	12																	1011 RRR1	SS00 1rrr	d−s	-****
	L	s (An) + d							2	20																				
DIVS Divide Signed	W	d:Dn s	2	< 158			2	< 162	2	< 162	2	< 164	4	< 166	4	< 168	4	< 166	6	< 170	4	< 166	4	< 168	4	< 162	1000 DDD1	11ee eeee	Dn32/s16→Dn(rq)	-***0
DIVU Divide Unsigned	W	d Dn s	2	< 140			2	< 144	2	< 144	2	< 146	4	< 148	4	< 150	4	< 148	6	< 152	4	< 148	4	< 150	4	< 144	1000 DDD0	11ee eeee	Dn32 s16 →Dn(rq)	-***0
EOR Exclusive OR Logical	B W	s:Dn d	2	4			2	12	2	12	2	14	4	16	4	18	4	16	6	20							1011 rrr1	SSEE EEEE	d ● Dn →d	-**00
	L	s:Dn d	2	8			2	20	2	20	2	22	4	24	4	26	4	24	6	28										
EORI Exclusive OR Immediate	B W	s·Imm d	4	8			4	16	4	16	4	18	6	20	6	22	6	20	8	24					4	20	0000 1010	SSEE EEEE	d ● # →d	-**00
	L	s:Imm d	6	16			6	28	6	28	6	30	8	32	8	34	8	32	10	36										
EXG Exchange Registers	L	s:Dn d	2	6																							1100 DDD1	0100 0DDD	s↔d	-----
		s:An d	2	6	2	6																					1100 AAA1	0100 1AAA		
																											1100 DDD1	1000 1AAA		
EXT Sign Extend	W	d	2	4																							0100 1000	1000 0DDD	bit 7 →bit 8~15	
	L	d	2	4																							0100 1000	1100 0DDD	bit 15 →bit 16~31	-**00
LEA Load Effective Address	L	d:An s					2	4					4	8	4	12	4	8	6	12	4	8	4	12			0100 AAA1	11ee eeee	s→An	-----
LINK Link and Allocate		disp Imm s			4	16																					0100 1110	0101 0AAA	An→−(SP) SP→An SP + disp→SP	-----

Note Refer to "Condition Code Computations as for condition Code
- • Word only
- < Maximum value
- # Number of Program Bytes
- ~ Number of Clock Periods

Opcode Bit Pattern Key

A Address Register #
C Test Condition
D Data Register #
e Source Effective Address
E Destination Effective Address
f Direction 0- Right 1-Left
M Destination EA Mode
P Displacement
Q Quick Immediate Data
r Source Register
R Destination Register
S Size 00 Byte
01 Word
10-Long Word
11 Another Operation
(In the MOVE Instruction
01 Byte
10 Long Word
11 Word)
V Vector #

(to be continued)

Mnemonic Operation	Size	Addr Mode	Dn		An		(An)		(An) +		- (An)		d(An)		d(An, Xi)		Abs W		Abs L		d(PC)		d(PC, Xi)		s = Immed d = SR/CC		Opcode Bit Pattern 1111 11 5432 1098	7654 3210	Boolean	Condition Codes XNZVC
			#	~	#	~	#	~	#	~	#	~	#	~	#	~	#	~	#	~	#	~	#	~	#	~				
LSL, LSR Logical Shift	B W	count Dn d	2	6 + 2n																							1110 rrrf	SS10 1DDD		***0*
		count #1~8 d	2	6 + 2n																							1110 QQQf	SS00 1DDD		
	L	count Dn d	2	8 + 2n																							1110 rrrf	1010 1DDD		
		count #1~8 d	2	8 + 2n																							1110 QQQf	1000 1DDD		
Memory	W	count 1 d					2•	12	2•	12	2•	14	4•	16	4•	18	4•	16	6•	20							1110 001f	11EE EEEE		
MOVE Move Data	B W	s Dn d	2	4	MOVEA		2	8	2	8	2	8	4	12	4	14	4	12	6	16							00SS RRRM	MMee eeee	s→d	-**00
		s An s	2	4	MOVEA		2	8	2	8	2	8	4	12	4	14	4	12	6	16										
		s (An) d	2	8	MOVEA		2	12	2	12	2	12	4	16	4	18	4	16	6	20										
		s (An)+ d	2	8	MOVEA		2	12	2	12	2	12	4	16	4	18	4	16	6	20										
		s (An) d	2	10	MOVEA		2	14	2	14	2	14	4	18	4	20	4	18	6	22										
		s d(An) d	4	12	MOVEA		4	16	4	16	4	16	6	20	6	22	6	20	8	24										
		s d(An X) d	4	14	MOVEA		4	18	4	18	4	18	6	22	6	24	6	22	8	26										
		s Abs W d	4	12	MOVEA		4	16	4	16	4	16	6	20	6	22	6	20	8	24										
		s Abs L d	6	16	MOVEA		6	20	4	20	6	20	8	24	8	26	8	24	10	28										
		s d(PC) d	4	12	MOVEA		4	16	4	16	4	16	6	20	6	22	6	20	8	24										
		s d(PC X) d	4	14	MOVEA		4	18	4	18	4	18	6	22	6	24	6	22	8	26										
		s Imm d	4	8	MOVEA		4	12	4	12	4	12	6	16	6	18	6	16	8	20										
	L	s Dn d	2	4	MOVEA		2	12	2	12	2	12	4	16	4	18	4	16	6	20										
		s An d	2	4	MOVEA		2	12	2	12	2	12	4	16	4	18	4	16	6	20										
		s (An) d	2	12	MOVEA		2	20	2	20	2	20	4	24	4	26	4	24	6	28										
		s (An) d	2	12	MOVEA		2	20	2	20	2	20	4	24	4	26	4	24	6	28										
		s (An) d	2	14	MOVEA		2	22	2	22	2	22	4	26	4	28	4	26	6	30										
		s d(An) d	4	16	MOVEA		4	24	4	24	4	24	6	28	6	30	6	28	8	32										
		s d(An X) d	4	18	MOVEA		4	26	4	26	4	26	6	30	6	32	6	30	8	34										
		s Abs W d	4	16	MOVEA		4	24	4	24	4	24	6	28	6	30	6	28	8	32										
		s Abs L d	6	20	MOVEA		6	28	4	28	6	28	8	32	8	34	8	32	10	36										
		s d(PC) d	4	16	MOVEA		4	24	4	24	4	24	6	28	6	30	6	28	8	32										
		s d(PC X) d	4	18	MOVEA		4	26	4	26	4	26	6	30	6	32	6	30	8	34										
		s Imm d	6	12	MOVEA		6	20	6	20	6	20	8	24	8	26	8	24	10	28										
MOVE Move to Condition Codes	W	d CCR s	2	12			2	16	2	16	2	18	4	20	4	22	4	20	6	24	4	20	4	22	4	16	0100 0100	11ee eeee	s→CCR	*****
MOVE Move to from Status Reg	W	d SR s	2	12			2	16	2	16	2	18	4	20	4	22	4	20	6	24	4	20	4	22	4	16	0100 0110	11ee eeee	s→SR	*****
		s SR d	2	6			2	12	2	12	2	14	4	16	4	18	4	16	6	20							0100 0000	11FE EEEE	d→MPU SR→d	-----
MOVE Move to from User SP (A7)	L	s USP d			2	4																					0100 1110	0110 1AAA	USP→An	-----
		d USP s			2	4																					0100 1110	0110 0AAA	An→USP	
MOVEA Move Address	W	d An s	2	4	2	4	2	8	2	8	2	10	4	12	4	14	4	12	6	16	4	12	4	14	4	8	0011 AAA0	01ee eeee	s→An	-----
	L	d An s	2	4	2	4	2	12	2	12	2	14	4	16	4	18	4	16	6	20	4	16	4	18	6	12	0010 AAA0	01ee eeee		
MOVEM Move Multiple Registers	W	s Xn d					4	8+4n			4	8+4n	6	12+4n	6	14+4n	6	12+4n	8	16+4n							0100 1000 a7~a0	10EE EEEE d7~d0†	Xn→d	-----
		d Xn s					4	12+4n	4	12+4n			6	16+4n	6	18+4n	6	16+4n	8	20+4n	6	16+4n	6	18+4n			0100 1100 a7~a0	10ee eeee d7~d0	s→Xn**	
	L	s Xn d					4	8+8n			4	8+8n	6	12+8n	6	14+8n	6	12+8n	8	16+8n							0100 1000 a7~a0	11EE EEEE d7~d0†	Xn→d	
		d Xn s					4	12+8n	4	12+8n			6	16+8n	6	18+8n	6	16+8n	8	20+8n	6	16+8n	6	18+8n			0100 1100 a7~a0	11ee eeee d7~d0	s→Xn**	
MOVEP Move Peripheral	W	s Dn d											4	16													0000 DDD1	1000 1AAA	Dn→d by bytes	-----
		s d(An) d	4	16																							0000 DDD1	0000 1AAA	s→Dn by bytes	
	L	s Dn d											4	24													0000 DDD1	1100 1AAA	Dn→d by bytes	
		s d(An) d	4	24																							0000 DDD1	0100 1AAA	s→Dn by bytes	
MOVEQ Move Quick	L	s Imm8 d	2	4																							0111 DDD0	QQQQ QQQQ	#→Dn	-**00
MULS Multiply Signed	W	d Dn s	2	70			2	74	2	74	2	< 76	4	78	4	80	4	< 78	6	< 82	4	78	4	< 80	4	< 74	1100 DDD1	11ee eeee	Dn×s→Dn	-**00
MULU Multiply Unsigned	W	d Dn s	2	< 70			2	< 74	2	< 74	2	< 76	4	< 78	4	< 80	4	< 78	6	< 82	4	< 78	4	< 80	4	< 74	1100 DDD0	11ee eeee	Dn×s→Dn	-**00
NBCD Negate Digit	B	d	2	6			2	12	2	12	2	14	4	16	4	18	4	16	6	20							0100 1000	00EE EFEE	0 d10 X →d	*U*U*
NEG Negate Binary	B W	d	2	4			2	12	2	12	2	14	4	16	4	18	4	16	6	20							0100 0100	SSEE EEEE	0−d→d	*****
	L	d	2	6			2	20	2	20	2	22	4	24	4	26	4	24	6	28										
NEGX Negative Multi precision	B W	d	2	4			2	12	2	12	2	14	4	16	4	18	4	16	6	20							0100 0000	SSEE EEEE	0−d X →d	*****
	L	d	2	6			2	20	2	20	2	22	4	24	4	26	4	24	6	28										
NOT Logical Complement	B W	d	2	4			2	12	2	12	2	14	4	16	4	18	4	16	6	20							0100 0110	SSEE EEEE	~d→d	-**00
	L	d	2	6			2	20	2	20	2	22	4	24	4	26	4	24	6	28										
OR Inclusive OR Logical	B W	s Dn d					2	12	2	12	2	14	4	16	4	18	4	16	6	20							1000 DDD1	SSEE EEEE	d<or>Dn →d	-**00
		d Dn s	2	4			2	8	2	8	2	10	4	12	4	14	4	12	6	16	4	12	4	14	4	8	1000 DDD0	SSee eeee	Dn<or>s →Dn	
	L	s Dn d					2	20	2	20	2	22	4	24	4	26	4	24	6	28							1000 DDD1	10EE EEEE	d<or>Dn →d	
		d Dn s	2	8			2	14	2	14	2	16	4	18	4	20	4	18	6	22	4	18	4	20	6	14	1000 DDD0	10ee eeee	Dn<or>s →Dn	
ORI OR Immediate	B W	s Imm d	4	8			4	16	4	16	4	18	6	20	6	22	6	20	8	24					4	20	0000 0000	SSEE EEEE	d<or>#→d	-**00
	L	s Imm d	6	16			6	30	6	30	6	32	8	34	8	36	8	34	10	38										
PEA Push Effective Address	L	s					2	14					4	18	4	22	4	18	6	22	4	18	4	22			0100 1000	01ee eeee	s → (SP)	-----
ROR, ROL Rotate without X	B W	count Dn d	2	6 + 2n																							1110 rrrf	SS11 1DDD		-**00
		count #1~8 d	2	6 + 2n																							1110 QQQf	SS01 1DDD		
	L	count Dn d	2	8 + 2n																							1110 rrrf	1011 1DDD		
		count #1~8 d	2	8 + 2n																							1110 QQQf	1001 1DDD		
Memory	W	count 1 d					2•	12	2•	12	2•	14	4•	16	4•	18	4•	16	6•	20							1110 011f	11EE EEEE		

Note Refer to Condition Code Computations
as for condition Code

• Word only

< Maximum value

Number of Program Bytes
Number of Clock Periods

** The MPU goes through an extra
null read cycle after a multiple read is done (The last EA+2)

Opcode Bit Pattern Key

A Address Register #
C Test Condition
D Data Register #
e Source Effective Address
E Destination Effective Address

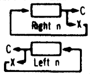
f Direction 0−Right 1−Left
M Destination EA Mode
P Displacement
Q Quick Immediate Data
r Source Register

R Destination Register
S Size 00−Byte
01−Word
10−Long Word
11−Another Operation
V Vector #

(In the MOVE Instruct on
01−Byte
10−Long Word
11−Word)

(to be continued)

HITACHI

Mnemonic Operation	Size	Addr Mode	Dn		An		(An)		(An) +		- (An)		d(An)		d(An, Xi)		Abs W		Abs L		d(PC)		d(PC, Xi)		s = Immed d = SR/CC		Opcode Bit Pattern 1111 11 5432 1098	7654 3210	Boolean	Condition Codes XNZVC
			#	~	#	~	#	~	#	~	#	~	#	~	#	~	#	~	#	~	#	~	#	~	#	~				
ROXR, ROXL Rotate through X Memory	B W L W	count Dn d- count #1~8 d count:Dn d count #1~8 d count 1 d	2 2 2 2	6 + 2n 6 + 2n 8 + 2n 8 + 2n			2 •	12	2 •	12	2 •	14	4 •	16	4 •	18	4 •	16	6 •	20							1110 rrrf 1110 QQQf 1110 rrrf 1110 QQQf 1110 010f	SS11 0DDD SS01 0DDD 1011 0DDD 1001 0DDD 11EE FEFF		* * * 0 *
SBCD Subtract digits	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Scc Set Conditionally	B	cc d	2	6 4			2	12	2	12	2	14	4	16	4	18	4	16	6	20							0101 CCCC	11FF FFFF	d →MPU If cc true 1s •d Else 0s •d	- - - - -
SUB Subtract Binary	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
SUBA Subtract Address	W L	d-An s d-An s	2 2	8 8	2 2	8 8	2 2	12 14	2 2	12 14	2 2	14 16	4 4	16 18	4 4	18 20	4 4	16 18	6 6	20 22	4 4	16 18	4 4	18 20	4 6	12 14	1001 AAA0 1001 AAA1	11ee eeee 11ee eeee	An s •An	- - - - -
SUBI Subtract Immediate	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
SUBQ Subtract Quick	B W L	s-Imm3 d s:Imm3 d	2 2	4 8	2 • 2	4 8	2 2	12 16	2 2	12 16	2 2	14 22	4 4	16 24	4 4	18 26	4 4	16 24	6 6	20 28							0101 QQQ1	SSFF FFFF	d = •d	* * * * *
SUBX Subtract Multiprecision	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
SWAP Swap Regis- ter Halves	W	d	2	4																							0100 1000	0100 0DDD	Dn(31 16)• • Dn(15 0)	- * * 0 0
TAS Test and Set Operand	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
TST Test	B W L	d d-	2 2	4 4			2 2	8 12	2 2	8 12	2 2	10 14	4 4	12 16	4 4	14 18	4 4	12 16	6 6	16 20							0100 1010	SSFF FFFF	test d •cc	- * * 0 0
UNLK Unlink	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Bcc Branch Conditionally	B W	disp disp																						bra taken bra not taken bra taken bra not taken	2 2 4 4	10 8 10 14	0110 CCCC	PPPP PPPP	if cc true PC • disp •PC	- - - - -
BRA Branch Always	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
BSR Branch to Subroutine	B W	disp disp																							2 4	20 20	0110 0001	PPPP PPPP	PC • (SP) PC • disp •PC	- - - - -
DBcc Decrement Counter & Branch Until Condition True or Count = 1	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
JMP Jump to		d-					2	8					4	10	4	14	4	10	6	12	4	10	4	14			0100 1110	11FF FFFE	d •PC	- - - - -
JSR Jump to Subroutine	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
NOP No Operation			2	4																							0100 1110	0111 0001	none	- - - - -
RESET Reset Exter- nal Devices	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
RTE Return from Exception			2	20																							0100 1110	0111 0011	(SP) • •SR (SP) • •PC	* * * * *
RTR Return from Subroutine/ Restore OC	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
RTS Return from Subroutine			2	16																							0100 1110	0111 0101	(SP) + •PC	- - - - -
STOP Load SR/Stop	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
TRAP Trap			2	34																							0100 1110	0100 VVVV	PC → (SSP) SR→ (SSP) (Vector) •PC	- - - - -
TRAPV Trap if Overflow Set	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

Note Refer to"Condition Code Computations
as for condition Code
• Word only
<. Maximum value
#. Number of Program Bytes
~. Number of Clock Periods

Opcode Bit Pattern Key

A. Address Register #
C. Test Condition
D Data Register #
e Source Effective Address
E Destination Effective Address

f Direction 0—Right 1—Left
M Destination EA Mode
P Displacement
Q Quick Immediate Data
r Source Register

R Destination Register
S Size 00 Byte
01 Word
10—Long Word
11 Another Operation
V Vector #

(In the MOVE Instruction
01 Byte
10 Long Word
11 Word)

4

■ INSTRUCTION FORMAT SUMMARY

This provides a summary of the first word in each instruction of the instruction set. Table 26 is an operation code (op-code) map which illustrates how bits 15 through 12 are used to specify the operations. The remaining paragraph groups the instructions according to the op-code map.

where, Size; Byte = 00 Sz; Word = 0
Word = 01 Long Word = 1
Long Word = 10

Table 26 Operation Code Map

Bits 15 thru 12	Operation
0000	Bit Manipulation/MOVEP/Immediate
0001	Move Byte
0010	Move Long
0011	Move Word
0100	Miscellaneous
0101	ADDQ/SUBQ/S_{CC}/DB_{CC}
0110	B_{CC}
0111	MOVEQ
1000	OR/DIV/SBCD
1001	SUB/SUBX
1010	(Unassigned)
1011	CMP/EOR
1100	AND/MUL/ABCD/EXG
1101	ADD/ADDX
1110	Shift/Rotate
1111	(Unassigned)

(1) BIT MANIPULATION, MOVE PERIPHERAL, IMMEDIATE INSTRUCTIONS

Dynamic Bit

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	Register			1	Type		Effective Address					

Static Bit

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	0	0	0	Type		Effective Address					

Bit Type Codes: TST = 00, CHG = 01, CLR = 10, SET = 11

MOVEP

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	Register			Op-Mode			0	0	1	Register		

Op-Mode, Word to Reg = 100, Long to Reg = 101, Word to Mem = 110, Long to Mem = 111

OR Immediate

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	Size		Effective Address					

AND Immediate

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	1	0	Size		Effective Address					

SUB Immediate

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	1	0	0	Size		Effective Address					

ADD Immediate

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	1	1	0	Size		Effective Address					

EOR Immediate

15	14	13	12	11	10	9	8	7	6	5	4'	3	2	1	0
0	0	0	0	1	0	1	0	Size		Effective Address					

CMP Immediate

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	1	0	0	Size		Effective Address					

(2) MOVE BYTE INSTRUCTION

MOVE Byte

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	Destination						Source					
				Register			Mode			Mode			Register		

(3) MOVE LONG INSTRUCTION

MOVE Long

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	Destination						Source					
				Register			Mode			Mode			Register		

(4) MOVE WORD INSTRUCTION

MOVE Word

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	Destination						Source					
				Register			Mode			Mode			Register		

(5) MISCELLANEOUS INSTRUCTIONS

NEGX

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	0	0	0	0	Size		Effective Address					

MOVE from SR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	0	0	0	0	1	1	Effective Address					

CLR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	0	0	1	0	Size		Effective Address					

NEG

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	0	1	0	0	Size		Effective Address					

MOVE to CCR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	0	1	0	0	1	1	Effective Address					

NOT

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	0	1	1	0	Size		Effective Address					

MOVE to SR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	0	1	1	0	1	1	Effective Address					

NBCD

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	0	0	0	0	0	Effective Address					

PEA

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	0	0	0	0	1	Effective Address					

SWAP

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	0	0	0	0	1	0	0	0	Register		

MOVEM Registers to EA

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	0	0	0	1	Sz	Effective Address					

EXTW

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	0	0	0	1	0	0	0	0	Register		

EXTL

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	0	0	0	1	1	0	0	0	Register		

TST

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	0	1	0	Size		Effective Address					

TAS

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	0	1	0	1	1	Effective Address					

MOVEM EA to Registers

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	0	0	1	Sz	Effective Address					

TRAP

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	0	0	Vector			

LINK

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	0	1	0	Register		

UNLK

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	0	1	1	Register		

MOVE to USP

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	1	0	0	Register		

MOVE from USP

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	1	0	1	Register		

RESET

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	1	1	0	0	0	0

NOP

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	1	1	0	0	0	1

STOP

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	1	1	0	0	1	0

RTE

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	1	1	0	0	1	1

RTS

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	1	1	0	1	0	1

TRAPV

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	1	1	0	1	1	0

RTR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	1	1	0	1	1	1

JSR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	1	0	Effective Address					

JMP

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	1	1	Effective Address					

CHK

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	Register			1	1	0	Effective Address					

LEA

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	Register			1	1	1	Effective Address					

(6) ADD QUICK, SUBTRACT QUICK, SET CONDITIONALLY, DECREMENT INSTRUCTIONS

ADDQ

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	Data			0	Size		Effective Address					

SUBQ

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	Data			1	Size		Effective Address					

S$_{CC}$

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	Condition				1	1	Effective Address					

DB$_{CC}$

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	Condition				1	1	0	0	1	Register		

(7) BRANCH CONDITIONALLY, BRANCH TO SUBROUTINE INSTRUCTION

B$_{CC}$

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	Condition				8 bit Displacement							

BSR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	0	0	0	1	8 bit Displacement							

(8) MOVE QUICK INSTRUCTION

MOVEQ

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	Register			0	Data							

(9) OR, DIVIDE, SUBTRACT DECIMAL INSTRUCTIONS

OR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	Register			Op-Mode			Effective Address					

Op-Mode			
B	W	L	
000	001	010	Dn ∨ EA → Dn
100	101	110	EA ∨ Dn → EA

DIVU

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	Register			0	1	1	Effective Address					

DIVS

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	Register			1	1	1	Effective Address					

SBCD

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	Destination Register			1	0	0	0	0	R/M	Source Register		

R/M (register/memory): register — register = 0, memory — memory = 1

(10) SUBTRACT, SUBTRACT EXTENDED INSTRUCTIONS

SUB

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	1	Register			Op-Mode			Effective Address					

Op-Mode			
B	W	L	
000	001	010	Dn—EA → Dn
100	101	110	EA—Dn → EA
—	011	111	An—EA → An

SUBX

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	1	Destination Register			1	Size		0	0	R/M	Source Register		

R/M (register/memory): register — register = 0, memory — memory = 1

(11) COMPARE, EXCLUSIVE OR INSTRUCTIONS

CMP

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	Register			Op-Mode			Effective Address					

Op-Mode			
B	W	L	
000	001	010	Dn—EA
—	011	111	An—EA

CMPM

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	Register			1	Size		0	0	1	Register		

EOR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	Register			1	Size		Effective Address					

(12) AND, MULTIPLY, ADD DECIMAL, EXCHANGE INSTRUCTIONS

AND

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	Register			Op-Mode			Effective Address					

Op-Mode			
B	W	L	
000	001	010	Dn ∧ EA → Dn
100	101	110	EA ∧ Dn → EA

MULU

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	Register			0	1	1	Effective Address					

MULS

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	Register			1	1	1	Effective Address					

ABCD

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	Destination Register			1	0	0	0	0	R/M	Source Register		

R/M (register/memory): register — register = 0, memory — memory = 1

EXGD

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	Data Register			1	0	1	0	0	0	Data Register		

EXGA

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	Address Register			1	0	1	0	0	1	Address Register		

EXGM

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	Data Register			1	1	0	0	0	1	Address Register		

(13) ADD, ADD EXTENDED INSTRUCTIONS

ADD

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	1	Register			Op-Mode			Effective Address					

	Op-Mode		
B	W	L	
000	001	010	Dn + EA → Dn
100	101	110	EA + Dn → EA
–	011	111	An + EA → An

ADDX

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	1	Destination Register			1	Size		0	0	R/M	Source Register		

R/M (register/memory): register — register = 0, memory — memory = 1

(14) SHIFT/ROTATE INSTRUCTIONS

Data Register Shifts

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	0	Count/Register			d	Size		i/r	Type		Register		

Memory Shifts

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	0	0	Type		d	1	1	Effective Address					

Shift Type Codes: AS = 00, LS = 01, ROX = 10, RO = 11
d (direction): Right = 0, Left = 1
i/r (count source): Immediate Count = 0, Register Count = 1

■ INSTRUCTION EXECUTION TIMES

The following paragraphs contain listings of the instruction execution times in terms of external clock (CLK) periods. In this timing data, it is assumed that both memory read and write cycle times are four clock periods. Any wait states caused by a longer memory cycle must be added to the total instruction time. The number of bus read and write cycles for each instruction is also included with the timing data. This data is enclosed in parenthesis following the execution periods and is shown as: (r/w) where r is the number of read cycles and w is the number of write cycles.

(NOTE) The number of periods includes instruction fetch and all applicable operand fetches and stores.

● EFFECTIVE ADDRESS OPERAND CALCULATION TIMING

Table 27 lists the number of clock periods required to compute an instruction's effective address. It includes fetching of any extension words, the address computation, and fetching of the memory operand. The number of bus read and write cycles is shown in parenthesis as (r/w). Note there are no write cycles involved in processing the effective address.

● MOVE INSTRUCTION CLOCK PERIODS

Table 28 and 29 indicate the number of clock periods for the move instruction. This data includes instruction fetch, operand reads, and operand writes. The number of bus read and write cycles is shown in parenthesis as: (r/w).

● STANDARD INSTRUCTION CLOCK PERIODS

The number of clock periods shown in Table 30 indicates the time required to perform the operations, store the results, and read the next instruction. The number of bus read and write cycles is shown in parenthesis as: (r/w). The number of clock periods and the number of read and write cycles must be added respectively to those of the effective address calculation where indicated.

In Table 30 the headings have the following meanings: An = address register operand, Dn = data register operand, ea = an operand specified by an effective address, and M = memory effective address operand.

● IMMEDIATE INSTRUCTION CLOCK PERIODS

The number of clock periods shown in Table 31 includes the time to fetch immediate operands, perform the operations, store the results, and read the next operation. The number of bus read and write cycles is shown in parenthesis as: (r/w). The number of clock periods and the number of read and write cycles must be added respectively to those of the effective address calculation where indicated.

In Table 31, the headings have the following meanings: # = immediate operand, Dn = data register operand, An = address register operand, M = memory operand, CCR = condition code register, and SR = status register.

● SINGLE OPERAND INSTRUCTION CLOCK PERIODS

Table 32 indicates the number of clock periods for the single operand instructions. The number of bus read and write cycles is shown in parenthesis as: (r/w). The number of clock periods and the number of read and write cycles must be added respectively to those of the effective address calculation where indicated.

Table 27 Effective Address Calculation Timing

Addressing Mode		Byte, Word	Long
	Register		
Dn	Data Register Direct	0(0/0)	0(0/0)
An	Address Register Direct	0(0/0)	0(0/0)
	Memory		
An@	Address Register Indirect	4(1/0)	8(2/0)
An@ +	Address Register Indirect with Postincrement	4(1/0)	8(2/0)
An@ -	Address Register Indirect with Predecrement	6(1/0)	10(2/0)
An@(d)	Address Register Indirect with Displacement	8(2/0)	12(3/0)
An@(d, ix)*	Address Register Indirect with Index	10(2/0)	14(3/0)
xxx.W	Absolute Short	8(2/0)	12(3/0)
xxx.L	Absolute Long	12(3/0)	16(4/0)
PC@(d)	Program Counter with Displacement	8(2/0)	12(3/0)
PC@(d, ix)*	Program Counter with Index	10(2/0)	14(3/0)
#xxx	Immediate	4(1/0)	8(2/0)

* The size of the index register (ix) does not affect execution time.

Table 28 Move Byte and Word Instruction Clock Periods

Source	Destination								
	Dn	An	An@	An@ +	An@ -	An@(d)	An@(d, ix)*	xxx.W	xxx.L
Dn	4(1/0)	4(1/0)	8(1/1)	8(1/1)	8(1/1)	12(2/1)	14(2/1)	12(2/1)	16(3/1)
An	4(1/0)	4(1/0)	8(1/1)	8(1/1)	8(1/1)	12(2/1)	14(2/1)	12(2/1)	16(3/1)
An@	8(2/0)	8(2/0)	12(2/1)	12(2/1)	12(2/1)	16(3/1)	18(3/1)	16(3/1)	20(4/1)
An@ +	8(2/0)	8(2/0)	12(2/1)	12(2/1)	12(2/1)	16(3/1)	18(3/1)	16(3/1)	20(4/1)
An@ -	10(2/0)	10(2/0)	14(2/1)	14(2/1)	14(2/1)	18(3/1)	20(3/1)	18(3/1)	22(4/1)
An@(d)	12(3/0)	12(3/0)	16(3/1)	16(3/1)	16(3/1)	20(4/1)	22(4/1)	20(4/1)	24(5/1)
An@(d, ix)*	14(3/0)	14(3/0)	18(3/1)	18(3/1)	18(3/1)	22(4/1)	24(4/1)	22(4/1)	26(5/1)
xxx.W	12(3/0)	12(3/0)	16(3/1)	16(3/1)	16(3/1)	20(4/1)	22(4/1)	20(4/1)	24(5/1)
xxx.L	16(4/0)	16(4/0)	20(4/1)	20(4/1)	20(4/1)	24(5/1)	26(5/1)	24(5/1)	28(6/1)
PC@(d)	12(3/0)	12(3/0)	16(3/1)	16(3/1)	16(3/1)	20(4/1)	22(4/1)	20(4/1)	24(5/1)
PC@(d, ix)*	14(3/0)	14(3/0)	18(3/1)	18(3/1)	18(3/1)	22(4/1)	24(4/1)	22(4/1)	26(5/1)
#xxx	8(2/0)	8(2/0)	12(2/1)	12(2/1)	12(2/1)	16(3/1)	18(3/1)	16(3/1)	20(4/1)

* The size of the index register (ix) does not affect execution time

Table 29 Move Long Instruction Clock Periods

Source	Destination								
	Dn	An	An@	An@ +	An@ -	An@(d)	An@(d, ix)*	xxx.W	xxx.L
Dn	4(1/0)	4(1/0)	12(1/2)	12(1/2)	12(1/2)	16(2/2)	18(2/2)	16(2/2)	20(3/2)
An	4(1/0)	4(1/0)	12(1/2)	12(1/2)	12(1/2)	16(2/2)	18(2/2)	16(2/2)	20(3/2)
An@	12(3/0)	12(3/0)	20(3/2)	20(3/2)	20(3/2)	24(4/2)	26(4/2)	24(4/2)	28(5/2)
An@ +	12(3/0)	12(3/0)	20(3/2)	20(3/2)	20(3/2)	24(4/2)	26(4/2)	24(4/2)	28(5/2)
An@ -	14(3/0)	14(3/0)	22(3/2)	22(3/2)	22(3/2)	26(4/2)	28(4/2)	26(4/2)	30(5/2)
An@(d)	16(4/0)	16(4/0)	24(4/2)	24(4/2)	24(4/2)	28(5/2)	30(5/2)	28(5/2)	32(6/2)
An@(d, ix)*	18(4/0)	18(4/0)	26(4/2)	26(4/2)	26(4/2)	30(5/2)	32(5/2)	30(5/2)	34(6/2)
xxx.W	16(4/0)	16(4/0)	24(4/2)	24(4/2)	24(4/2)	28(5/2)	30(5/2)	28(5/2)	32(6/2)
xxx.L	20(5/0)	20(5/0)	28(5/2)	28(5/2)	28(5/2)	32(6/2)	34(6/2)	32(6/2)	36(7/2)
PC@(d)	16(4/0)	16(4/0)	24(4/2)	24(4/2)	24(4/2)	28(5/2)	30(5/2)	28(5/2)	32(6/2)
PC@(d, ix)*	18(4/0)	18(4/0)	26(4/2)	26(4/2)	26(4/2)	30(5/2)	32(5/2)	30(5/2)	34(6/2)
#xxx	12(3/0)	12(3/0)	20(3/2)	20(3/2)	20(3/2)	24(4/2)	26(4/2)	24(4/2)	28(5/2)

* The size of the index register (ix) does not affect execution time

Table 30 Standard Instruction Clock Periods

Instruction	Size	op < ea >, An	op < ea >, Dn	op Dn, < M >
ADD	Byte, Word	8(1/0) +	4(1/0) +	8(1/1) +
	Long	6(1/0) + **	6(1/0) + **	12(1/2) +
AND	Byte, Word	–	4(1/0) +	8(1/1) +
	Long	–	6(1/0) + **	12(1/2) +
CMP	Byte, Word	6(1/0) +	4(1/0) +	–
	Long	6(1/0) +	6(1/0) +	–
DIVS	–	–	158(1/0) + *	–
DIVU	–	–	140(1/0) + *	–
EOR	Byte, Word	–	4(1/0) ***	8(1/1) +
	Long	–	8(1/0) ***	12(1/2) +
MULS	–	–	70(1/0) + *	–
MULU	–	–	70(1/0) + *	–
OR	Byte, Word	–	4(1/0) +	8(1/1) +
	Long	–	6(1/0) + **	12(1/2) +
SUB	Byte, Word	8(1/0) +	4(1/0) +	8(1/1) +
	Long	6(1/0) + **	6(1/0) + **	12(1/2) +

+ add effective address calculation time ** total of 8 clock periods for instruction if the effective address is register direct

* indicates maximum value *** only available effective address mode is data register direct

DIVS, DIVU – The divide algorithm used by the 68000 provides less than 10% difference between the best and worst case timings.

MULS, MULU – The multiply algorithm requires 38+2n clocks when n is defined as

MULU; n = the number of ones in the < ea >

MULS; n = concatanate the < ea > with a zero as the LSB; n is the resultant number of 10 or 01 patterns in the 17-bit source, i.e worst case happens when the source is $5555.

Table 31 Immediation Instruction Clock Periods

Instruction	Size	op #, Dn	op #, An	op #, M	op #, CCR/SR
ADDI	Byte, Word	8(2/0)	–	12(2/1) +	–
	Long	16(3/0)	–	20(3/2) +	–
ADDQ	Byte, Word	4(1/0)	8(1/0)*	8(1/1) +	–
	Long	8(1/0)	8(1/0)	12(1/2) +	–
ANDI	Byte, Word	8(2/0)	–	12(2/1) +	20(3/0)
	Long	16(3/0)	–	20(3/1) +	–
CMPI	Byte, Word	8(2/0)	8(2/0)	8(2/0) +	–
	Long	14(3/0)	14(3/0)	12(3/0) +	–
EORI	Byte, Word	8(2/0)	–	12(2/1) +	20(3/0)
	Long	16(3/0)	–	20(3/2) +	–
MOVEQ	Long	4(1/0)	–	–	–
ORI	Byte, Word	8(2/0)	–	12(2/1) +	20(3/0)
	Long	16(3/0)	–	20(3/2) +	–
SUBI	Byte, Word	8(2/0)	–	12(2/1) +	–
	Long	16(3/0)	–	20(3/2) +	–
SUBQ	Byte, Word	4(1/0)	8(1/0)*	8(1/1) +	–
	Long	8(1/0)	8(1/0)	12(1/2) +	–

+ add effective address calculation time
* word only

Table 32 Single Operand Instruction Clock Periods

Instruction	Size	Register	Memory
CLR	Byte, Word	4(1/0)	8(1/1) +
	Long	6(1/0)	12(1/2) +
NBCD	Byte	6(1/0)	8(1/1) +
NEG	Byte, Word	4(1/0)	8(1/1) +
	Long	6(1/0)	12(1/2) +
NEGX	Byte, Word	4(1/0)	8(1/1) +
	Long	6(1/0)	12(1/2) +
NOT	Byte, Word	4(1/0)	8(1/1) +
	Long	6(1/0)	12(1/2) +
S_{CC}	Byte, False	4(1/0)	8(1/1) +
	Byte, True	6(1/0)	8(1/1) +
TAS	Byte	4(1/0)	10(1/1) +
TST	Byte, Word	4(1/0)	4(1/0) +
	Long	4(1/0)	4(1/0) +

+ add effective address calculation time

- **SHIFT/ROTATE INSTRUCTION CLOCK PERIODS**

Table 33 indicates the number of clock periods for the shift and rotate instructions. The number of bus read and write cycles is shown in parenthesis as: (r/w). The number of clock periods and the number of read and write cycles must be added respectively to those of the effective address calculation where indicated.

- **BIT MANIPULATION INSTRUCTION CLOCK PERIODS**

Table 34 indicates the number of clock periods required for the bit manipulation instructions. The number of bus read and write cycles is shown in parenthesis as: (r/w). The number of clock periods and the number of read and write cycles must be added respectively to those of the effective address calculation where indicated.

- **CONDITIONAL INSTRUCTION CLOCK PERIODS**

Table 35 indicates the number of clock periods required for the conditional instructions. The number of bus read and write cycles is indicated in parenthesis as: (r/w). The number of clock periods and the number of read and write cycles must be added respectively to those of the effective address calculation where indicated.

- **JMP, JSR, LEA, PEA, MOVEM INSTRUCTION CLOCK PERIODS**

Table 36 indicates the number of clock periods required for the jump, jump to subroutine, load effective address, push effective address, and move multiple registers instructions. The number of bus read and write cycles is shown in parenthesis as: (r/w).

Table 33 Shift/Rotate Instruction Clock Periods

Instruction	Size	Register	Memory
ASR, ASL	Byte, Word	6 + 2n(1/0)	8(1/1) +
	Long	8 + 2n(1/0)	–
LSR, LSL	Byte, Word	6 + 2n(1/0)	8(1/1) +
	Long	8 + 2n(1/0)	–
ROR, ROL	Byte, Word	6 + 2n(1/0)	8(1/1) +
	Long	8 + 2n(1/0)	–
ROXR, ROXL	Byte, Word	6 + 2n(1/0)	8(1/1) +
	Long	8 + 2n(1/0)	–

Table 34 Bit Manipulation Instruction Clock Periods

Instruction	Size	Dynamic		Static	
		Register	Memory	Register	Memory
BCHG	Byte	–	8(1/1) +	–	12(2/1) +
	Long	8(1/0)*	–	12(2/0)*	–
BCLR	Byte	–	8(1/1) +	–	12(2/1) +
	Long	10(1/0)*	–	14(2/0)*	–
BSET	Byte	–	8(1/1) +	–	12(2/1) +
	Long	8(1/0)*	–	12(2/0)*	–
BTST	Byte	–	4(1/0) +	–	8(2/0) +
	Long	6(1/0)	–	10(2/0)	–

+ add effective address calculation time
* indicates maximum value

Table 35 Conditional Instruction Clock Periods

Instruction	Displacement	Trap or Branch Taken	Trap of Branch Not Taken
B_{CC}	Byte	10(2/0)	8(1/0)
	Word	10(2/0)	12(2/0)
BRA	Byte	10(2/0)	–
	Word	10(2/0)	–
BSR	Byte	18(2/2)	–
	Word	18(2/2)	–
DB_{CC}	CC true	–	12(2/0)
	CC false	10(2/0)	14(3/0)
CHK	–	40(5/3) + *	10(1/0) +
TRAP	–	34(4/3)	–
TRAPV	–	34(5/3)	4(1/0)

+ add effective address calculation time
* indicates maximum value

Table 36 JMP, JSR, LEA, PEA, MOMEM Instruction Clock Periods

Instr	Size	An@	An@ +	An@ -	An@(d)	An@(d, ix)*	xxx. W	xxx. L	PC@(d)	PC@(d, ix)*
JMP	–	8(2/0)	–	–	10(2/0)	14(3/0)	10(2/0)	12(3/0)	10(2/0)	14(3/0)
JSR	–	16(2/2)	–	–	18(2/2)	22(2/2)	18(2/2)	20(3/2)	18(2/2)	22(2/2)
LEA	–	4(1/0)	–	–	8(2/0)	12(2/0)	8(2/0)	12(3/0)	8(2/0)	12(2/0)
PEA	–	12(1/2)	–	–	16(2/2)	20(2/2)	16(2/2)	20(3/2)	16(2/2)	20(2/2)
MOVEM	Word	12+4n (3+n/0)	12+4n (3+n/0)	– –	16+4n (4+n/0)	18+4n (4+n/0)	16+4n (4+n/0)	20+4n (5+n/0)	16+4n (4+n/0)	18+4n (4+n/0)
M → R	Long	12+8n (3+2n/0)	12+8n (3+2n/0)	– –	16+8n (4+2n/0)	18+8n (4+2n/0)	16+8n (4+2n/0)	20+8n (5+2n/0)	16+8n (4+2n/0)	18+8n (4+2n/0)
MOVEM	Word	8+4n (2/n)	– –	8+4n (2/n)	12+4n (3/n)	14+4n (3/n)	12+4n (3/n)	16+4n (4/n)	– –	– –
R → M	Long	8+8n (2/2n)	– –	8+8n (2/2n)	12+8n (3/2n)	14+8n (3/2n)	12+8n (3/2n)	16+8n (4/2n)	– –	– –

n is the number of registers to move

* is the size of the index register (ix) does not affect the instruction's execution time

• MULTI-PRECISION INSTRUCTION CLOCK PERIODS

Table 37 indicates the number of clock periods for the multi-precision instructions. The number of clock periods includes the time to fetch both operands, perform the operations, store the results, and read the next instructions. The number of read and write cycles is shown in parenthesis as: (r/w).

In Table 37, the headings have the following meanings. Dn = data register operand and M = memory operand.

Table 37 Multi-Precision Instruction Clock Periods

Instruction	Size	op Dn, Dn	op M, M
ADDX	Byte, Word	4(1/0)	18(3/1)
	Long	8(1/0)	30(5/2)
CMPM	Byte, Word	–	12(3/0)
	Long	–	20(5/0)
SUBX	Byte, Word	4(1/0)	18(3/1)
	Long	8(1/0)	30(5/2)
ABCD	Byte	6(1/0)	18(3/1)
SBCD	Byte	6(1/0)	18(3/1)

• MISCELLANEOUS INSTRUCTION CLOCK PERIODS

Table 38 indicates the number of clock periods for the following miscellaneous instructions. The number of bus read and write cycles is shown in parenthesis as: (r/w). The number of clock periods plus the number of read and write cycles must be added to those of the effective address calculation where indicated.

• EXCEPTION PROCESSING CLOCK PERIODS

Table 39 indicates the number of clock periods for exception processing. The number of clock periods includes the time for all stacking, the vector fetch, and the fetch of the first instruction of the handler routine. The number of bus read and write cycles is shown in parenthesis as: (r/w).

Table 38 Miscellaneous Instruction Clock Periods

Instruction	Size	Register	Memory	Register → Memory	Memory → Register
MOVE from SR	–	6(1/0)	8(1/1) +	–	–
MOVE to CCR	–	12(2/0)	12(2/0) +	–	–
MOVE to SR	–	12(2/0)	12(2/0) +	–	–
MOVEP	Word	–	–	16(2/2)	16(4/0)
	Long	–	–	24(2/4)	24(6/0)
EXG	–	6(1/0)	–	–	–
EXT	Word	4(1/0)	–	–	–
	Long	4(1/0)	–	–	–
LINK	–	16(2/2)	–	–	–
MOVE from USP	–	4(1/0)	–	–	–
MOVE to USP	–	4(1/0)	–	–	–
NOP	–	4(1/0)	–	–	–
RESET	–	132(1/0)	–	–	–
RTE	–	20(5/0)	–	–	–
RTR	–	20(5/0	–	–	–
RTS	–	16(4/0)	–	–	–
STOP	–	4(0/0)	–	–	–
SWAP	–	4(1/0)	–	–	–
UNLK	–	12(3/0)	–	–	–

+ add effective address calculation time

Table 39 Exception Processing Clock Periods

Exception	Periods
Reset**	38.5 (6/0)
Address Error	50(4/7)
Bus Error	50(4/7)
Interrupt	44(5/3)*
Illegal Instruction	34(4/3)
Privileged Violation	34(4/3)
Trace	34(4/3)

* The interrupt acknowledge bus cycle is assumed to take four external clock periods.

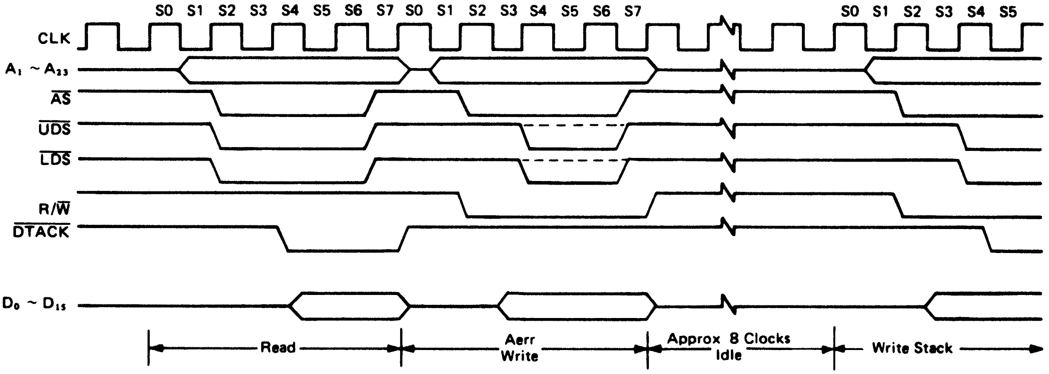
** Indicates the time from when RES and HALT are first sampled as negated to when instruction execution starts.

■ **MASK VERSION**

Type No.	Mask version
HD68000-8 HD68000-10 HD68000-12	68000S1
HD68000Y-8 HD68000Y-10 HD68000Y-12	
HD68000P8 HD68000PS8 HD68000CP8	68000U

The difference of function between mask version 68000S1 and 68000U is only as following (Figure 59).

The function of HD68HC000 is as same as mask version 68000U.



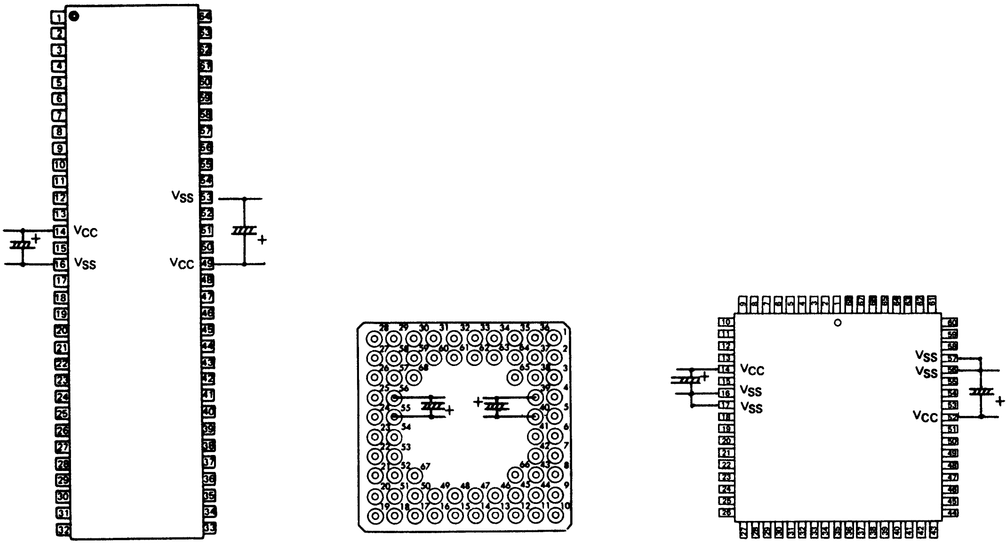
Broken line: mask version 68000U and HD68HC000.

Figure 59 Address Error Timing

■ **NOTE FOR USE**

● **Power Supply Circuit**

When designing V_{CC} and V_{SS} pattern of the circuit board, the capacitors need to be located nearest to V_{CC} and V_{SS} as shown in the Figure 60.



(Top View)
(a) DIP

(Bottom View)
(b) PGA

(Top View)
(c) PLCC

1μF/35V Tantalum Capacitor (2 pairs)

Figure 60 Power Supply Circuit

Hitachi America, Ltd.

SEMICONDUCTOR & I.C. DIVISION
Hitachi Plaza
2000 Sierra Point Parkway
Brisbane, CA 94005–1819
Telephone: 415–589–8300
Twx: 910-338-2103
Fax: 415–583–4207

REGIONAL OFFICES

NORTHEAST REGION

Hitachi America, Ltd.
77 South Bedford Street
Burlington, MA 01803
Telephone: 617-229-2150
Fax: 617-229-6554

NORTH CENTRAL REGION

Hitachi America, Ltd.
500 Park Boulevard, Suite 415
Itasca, IL 60143
Telephone: 708-773-4864
Fax: 708-773-9006

NORTHWEST REGION

Hitachi America, Ltd.
1900 McCarthy Boulevard, Suite 310
Milpitas, CA 95035
Telephone: 408-954-8100
Fax: 408-954-0499

SOUTHEAST REGION

Hitachi America, Ltd.
5511 Capital Center Drive, Suite 204
Raleigh, NC 27606
Telephone: 919-233-0800
Fax: 919-233-050

SOUTH CENTRAL REGION

Hitachi America, Ltd.
Two Lincoln Centre, Suite 865
5420 LBJ Freeway
Dallas, TX 75240
Telephone: 214-991-4510
Fax: 214-991-6151

SOUTHWEST REGION

Hitachi America, Ltd.
2030 Main Street, Suite 450
Irvine, CA 92714
Telephone: 714-553-8500
Fax: 714-553-8561

AUTOMOTIVE REGION

Hitachi America, Ltd.
330 Town Center Drive, Suite 311
Dearborn, MI 48126
Telephone: 313-271-4410
Fax: 313-271-5707

TELECOM REGION

Hitachi America, Ltd.
325 Columbia Turnpike, Suite 203
Florham Park, NJ 07932
Telephone: 201-514-2100
Fax: 201-514-2020

DISTRICT OFFICES

Hitachi America, Ltd.
3800 W. 80th Street, Suite 1050
Bloomington, MN 55431
Telephone: 612-896-3444
Fax: 612-896-3443

Hitachi America, Ltd.
21 Old Main Street, Suite 104
Fishkill, NY 12524
Telephone: 914-897-3000
Fax: 914-897-3007

Hitachi America, Ltd.
6161 Savoy Drive, Suite 850
Houston, TX 77036
Telephone: 713-974-0534
Fax: 713-974-0587

Hitachi America, Ltd.
4901 N.W. 17th Way, Suite 302
Fort Lauderdale, FL 33309
Telephone: 305-491-6154
Fax: 305-771-7217

Hitachi America, Ltd.
4600 S. Ulster Street, Suite 700
Denver, CO 80237
Telephone: 303-740-6644
Fax: 303-740-6609

Hitachi (Canadian) Ltd.
320 March Road, Suite 602
Kanata, Ontario, Canada K2K 2E3
Telephone: 613-591-1990
Fax: 613-591-1994

MANUFACTURING FACILITY

Hitachi Semiconductor (America) Inc.
6321 East Campus Circle Drive
Irving, TX 75063-2712

ENGINEERING FACILITY

Hitachi Micro Systems, Inc.
180 Rose Orchard Way
San Jose, CA 95134

Technical product or pricing questions can be answered by your nearest Hitachi office.
You may order product literature either by calling your nearest Hitachi office or by calling 1–800–285–1601.

HITACHI®

HITACHI®

Our Standards Set Standards

Hitachi America, Ltd.
Semiconductor & I.C. Division
Hitachi Plaza
2000 Sierra Point Parkway, Brisbane, CA 94005-1819
1-415-589-8300

991/10M/GI/LP/MC
Order Number: M21T132