



This document and the information contained herein are confidential to and the property of Honeywell Information Systems Inc. and are made available only to Honeywell employees for the sole purpose of maintaining Honeywell's products. This document, any copy thereof and the information contained herein shall be maintained in strictest confidence; shall not be copied in whole or in part except as authorized by the employee's manager; and shall not be disclosed or distributed (a) to persons who are not Honeywell employees, or (b) to Honeywell employees for whom such information is not necessary in connection with their assigned responsibilities. Upon request, or when the employee in possession of this document no longer has need for the document for the authorized Honeywell purpose, this document and any copies thereof shall be returned to the employee's manager. There shall be no exceptions to the terms and conditions set forth herein except as authorized in writing by the responsible Honeywell Vice President.

SERIES 60 (LEVEL 6)

TYPE MLC9103 MULTILINE COMMUNICATIONS PROCESSOR MANUAL

Doc. No. 71010230-200 Order No. FL48 Rev. 1

THIS MANUAL HAS BEEN UPDATED TO INCLUDE CHANGE=PAGE PACKAGE -201.

Honeywell

This document and the information contained herein are confidential to and the property of Honeywell Information Systems Inc. and are made available only to Honeywell employees for the sole purpose of maintaining Honeywell's products. This document, any copy thereof and the information contained herein shall be maintained in strictest confidence; shall not be copied in whole or in part except as authorized by the employee's manager; and shall not be disclosed or distributed (a) to persons who are not Honeywell employees, or (b) to Honeywell employees for whom such information is not necessary in connection with their assigned responsibilities. Upon request, or when the employee in possession of this document no longer has need for the document for the authorized Honeywell purpose, this document and any copies thereof shall be returned to the employee's manager. There shall be no exceptions to the terms and conditions set forth herein except as authorized in writing by the responsible Honeywell Vice President.

SERIES 60 (LEVEL 6)

TYPE MLC9103 MULTILINE COMMUNICATIONS PROCESSOR MANUAL

Doc. No. 71010230-200 Order No. FL48 Rev. 1

THIS MANUAL HAS BEEN UPDATED TO INCLUDE CHANGE=PAGE PACKAGE -201.

RECORD OF REVISIONS

REVISION	DATE	AUTHORITY	AFFECTED PAGES
-100	May 1976	--	--
-200	May 1977	BLC060191 BLC060920 BLC061148 BLC061263 BLC061305	--
-201	Mar. 1978	BLC061574 BLC061768 BLC070197 BLC071207 BLC071230 BLC071421	Title, ROR, 1-1, 1-3, 3-39, 3-70, 4-1, 4-10, 4-14, 4-17, 4-18, 4-46

Hardware Publications, M&TO, Billerica, MA 01821

Printed in the United States of America
All rights reserved

FL48

CONTENTS

Section		Page
I	INTRODUCTION	1-1
1.1	Scope and Purpose of This Document	1-1
	1.1.1 Organization of This Document	1-1
	1.1.2 Reference Documents	1-2
1.2	MLCP Description	1-2
1.3	Specifications	1-5
	1.3.1 Line Capabilities	1-5
	1.3.2 Block Mode	1-5
	1.3.3 Line Configuration	1-5
	1.3.3.1 Configuration of Synchronous Lines	1-5
	1.3.3.2 Configuration of Asynchronous Lines	1-5
	1.3.4 Programmable Line Controls	1-6
	1.3.5 Physical Characteristics	1-6
	1.3.6 Environmental Requirements	1-6
	1.3.7 Power Requirements	1-6
1.4	Abbreviations/Definitions	1-8
II	THEORY OF OPERATION - OVERVIEW	2-1
2-1	Interface Description	2-1
	2.1.1 MLCP/Megabus Interface	2-1
	2.1.2 MLCP/CLA Interface	2-1
2-2	MLCP Functional Requirements	2-7
	2.2.1 Software	2-8
	2.2.2 Firmware	2-8
	2.2.3 Hardware	2-11
	2.2.3.1 Line Adapter Interface Control Logic	2-11
	2.2.3.2 Megabus Interface Control Logic	2-11
	2.2.3.3 Read/Write RAM	2-12
	2.2.3.4 Cycle Redundancy Check (CRC) Logic	2-12
	2.2.3.5 Microprocessor and Control Store Logic	2-13

HONEYWELL PROPRIETARY AND CONFIDENTIAL

Section		Page
	2.2.3.6 Hardware-Implemented I/O Command Logic	2-13
2.3	Summary Operational Description	2-14
	2.3.1 Operational Overview	2-14
	2.3.2 Line Setup and Configuration	2-18
	2.3.3 Data Transfer	2-18
	2.3.4 Pause Function	2-19
	2.3.5 Data Transfer Clock	2-19
	2.3.6 Operational States (Figure 2-9)	2-20
	2.3.7 Operational Modes	2-21
	2.3.7.1 Receive Mode	2-21
	2.3.7.2 Transmit Mode	2-22
2.4	Line Control Table (LCT)	2-23
2.5	Communication Control Blocks (CCBs)	2-23
2.6	Channel Control Program (CCP)	2-24
III	THEORY OF OPERATION - INTERMEDIATE	3-1
3.1	Hardware Major Block Diagram	3-1
	3.1.1 Functional Areas	3-1
	3.1.1.1 Microprocessor Control Logic	3-1
	3.1.1.2 Hardware Implemented I/O Command Logic	3-1
	3.1.1.3 Megabus Interface Register Logic	3-1
	3.1.1.4 Read/Write RAM	3-3
	3.1.1.5 Line Adapter Interface Control Logic	3-3
	3.1.1.6 CRC (Block Check) Logic	3-3
	3.1.2 Data Flow and Control Paths	3-3
3.2	Microprocessor and Control Store Logic	3-4
	3.2.1 Microprocessor	3-4
	3.2.1.1 Central Processing Element	3-4
	3.2.1.2 Input and Output Functions	3-5
	3.2.1.3 Microfunction Decoder and Scratch Pad Registers	3-5
	3.2.1.4 M-Bus and I-Bus Inputs (M-, I-, and CRC Mux's)	3-12
	3.2.1.5 Accumulator Register	3-12
	3.2.1.6 A- and B-Multiplexers	3-12
	3.2.1.7 Arithmetic Logic Section and K-Bus	3-12
	3.2.1.8 Memory Address Register and Bus	3-13
	3.2.2 Control Multiplexers	3-15
	3.2.3 Fast Carry Logic	3-15
	3.2.4 Microprogram Control Store	3-15

HONEYWELL PROPRIETARY AND CONFIDENTIAL

Section		Page
3.2.5	Microprogram Control Unit	3-15
	3.2.5.1 MCU Element	3-15
	3.2.5.2 Input and Output Functions	3-20
	3.2.5.3 Next Address Logic	3-20
	3.2.5.4 Flag Logic	3-23
	3.2.5.5 Load Function	3-23
3.2.6	Subroutine Address Buffer/Address Multiplexer	3-24
3.2.7	Control Store Buffer	3-26
3.2.8	Strobe Generation Logic	3-26
3.2.9	Master Clear and QLT Logic	3-27
	3.2.9.1 Soft Initialize	3-27
	3.2.9.2 Hard Initialize	3-27
3.3	Megabus Control and Hardware-Implemented I/O Command Logic	3-30
3.3.1	Firmware and Hardware Implementation of I/O Commands	3-30
	3.3.1.1 Firmware-Implemented Commands	3-31
	3.3.1.2 Hardware-Implemented Commands	3-35
3.3.2	Megabus Control Logic	3-37
	3.3.2.1 Master Cycle Logic	3-37
	3.3.2.2 Slave Response Logic	3-38
	3.3.2.3 Function Code Control Logic	3-42
3.3.3	CCB Address Control Logic	3-43
	3.3.3.1 Loading a Single CCB	3-43
	3.3.3.2 Loading All CCBs in a Line	3-50
3.3.4	Input Status/Input Next Status	3-50
3.3.5	PROM Bit Decoder Logic	3-51
3.4	Megabus Interface Register and Control Logic	3-52
3.4.1	MIR Organization and Control	3-52
3.4.2	MIR Operation When MLCP is Slave	3-53
3.4.3	MIR Operation When MLCP is Master	3-54
3.4.4	Parity Checking and Generation	3-55
	3.4.4.1 Parity Checking - Data Received by MLCP From Megabus	3-55
	3.4.4.2 Parity Generation - Data Sent From MLCP to Megabus	3-62
3.5	Read/Write RAM Logic	3-63
3.5.1	Memory Cycle Requests	3-63
3.5.2	I/O Hardware Requests for Memory	3-63
3.5.3	I/O Firmware Requests for Memory	3-64
3.5.4	System Clock and MLCP Timing	3-68

HONEYWELL PROPRIETARY AND CONFIDENTIAL

Section		Page
3.6	Line Adapter Interface Control Logic	3-73
3.6.1	Line Adapter Presence and Identification	3-75
3.6.2	Service of Lines by Channel Control Program	3-78
3.6.2.1	Transmit Channels	3-78
3.6.2.2	Receive Channels	3-79
3.6.3	Clock Counter/Baud Rate Generator	3-83
3.7	CRC Logic	3-83
IV	THEORY OF OPERATION - CYCLE FLOW	4-1
4.1	MLCP Firmware	4-1
4.2	Firmware Instructions	4-2
4.3	Scratch Pad Locations/Registers	4-2
4.4	Random Access Memory	4-2
4.4.1	Line Control Tables (LCT)	4-6
4.4.2	Channel Control Program (CCP)	4-13
4.4.3	Communication Control Blocks	4-19
4.5	Intermediate Flow Charts	4-21
4.5.1	Intermediate Flow Chart Symbology	4-21
4.5.2	Flow Chart Organization	4-22
4.5.3	Flow Chart Usage	4-23
4.6	Firmware Cycle Flow	4-28
4.6.1	MLCP Overview Flow Chart	4-28
4.6.2	Firmware Flow Charts	4-29
4.6.2.1	Quality Logic Test (QLT)	4-29
4.6.2.2	Service Scan Routine	4-30
4.6.2.3	Channel Scanner Routine	4-30
4.6.2.4	Instruction Fetch/Execution Routines	4-30
4.6.2.5	I/O Command Decode/Execution Routines	4-31
4.6.2.6	DMA Service/Execution Routines	4-31
4.6.2.7	Deferred Interrupt Handling Routines	4-32
4.6.2.8	Data Set Scan Routine	4-33

HONEYWELL PROPRIETARY AND CONFIDENTIAL

ILLUSTRATIONS

Figure		Page
1-1	Multiline Communications Processor (MLCP) In Series 60 Level 6 System Configuration	1-4
1-2	MLCP Layout and Dimensions	1-7
2-1	MLCP/Megabus Interface	2-4
2-2	MLCP/Line Adapter Interface	2-5
2-3	MLCP Functional Components	2-9
2-4	Megabus Line Configuration for MLCP Instructions	2-10
2-5	MLCP Hardware Major Functional Components	2-11
2-6	Read/Write RAM Organization	2-13
2-7	MLCP Transmit Functions	2-16
2-8	MLCP Receive Functions	2-17
2-9	MLCP Operational States	2-21
3-1	MLCP Hardware Major Block Diagram	3-2
3-2	Microprocessor and Control Store Logic Functional Block Diagram	3-6
3-3	Microprocessor Intermediate Block Diagram	3-7
3-4	Microprocessor CPE Chip Functional Block Diagram	3-8
3-5	Fast Carry Logic	3-17
3-6	Control Store Word Format	3-19
3-7	Control Store Logic	3-19
3-8	Microprogram Control Unit Functional Block Diagram	3-20
3-9	Microprogram Control Unit (MCU) Chip Functional Block Diagram	3-21
3-10	Subroutine Address Buffer/Address Multiplexer Logic	3-25
3-11	Control Store Buffer Logic	3-26
3-12	Strobe Generation Logic	3-28
3-13	Master Clear and QLT Logic	3-29
3-14	Megabus Control and Hardware-Implementation of I/O Instructions	3-32
3-15	Hardware and Firmware Implementations	3-33
3-16	Master Cycle Logic	3-39
3-17	ACK/NAK/Wait/Respond Logic	3-40
3-18	Busy Logic	3-41
3-19	Function Code Control Logic	3-42
3-20	CCB Address Control Logic	3-44
3-21	PROM Bit Decoder Logic	3-51
3-22	Megabus Interface Register and Control Logic	3-56
3-23	Megabus Interface Register (MIR) and B-Mux Functionality	3-57
3-24	MIR (MLCP is Master Sending Data to Megabus)	3-58
3-25	MIR (MLCP is Slave Receiving Data from Megabus)	3-59
3-26	Content of MIR During an Output (IOLD Output Address) Instruction	3-60
3-27	Content of MIR During an Input (Input ID) Instruction	3-61
3-28	Parity Checking and Generation Logic	3-62

HONEYWELL PROPRIETARY AND CONFIDENTIAL

Figure		Page
3-29	R/W RAM Logic Functional Block Diagram	3-65
3-30	R/W RAM Address Control Logic	3-66
3-31	Request Memory/Start Memory Cycle Logic	3-67
3-32	System Check	3-69
3-33	MLCP Timing Diagram	3-71
3-34	Line Adapter Interface Logic	3-74
3-35	LACOD/LAHERE Multiplexer Selection	3-77
3-36	Line Adapter Request Priority Logic	3-80
3-37	Addressing the Line Adapter	3-81
3-38	CRC Logic	3-85
3-39	Character Size Logic	3-86
3-40	CRC-16 Accumulation	3-87
3-41	CCITT Accumulation	3-88
3-42	CRC-12 Accumulation	3-89
3-43	LRC-8 Accumulation	3-90
4-1	Register 0 Bit Significance (Copy of LCT 21/43)	4-3
4-2	Register 1 Bit Significance in Block Mode (Copy of LCT 5/37)	4-4
4-3	Register 1 Bit Significance During Instruction Execution	4-4
4-4	Channel Number Assignments Within Register 6 and Register 7	4-5
4-5	Channel Control Program Counter Arrangement Within Register 8 and Register 9	4-5
4-6	Random Access Memory Segmentation	4-6
4-7	LCT Channel Command Byte LCT 8/40 Bit Significance	4-10
4-8	LCT Channel Control Byte Bit Significance	4-10
4-9	LCT CLA Status Byte Bit Significance	4-11
4-10	LCT Status Byte 1 Bit Significance	4-11
4-11	LCT Status Byte 2 Bit Significance	4-12
4-12	LCT CLA Control Byte Bit Significance	4-12
4-13	Eight-Byte CCB Format	4-20
4-14	CCB Control Byte Bit Significance	4-21
4-15	MLCP Firmware Overview Flow Chart	4-35
4-16	Quality Logic Test/Service Scan/Channel Scanner Routines	4-37
4-17	Line Adapter Ready/Start I/O/Form Four-Bit Channel Number for I/O Routines	4-39
4-18	I/O Command Decode/Output Channel Control/Start Write/Read Routines	4-40
4-19	Input LCT Byte/Input Data Set Status/Output Interrupt Control/Output LCT Bytes/Set Start I/O in LCT 9/Input Extended Device ID Number Routines	4-41
4-20	Initialize Channel Routine	4-43
4-21	Stop I/O/CCB List Reset Routines	4-44
4-22	Instruction Fetch Routine	4-45
4-23	Generic Instruction Routines	4-46
4-24	Next Character Instruction Routines	4-49

HONEYWELL PROPRIETARY AND CONFIDENTIAL

ILLUSTRATIONS

Figure		Page
1-1	Multiline Communications Processor (MLCP) In Series 60 Level 6 System Configuration	1-4
1-2	MLCP Layout and Dimensions	1-7
2-1	MLCP/Megabus Interface	2-4
2-2	MLCP/Line Adapter Interface	2-5
2-3	MLCP Functional Components	2-9
2-4	Megabus Line Configuration for MLCP Instructions	2-10
2-5	MLCP Hardware Major Functional Components	2-11
2-6	Read/Write RAM Organization	2-13
2-7	MLCP Transmit Functions	2-16
2-8	MLCP Receive Functions	2-17
2-9	MLCP Operational States	2-21
3-1	MLCP Hardware Major Block Diagram	3-2
3-2	Microprocessor and Control Store Logic Functional Block Diagram	3-6
3-3	Microprocessor Intermediate Block Diagram	3-7
3-4	Microprocessor CPE Chip Functional Block Diagram	3-8
3-5	Fast Carry Logic	3-17
3-6	Control Store Word Format	3-19
3-7	Control Store Logic	3-19
3-8	Microprogram Control Unit Functional Block Diagram	3-20
3-9	Microprogram Control Unit (MCU) Chip Functional Block Diagram	3-21
3-10	Subroutine Address Buffer/Address Multiplexer Logic	3-25
3-11	Control Store Buffer Logic	3-26
3-12	Strobe Generation Logic	3-28
3-13	Master Clear and QLT Logic	3-29
3-14	Megabus Control and Hardware-Implementation of I/O Instructions	3-32
3-15	Hardware and Firmware Implementations	3-33
3-16	Master Cycle Logic	3-39
3-17	ACK/NAK/Wait/Respond Logic	3-40
3-18	Busy Logic	3-41
3-19	Function Code Control Logic	3-42
3-20	CCB Address Control Logic	3-44
3-21	PROM Bit Decoder Logic	3-51
3-22	Megabus Interface Register and Control Logic	3-56
3-23	Megabus Interface Register (MIR) and B-Mux Functionality	3-57
3-24	MIR (MLCP is Master Sending Data to Megabus)	3-58
3-25	MIR (MLCP is Slave Receiving Data from Megabus)	3-59
3-26	Content of MIR During an Output (IOLD Output Address) Instruction	3-60
3-27	Content of MIR During an Input (Input ID) Instruction	3-61
3-28	Parity Checking and Generation Logic	3-62

Figure		Page
3-29	R/W RAM Logic Functional Block Diagram	3-65
3-30	R/W RAM Address Control Logic	3-66
3-31	Request Memory/Start Memory Cycle Logic	3-67
3-32	System Check	3-69
3-33	MLCP Timing Diagram	3-71
3-34	Line Adapter Interface Logic	3-74
3-35	LACOD/LAHERE Multiplexer Selection	3-77
3-36	Line Adapter Request Priority Logic	3-80
3-37	Addressing the Line Adapter	3-81
3-38	CRC Logic	3-85
3-39	Character Size Logic	3-86
3-40	CRC-16 Accumulation	3-87
3-41	CCITT Accumulation	3-88
3-42	CRC-12 Accumulation	3-89
3-43	LRC-8 Accumulation	3-90
4-1	Register 0 Bit Significance (Copy of LCT 21/43)	4-3
4-2	Register 1 Bit Significance in Block Mode (Copy of LCT 5/37)	4-4
4-3	Register 1 Bit Significance During Instruction Execution	4-4
4-4	Channel Number Assignments Within Register 6 and Register 7	4-5
4-5	Channel Control Program Counter Arrangement Within Register 8 and Register 9	4-5
4-6	Random Access Memory Segmentation	4-6
4-7	LCT Channel Command Byte LCT 8/40 Bit Significance	4-10
4-8	LCT Channel Control Byte Bit Significance	4-10
4-9	LCT CLA Status Byte Bit Significance	4-11
4-10	LCT Status Byte 1 Bit Significance	4-11
4-11	LCT Status Byte 2 Bit Significance	4-12
4-12	LCT CLA Control Byte Bit Significance	4-12
4-13	Eight-Byte CCB Format	4-20
4-14	CCB Control Byte Bit Significance	4-21
4-15	MLCP Firmware Overview Flow Chart	4-35
4-16	Quality Logic Test/Service Scan/Channel Scanner Routines	4-37
4-17	Line Adapter Ready/Start I/O/Form Four-Bit Channel Number for I/O Routines	4-39
4-18	I/O Command Decode/Output Channel Control/Start Write/Read Routines	4-40
4-19	Input LCT Byte/Input Data Set Status/Output Interrupt Control/Output LCT Bytes/Set Start I/O in LCT 9/Input Extended Device ID Number Routines	4-41
4-20	Initialize Channel Routine	4-43
4-21	Stop I/O/CCB List Reset Routines	4-44
4-22	Instruction Fetch Routine	4-45
4-23	Generic Instruction Routines	4-46
4-24	Next Character Instruction Routines	4-49

HONEYWELL PROPRIETARY AND CONFIDENTIAL

Figure		Page
4-25	Input Line Register/Receive Routines	4-52
4-26	Output to Line Register/Send Routines	4-53
4-27	LCT and Displacement Addressing Instruction Routines	4-54
4-28	Immediate Operand Instruction Routines	4-55
4-29	Branch True Instruction Routines	4-56
4-30	Branch False Instruction Routines	4-58
4-31	Start Block Mode DMA Routine	4-60
4-32	Block Mode Write DMA Read/DMA Read Routines	4-61
4-33	Block Mode Read DMA Write Routines	4-63
4-34	Block Mode DMA Overhead Routine	4-64
4-35	DMA Overhead Routine	4-65
4-36	Transfer Status in LCT to CCB Routines	4-66
4-37	Range Decrement Routine	4-67
4-38	Fetch Data From Buffer Routine	4-68
4-39	Termination Routine	4-69
4-40	Deferred Interrupt Routine	4-70
4-41	Interrupt I/O Interrupt CCP Routines	4-71
4-42	Data Set Scan Routine	4-72

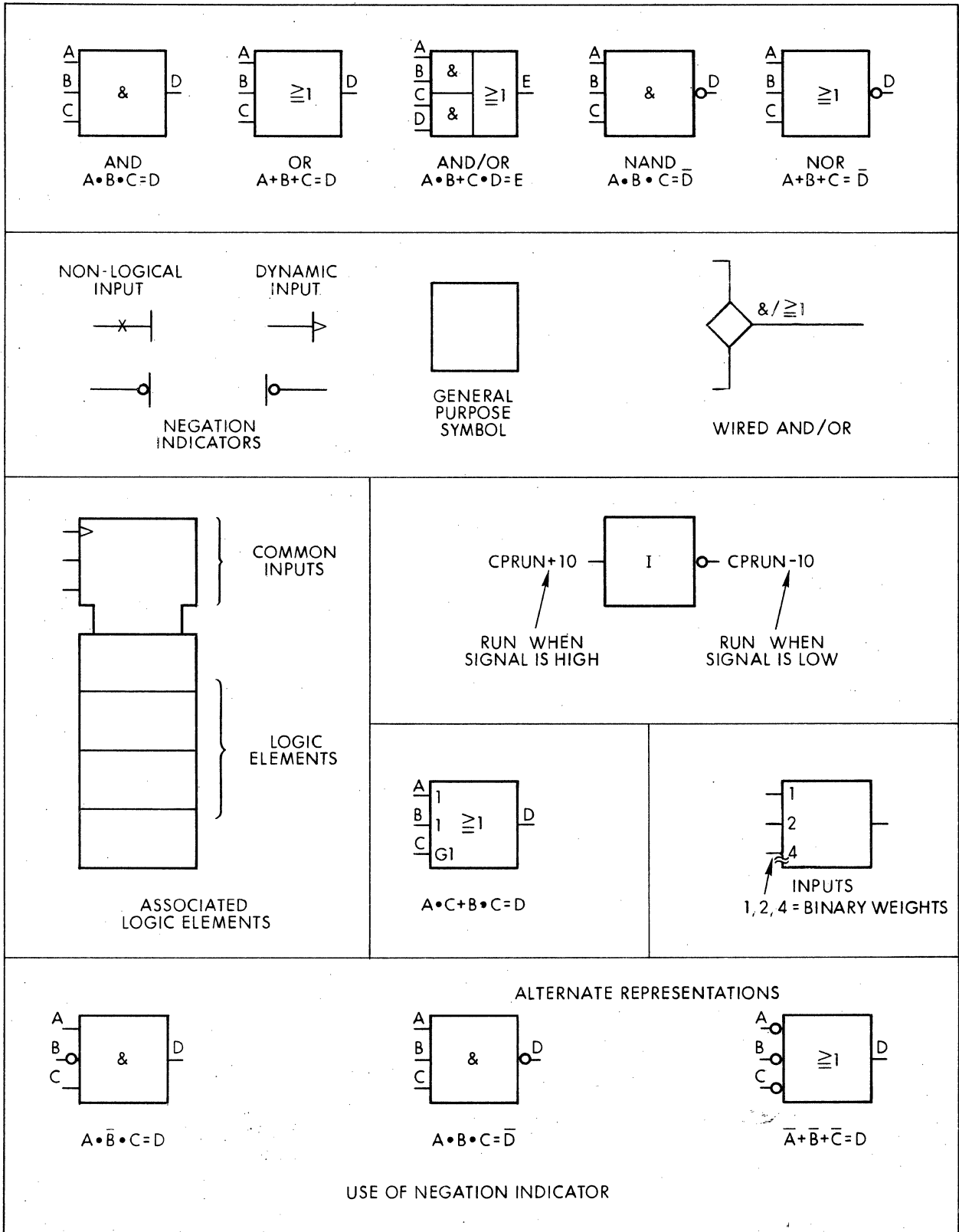
TABLES

Table		Page
1-1	Reference Documents	1-3
1-2	Adapters and Attachable Communications Devices	1-4
2-1	Megabus/MLCP Interface Signal Lines	2-2
2-2	MLCP/Line Adapter Interface Signal Lines	2-6
2-3	CRC Polynomials	2-14
3-1	Microprocessor Input/Output Functions	3-9
3-2	Microprocessor Function Selection	3-10
3-3	Scratch Pad Register Selection	3-10
3-4	Microprocessor Function Summary	3-11
3-5	M-, I-, and CRC Mux Input Selection	3-13
3-6	Microprocessor Function with K-Bus = 0000000 or 1111111	3-14
3-7	MLCP Control Multiplexers	3-16
3-8	Control Store Output Functions	3-18
3-9	MCU Input/Output Functions	3-22
3-10	Address Control Functions	3-23
3-11	Flag Control and Load Functions	3-24
3-12	Strobe Functions	3-29
3-13	Command Function Codes	3-34
3-14	Firmware-Implemented Commands	3-34
3-15	Function Code Translation for Firmware-Implemented Commands	3-35
3-16	Hardware-Implemented Commands	3-36

HONEYWELL PROPRIETARY AND CONFIDENTIAL

Table		Page
3-17	Function Code Translation for Hardware-Implemented Commands	3-36
3-18	CCB Locations in R/W RAM	3-45
3-19	Synchronous Clock Line Frequencies	3-73
3-20	Determination of Line Adapter Presence and ID Codes	3-76
3-21	Address and Control Line Functionality at the MLCP/Line Adapter Interface	3-82
4-1	Register Application	4-3
4-2	Line Control Table Topology	4-7
4-3	Line Control Table Byte Description	4-8
4-4	Instruction Formats and Decode	4-14
4-5	General Instruction Set	4-15
4-6	Double Operand Instruction Set	4-16
4-7	Line Register Input/Output Instructions	4-17
4-8	Branch Instruction	4-17
4-9	Data Input/Output Instructions	4-19
4-10	Cycle Flow Chart Symbols	4-23

LOGIC SYMBOLY





INTRODUCTION

1.1 SCOPE AND PURPOSE OF THIS DOCUMENT

This manual* describes the functionality and operation of the Series 60, Level 6 Type MLC9103 Multiline Communications Processor (MLCP). Programming considerations are presented to the extent necessary to make the hardware/firmware descriptions comprehensible. The reader is referred to the MLCP Programmer's Reference Manual (Order No. AT97), the Level 6 Minicomputer Handbook (Order No. AS22), or the Peripherals Handbook (Order No. AT04) for programming details. Operational theory contained in this document is designed to acquaint the reader with the major functional hardware elements of the MLCP and to aid in the analysis of MLCP operation at the detailed level presented in the logic block diagrams found in the MLCP Reference Manual.

1.1.1 Organization of this Document

This manual is composed of four sections:

1. Section I: INTRODUCTION. Describes the scope and purpose of this document; lists related reference documents; provides a summary description of the MLCP within the context of the Series 60, Level 6 System; itemizes MLCP specifications and performance characteristics, and defines abbreviations and special terminology used within the document.

*This manual reflects Firmware Release Level 11.0.

2. Section II: THEORY OF OPERATION-OVERVIEW. Defines the interfaces which the MLCP has with the Series 60, Level 6 Megabus* network and with attachable (pluggable) line adapters; describes the interrelationship between MLCP software, firmware and hardware; introduces the major hardware functional components; provides an operational overview in terms of functions performed and operational states and modes; provides a limited description of programming influence upon certain hardware elements.
3. Section III: THEORY OF OPERATION - INTERMEDIATE. Provides an intermediate level theory of operation of each area of the major block diagram discussed in Section III. Function names are included in many places to enable the reader to enter the logic block diagrams (see MLCP Reference Manual) if desired.
4. Section IV: THEORY OF OPERATION - CYCLE FLOW. Provides a description of the firmware cycle flow within the MLCP, the flows (operations) the MLCP can execute and how and when they are executed. The overview flow chart contains one block for each cycle grouping the CPU can execute and shows entries and exits from various cycle groupings. The intermediate flow charts contain one block for each major operation that takes place within a cycle grouping.

1.1.2 Reference Documents

See Table 1-1.

1.2 MLCP DESCRIPTION

The MLCP, in conjunction with a variable configuration of up to four communications line adapters (CLAs), is a programmable processor which is supported by a firmware operating environment for use with the Series 60 Level 6 Megabus. The MLCP has the logic required for attachment to the Megabus and for control (via the appropriate adapter) of the communications devices listed in Table 1-2.

The MLCP (Figure 1-1) supports the data transfer and control of data for up to eight full duplex, low-speed (300 bits per second or less) and medium-speed (600 to 9600 bits per second) lines on a single Series 60 Level 6 Communications Board. The MLCP accommodates high line speeds as well with reduced line connectibility. One line up to 72kb full duplex may be attached.

The MLCP provides the firmware and hardware common to all line adapters and channel-specific firmware. Channel-specific logic is contained on interchangeable, pluggable line adapters. Each line adapter consists of either one-line or two-line interfaces, depending upon the complexity and circuit requirements of the particular line adapter functions and data set interface to be supported. Information is transmitted between the line adapter and the MLCP in

*Trademark of Honeywell Information Systems Inc.

HONEYWELL PROPRIETARY AND CONFIDENTIAL

byte-serial form. The adapter serializes byte information received from the MLCP for transmission to the attached data set and assembles serial data from the data set into byte form for transmission to the MLCP.

Each communication line operating with the MLCP is a full duplex data path composed of two independently controlled channels. Each channel is capable of either sending or receiving a communications type data stream between the main memory visible at the Megabus and any of the 16 communications channels via Direct Memory Access (DMA) operation. Typically, a pair of adjacent (even, odd) channels is set up and configured as a communication line.

MLCP design is based on a microprocessor which has been tailored by a microprogram to serve as a communication processor. Each MLCP has 4096 bytes of self-contained read/write memory, most of which is directly visible to a user of the MLCP. This visibility allows each user to interpret and manipulate the communications data stream according to the requirements of the application, since the byte level operations in the MLCP are software configurable.

In the process of transferring data, the MLCP is capable of fully delimiting the data with special character generation and detection and block check information. The MLCP is also capable of editing and converting certain prespecified sequences in the data stream. Control of these functions is designated by configuration of a Line Control Table (LCT) and a Channel Control Program (CCP) which can be loaded by software into the MLCP.

Table 1-1 Reference Documents

TITLE	DOCUMENT NUMBER	ORDER NUMBER
Series 60 Level 6 Minicomputer Handbook	-	AS22
Series 60 Level 6 Peripherals Handbook	-	AT84
Series 60 Level 6 MLCP Programmer's Reference Manual	-	AT97
Model 33 Systems Description Manual	71010669-100	FW89
Model 6/34/36 Systems Handbook	71010200-200	FL35
Model 33 CPU Manual	71010670-100	FW90
Model 6/34/36 CPU Manual	71010201-200	FL36
Type DCM9101 Dual Asynchronous Communications Line Adapter Manual	71010232-200	FL50
Type DCM9102 Single Asynchronous Communications Line Adapter Manual	71010236-200	FL75
Type DCM9103 Dual Synchronous Communications Line Adapter Manual	71010231-200	FL49
Type ECM9104 Single Synchronous Communications Line Adapter Manual	71010237-200	FL76
Type DCM9105 Synchronous Current Mode Adapter Manual	71010434-100	FM69
Type DCM9106 HDLC Adapter Manual	70010961-100	FN41
Type DCM9108 Synchronous Balanced Line Adapter Manual	71010431-100	FN24
Type DCM9109 Synchronous MIL STD, 188C Manual	71010217-100	FM75
Type DCM9110 Dual Autocall Adapter Manual	71010262-100	FN22
Model 3x, 4x, 5x System Checkout and T&V Manual	71010207-100	AW94
Type MLC9103 MLCP Reference Manual (Assembly No. 60127901-100)	71010380-101	FM39
Type MLC9103 MLCP Reference Manual (Assembly No. 60130483-001)	71010440-100	FM65

Table 1-2 Adapters and Attachable Communications Devices

LINE ADAPTER	COMMUNICATIONS DEVICE
1. Asynchronous Adapter (EIA RS232 Interface)	Dataphone Data Set 103, 112, 113, 202 or equivalent, or modem bypass
2. Synchronous Adapter (EIA RS232 Interface)	Dataphone Data Set 201, 203, 208 or equivalent, or modem bypass
3. Current Mode Synchronous Adapter	Dataphone Data Set 301, 303, or equivalent, or modem bypass
4. Balanced Line Synchronous Adapter	Digital Data System or equivalent
5. HDLC Adapter (EIA RS232 Interface)	See item 2
6. MIL 188C Synchronous Adapter	See item 2
7. Dual Autocall Adapter	801 ACU

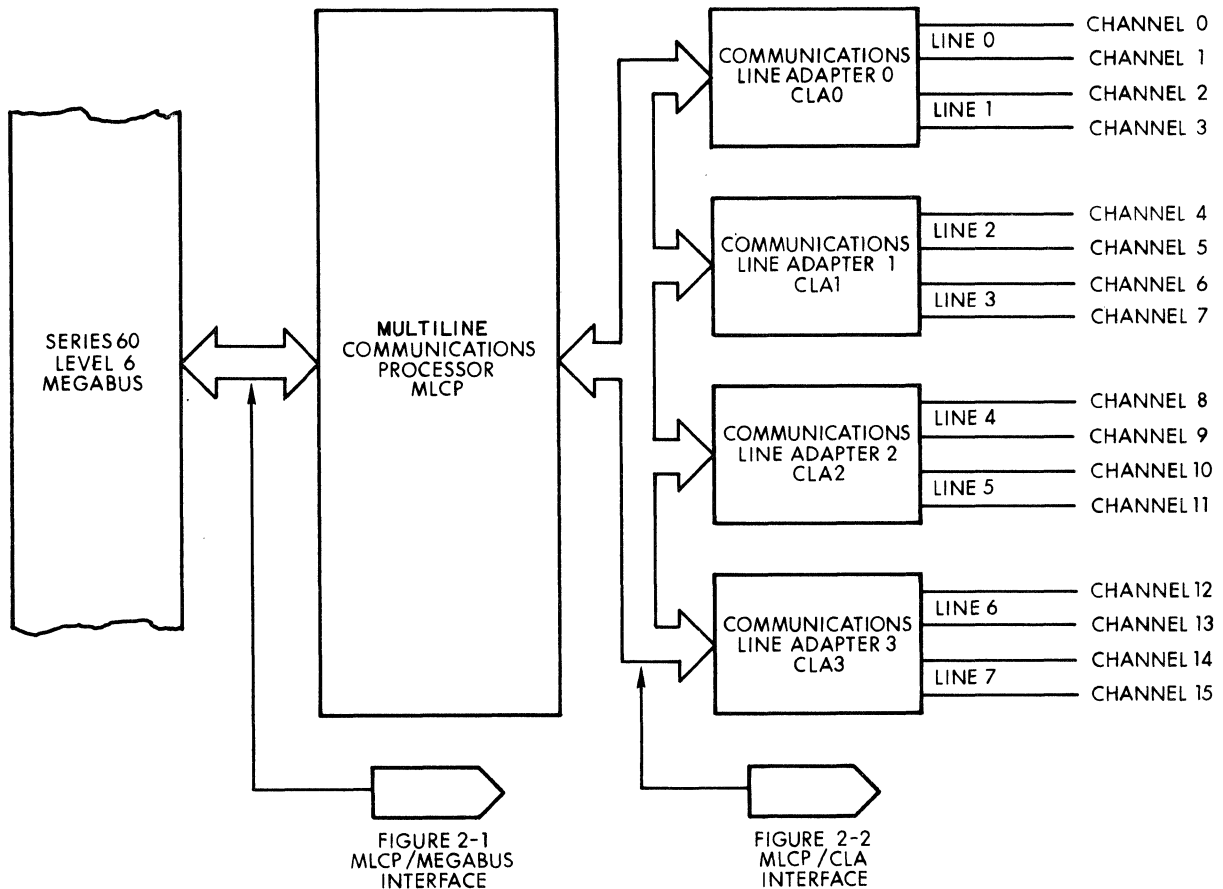


Figure 1-1 Multiline Communications Processor (MLCP) In Series 60 Level 6 System Configuration

1.3 SPECIFICATIONS

1.3.1 Line Capabilities

1. Controls up to eight full- or half-duplex communications lines.
2. Operates with both synchronous and asynchronous lines.
3. Up to two lines (four channels) may be controlled by each line adapter. Each channel connected to a line adapter uses a pair of channels of opposite types (i.e., one transmit and one receive channel).

1.3.2 Block Mode

Blocks of data are transferred between the CPU and the MLCP memory (R/W RAM) by firmware under Communications Control Block (CCB) control.

1.3.3 Line Configuration

Lines which interface with the MLCP may be synchronous or asynchronous. Configurable items for both types are described in the following paragraphs.

1.3.3.1 Configuration of Synchronous Lines

The following items are configurable for each synchronous line:

1. Character length of five, six, seven, or eight bits
2. Even, odd, or no parity
3. A synchronization character may be loaded into the CLA register by the CCP. When receive is turned on, the incoming data must match this character before transfer from the CLA is initiated by the MLCP. In transmit, a transmit-fill character is substituted for the absent data from the MLCP in case of an underrun condition.
4. One of four Cyclic Redundancy Check (CRC) codes (CRC-12, CRC-16, CRC-CCITT or LRC-8) may be specified.

1.3.3.2 Configuration of Asynchronous Lines

The following items are configurable for each asynchronous line:

1. Character length of five, six, seven, or eight bits
2. Even, odd, or no parity
3. One or two stop bits for six-, seven-, or eight-bit characters. One or $1\frac{1}{2}$ stop bits for five-bit characters.

The Communications-Pac adds start and stop bits on transmit and strips start and stop bits on receive.

1.3.4 Programmable Line Controls

The MLCP can be programmed to control the following functions on a per-channel basis via CCP instructions:

1. Enable transmit with parity
2. Enable transmit without parity
3. Disable transmit
4. Enable receive with parity
5. Enable receive without parity
6. Disable receiver - report open line (asynchronous lines)
7. Enable search for sync (synchronous lines)
8. Strip sync in receive (synchronous lines)
9. Do not strip sync (synchronous lines)

1.3.5 Physical Characteristics

The MLCP is a single Series 60 Level 6 module which accommodates up to four CLAs. Cables to external lines connect via the back edge of the CLAs (refer to the appropriate CLA manual) rather than directly to the MLCP.

Figure 1-2 shows the MLCP (BF4MLC) layout and dimensions and indicates the positioning of the Communications-Pacs. At the base of the module are two 50-pin connectors (Z01 and Z02) which are used to connect the MLCP to the Series 60 Level 6 Megabus. Connectors W01 through W08 are 25-pin connectors located on the component side of the MLCP for the physical and electrical attachment of Communications-Pacs. The two hex rotary switches located in positions B14 and B15 on the MLCP are set to the channel number of the MLCP as specified in the system configuration. The hex rotary switch in position E04 or the jumpers located in positions N15Y, N16S, N16U, and N16W are configured to generate a specific baud rate for the synchronous clock signal. The setting of the hex rotary switch or the installation or removal of jumpers is determined by the board assembly number of the MLPC (see subsection 3.5.4 for further details).

1.3.6 Environmental Requirements

1. Temperature: 0 to 50°C ambient
2. Relative Humidity: 5 to 95%.

1.3.7 Power Requirements

There are no special power requirements for MLCP logic. Certain CLAs require ±12 volts. This voltage and +5 volts are supplied by the MLCP board through the stacking connector.

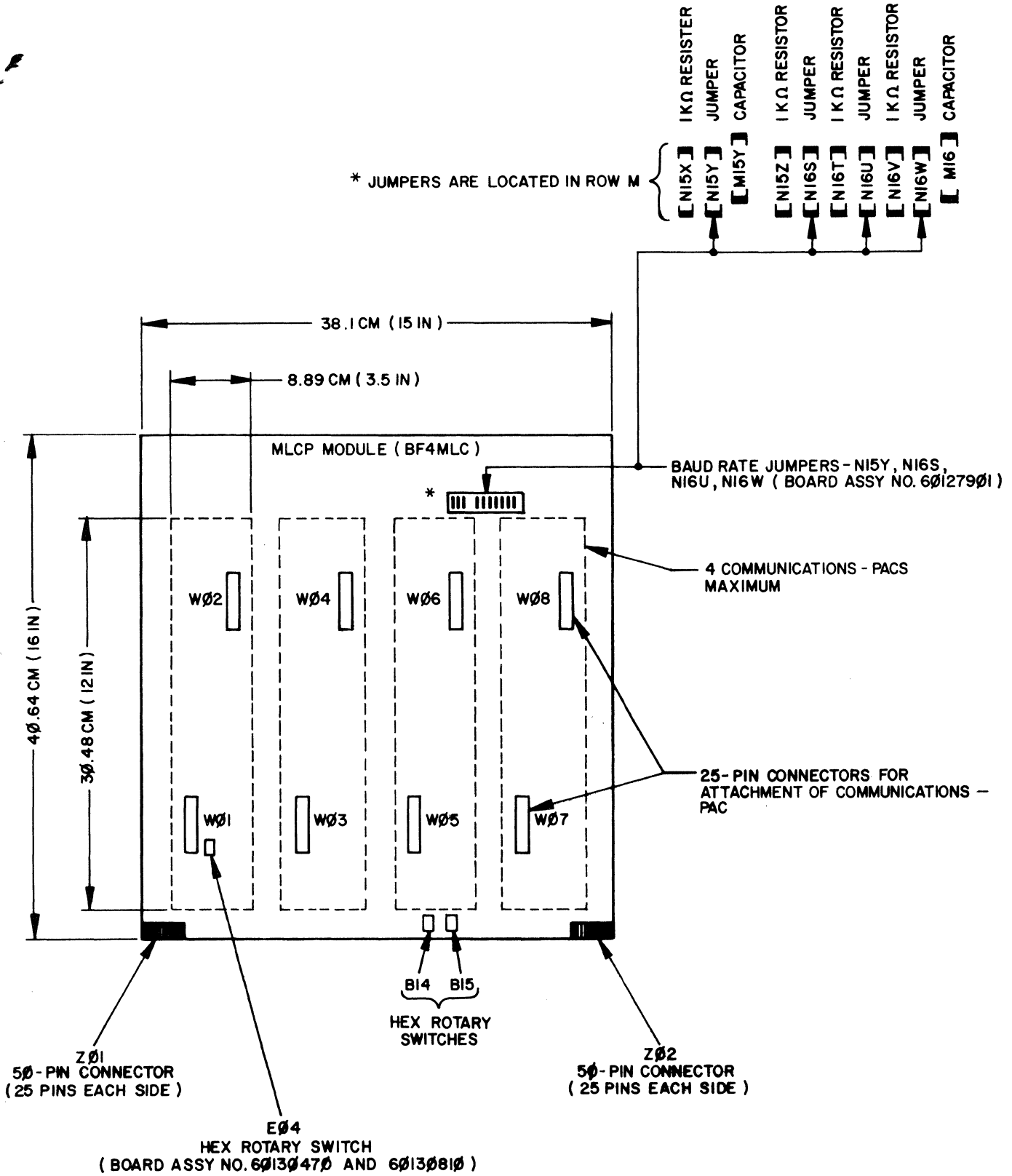


Figure 1-2 MLCP Layout and Dimensions

1.4 ABBREVIATIONS/DEFINITIONS

ACK	Acknowledge
ACLA	Asynchronous Communications Line Adapter
ACU	Automatic Calling Unit
BSC	Binary Synchronous Communication
CCB	Communications Control Block
CCH	Calculate Block Check
CCP	Channel Control Program
CDB	Communications Data Block
CH	Channel
CHAR	Character
CLA	Communications Line Adapter
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
CRI	Channel Request Interrupt
DCE	Data Communications Equipment
DMA	Direct Memory Access
DTE	Data Terminal Equipment
EA	Effective Address
FC	Function Code
FDX	Full Duplex
FIFO	First In First Out
HDX	Half Duplex
ID	Identification (Device)
I/O	Input/Output
LCT	Line Control Table
LR	Line Register
LRC	Longitudinal Redundancy Check
LSI	Large Scale Integration
MLCP	Multiline Communications Processor
NAK	Negative Acknowledgement
ORU	Optimum Replaceable Unit
QLT	Quality Logic Test
R	Register of the MLCP
RAM	Random Access Memory
RFU	Reserved for Future Use
ROM	Read Only Memory
PWCB	Read Write Control Block
RWCT	Read Write Control Table
RWDB	Read Write Data Block
SCLA	Synchronous Communications Line Adapter
TBS	To Be Supplied

II THEORY OF OPERATION - OVERVIEW

2.1 INTERFACE DESCRIPTION

2.1.1 MLCP/Megabus Interface

The MLCP attaches to the Series 60 Level 6 Megabus via the interface shown in Figure 2-1. (Terms and mnemonics are listed and defined in Table 2-1.) This interface is the control and transfer link between the MLCP and any other controller (i.e., CPU, main memory, etc.,) within the system and provides a path for address, data, and control information. This interface also supplies the paths for determining priority of a request from the MLCP.

2.1.2 MLCP/CLA Interface

The MLCP/CLA interface is shown in Figure 2-2 and described in Table 2-2. Interface signal lines are depicted as seen by the MLCP. This interface provides a common link between the MLCP and up to four adapters and gives firmware the capability of selecting the proper adapter and performing the designated operation. It provides the paths necessary to supply the adapters with data, control, and timing signals, and allows the adapters to communicate to the MLCP requests for service and data from the communications devices.

HONEYWELL PROPRIETARY AND CONFIDENTIAL

Table 2-1 Megabus/MLCP Interface
Signal Lines (Sheet 1 of 2)

TERM/MNEMONIC	DESCRIPTION
Data Bits 0 through 7 (BSDT00 to 07)	These eight data bit lines represent the most significant byte of data.
Data Bits 8 through 15 (BSDT08 to 15)	These eight data bit lines represent the least significant byte of data.
Data Parity-Left Byte (BSDP00)	This signal contains odd parity for data bits 0 through 7.
Data Parity-Right Byte (BSDP08)	This signal contains odd parity for data bits 8 through 15.
Address Bus Bits 0 through 23 (BSAD00 to 23)	These 24 address bit lines contain an address to be used by memory or by a controller or central processor.
Address Parity (BSAP00)	This signal contains odd parity for the most significant byte of the Address bus, bits 0 through 7.
Priority Lines (BSAUOK through BSIUOK)	These lines are used to establish priority of the units attached to the bus.
MY OK (BSMYOK)	This signal indicates the unit that is presently using the bus.
Power On (BSPWON)	This signal is true when all power supplies in the system are operating correctly.
Logic Test In (BSQLTI)	This signal initiates the Internal Logic Test in a unit attached to the bus.
Logic Test Out (BSQLTO)	This signal indicates that the unit has successfully completed running its Internal Logic Test and is used as the Logic Test In signal for the next unit attached to the bus.
Acknowledge (BSACKR)	This signal indicates that the information on the bus has been accepted.
Data Cycle Now (BSDCNN)	This signal indicates that the information on the bus is valid.
Yellow (BSYELO)	This signal indicates that the accompanying transferred information is correct, but that a correction operation was performed.
Red (BSREDD)	This signal indicates that the accompanying transferred information is in error.

HONEYWELL PROPRIETARY AND CONFIDENTIAL

Table 2-1 Megabus/MLCP Interface
Signal Lines (Sheet 2 of 2)

TERM/MNEMONIC	DESCRIPTION
Byte (BSBYTE)	This signal indicates that the current transfer is a byte transfer rather than a word transfer.
Memory Reference (BSMREF)	This signal indicates that the address lines contain a memory address.
Wait (BSWAIT)	This signal indicates that the transfer will be accepted when the bus data register is available.
Negative Acknowledge (BSNAKR)	This signal indicates that the information on the bus has been refused.
Bus Request (BSREQT)	This signal indicates that one or more units on the bus have requested a bus cycle.
Second Half Bus Cycle (BSSHBC)	This signal identifies the second bus cycle in response to a memory read request.
Bus Write (BSWRIT)	This signal indicates that information on the bus is ready to be transferred.
Master Clear (BSMCLR)	This signal initializes the units attached to the bus.
Resume Interrupt (BSRINT)	This signal is a 200-nanosecond pulse which is issued by the central processor when it is capable of receiving interrupts again.

HONEYWELL PROPRIETARY AND CONFIDENTIAL

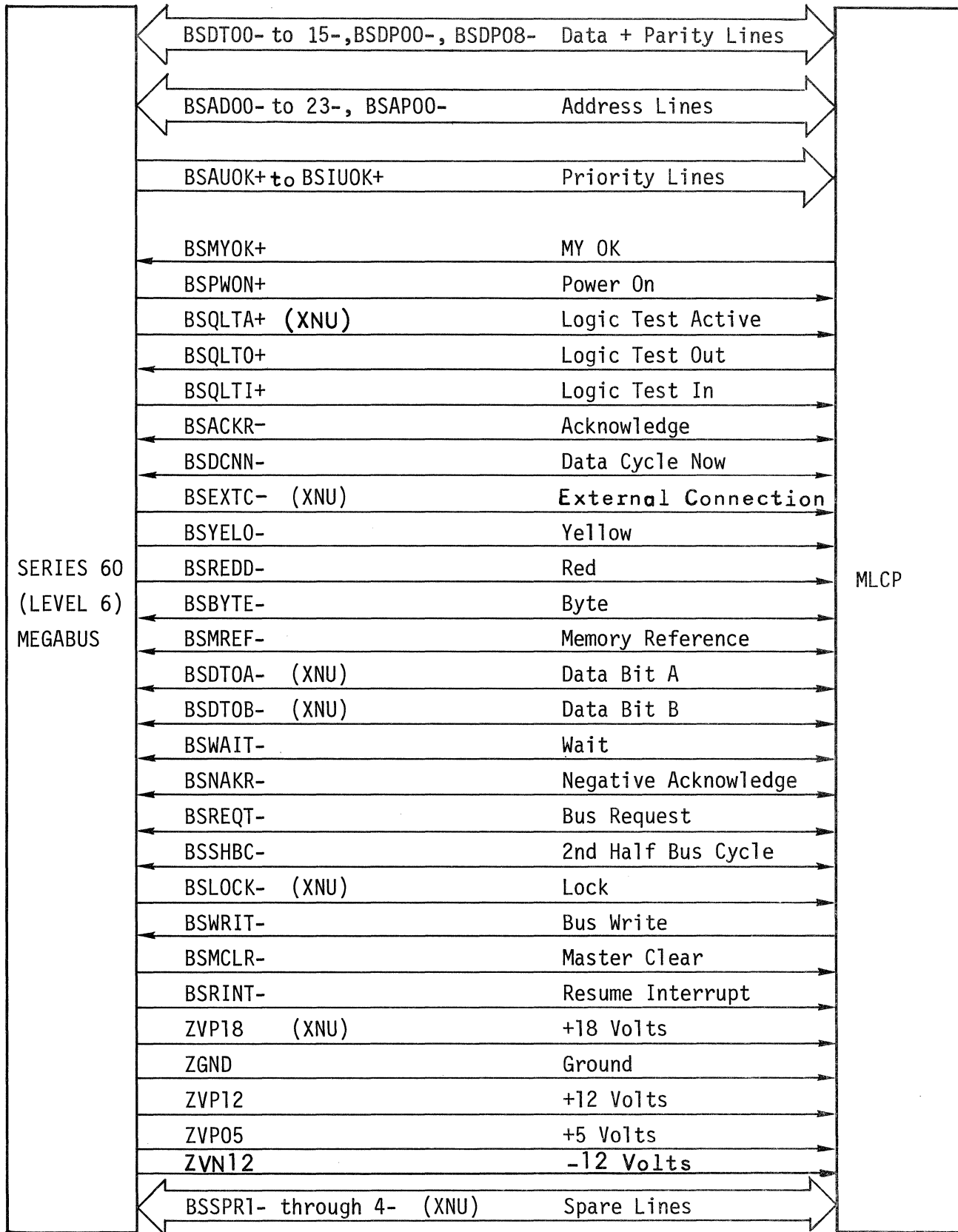
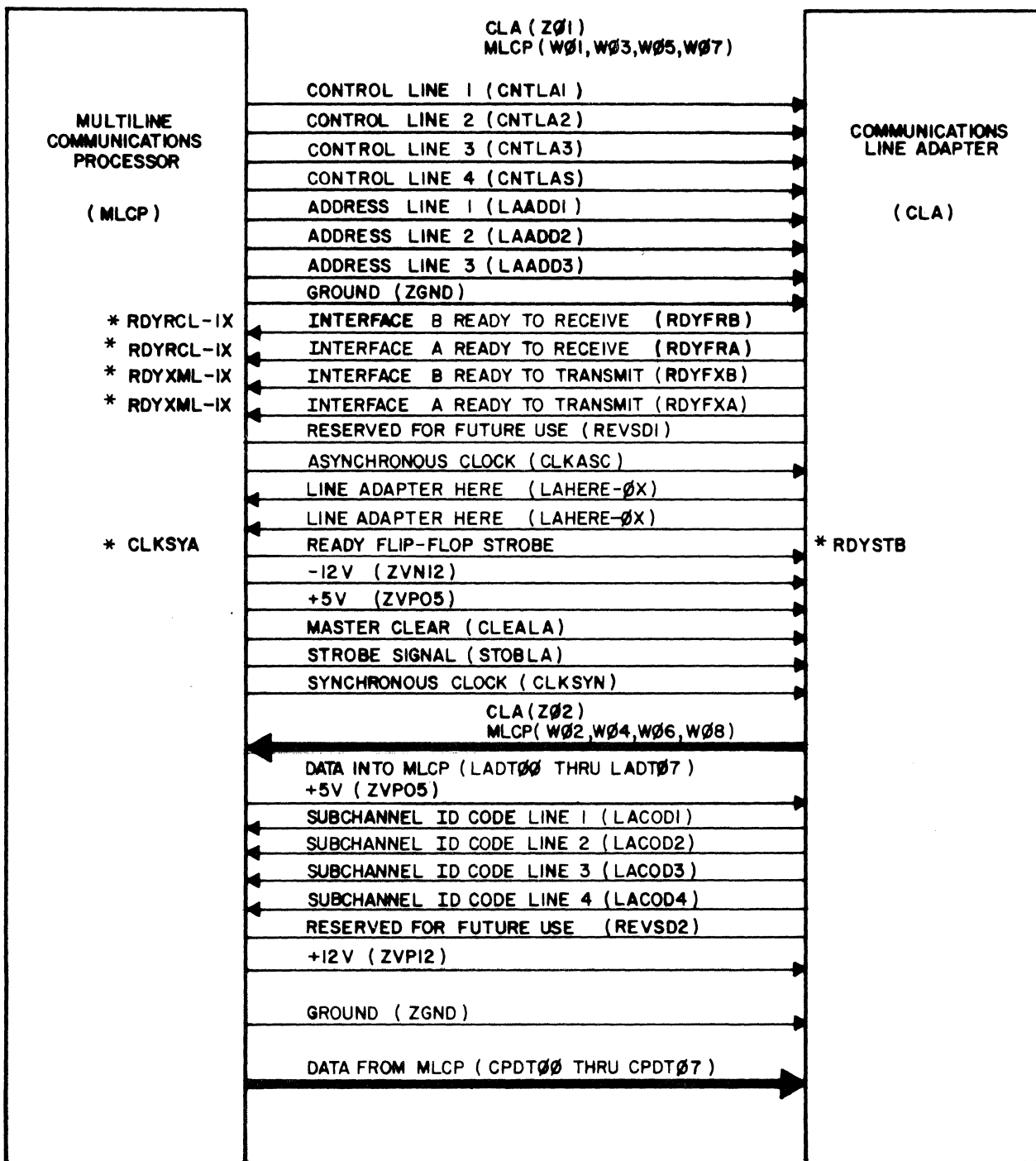


Figure 2-1 MLCP/Megabus Interface



* NOTE THAT MNEMONICS OF THESE LINES CHANGE AT THE CONNECTOR.

Figure 2-2 MLCP/Line Adapter Interface

HONEYWELL PROPRIETARY AND CONFIDENTIAL

Table 2-2 MLCP/Line Adapter Interface
Signal Lines (Sheet 1 of 2)

MNEMONIC	NAME	DESCRIPTION
CLEALA	Master Clear (General Reset)	Clears hardware in the line adapter. In the MLCP, terminates all line adapter status and error conditions and resets all data set control latches. The Ready Interrupt from channels configured to receive are disabled following general reset, while the data channel to the modem is held in the marking condition for channels configured to transmit.
CLKASC	Asynchronous Clock	Basic 921.6 kHz clock signal from the MLCP. Used to drive the baud rate generator in asynchronous line adapters.
CLKSYA	Ready Flip-Flop Strobe	The system clock signal on this line is used to synchronize the ready flip-flops in the line adapters with the MLCP.
CLKSYN	Synchronous Clock	The signal on this line is substituted for the modem or local clock for synchronous lines when the test mode bit is set in a data set control transfer. This clock is used to shift data in the data loopback path established by the test mode. This signal is also used for direct connect on synchronous lines.
CNTLA1 thru CNTLA4	Control Lines	Four lines which define the function to be performed by the line adapter. The Master Strobe line must be true for information on these lines to be considered as valid.
CPDT00 thru CPDT07	Data to CLA	Eight lines that carry data and address control information to the line adapter, as defined by the state of the four control lines and/or strobe STOBLA.
LAADD1 thru LAADD3	Address Lines	Three lines, two of which are decoded in the line adapter as the address of one of the four channels connected to the line adapter. The remaining line is the individual line adapter select line.

HONEYWELL PROPRIETARY AND CONFIDENTIAL

Table 2-2 MLCP/Line Adapter Interface
Signal Lines (Sheet 2 of 2)

MNEMONIC	NAME	DESCRIPTION
LACOD1 thru LACOD4	Subchannel ID Code	Four lines by which the MLCP identifies the specific type of the attached line adapter.
LADT00 thru LADT07	Data to MLCP	Eight lines that carry data or status information from the line adapter to the MLCP.
LAHERE-01 LAHERE-02	Line Adapter Here	Indicate to the MLCP that the line adapter is plugged into the MLCP board. If the line adapter can interface with only one modem, only one LAHERE line will be true. For a line adapter capable of interfacing with two modems, both LAHERE lines will be true, regardless of whether one or two modems are actually connected.
RDYRCL-1x RDYRCL-1x RDYXML-1x RDYXML-1x	Ready Lines	Four lines, one for each channel connected to the line adapter, which, when true, signify to the MLCP that a channel is ready to send or to receive information. These ready lines serve as the interrupt function, and there are a total of up to sixteen such lines entering the MLCP (four from each line adapter).
STOBLA	Strobe Signal (Master Strobe)	When the signal on this line is true, it causes information on the data lines to be loaded into the line adapter storage elements defined by the status of the four control lines.

2.2 MLCP FUNCTIONAL REQUIREMENTS

The basic functional components necessary for MLCP operation are software, firmware, and hardware as shown in Figure 2-3 and described in the following paragraphs.

2.2.1 Software

MLCP operations are a direct result of an input or output instruction from the Central Processor Unit (CPU). The general format of the Megabus as a result of I/O commands or MLCP responses is shown in Figure 2-4. For output instructions, the Megabus, as seen by the MLCP, is configured such that the address lines reflect the main memory module select address bits, the MLCP channel number, Communications Line Adapter (CLA) channel number, and a function code indicating the function to be performed by the MLCP and CLA. The data lines may reflect actual data to be sent to a line adapter, or control information for the MLCP, or address information for the MLCP.

The input instruction has the Megabus configured such that the address lines may reflect the channel number of the controller being addressed and the function code. The data lines have the number of the channel initiating the command.

In response to the input instruction, the MLCP loads the Megabus address lines with the channel number of the unit to receive the information on the data lines. An exception is main memory, where the address lines are set to the address into which the information on the data lines is to be written.

The following I/O commands are provided for control of the MLCP:

Output Commands

- | | | |
|----|----------------|---------------------------|
| 1. | IO (FC - 01) | Output MLCP Control |
| 2. | IO (FC - 05) | Output Channel Control * |
| 3. | IO (FC - 03) | Output Interrupt Control* |
| 4. | IO (FC - 0F) | Output CCB Control |
| 5. | IO (FC - 0B) | Output LCT Bytes* |
| 6. | IOLD (FC - 09) | Output Address and Range |

Input Commands

- | | | |
|----|--------------|------------------------------------|
| 1. | IO (FC - 1C) | Input Data Set Status* |
| 2. | IO (FC - 18) | Input Status |
| 3. | IO (FC - 1A) | Input Next Status |
| 4. | IO (FC - 0C) | Input Range |
| 5. | IO (FC - 26) | Input Device Identification Number |
| 6. | IO (FC - 1E) | Input LCT Byte* |
| 7. | IO (FC - 08) | Input Extended Device ID Number* |

2.2.2 Firmware

Firmware, a sequence of microinstructions resident in the microprogram control store, is the primary control element of the MLCP. The main function of firmware is to interpret external and internal events or conditions and react in a prescribed manner (i.e., setting or resetting of hardware functions). Efficient data transfer is also a result of firmware control of hardware components in the data path.

Commands marked with an asterisk () are implemented in firmware; all others are implemented in hardware.

The MLCP must be hard initialized before it can be set up and configured for operation. MLCP hard initialize is controlled by the Output MLCP Control command, the only operational mode command that must be generated by CPU software on an MLCP basis. All subsequent commands are channel specific. After the hard initialize, a series of firmware operations called the Quality Logic Test is performed to verify the integrity of MLCP hardware.

Upon completion of the Quality Logic test, the firmware conditions the MLCP for reception of certain I/O instructions which configure the channels for operation. Channel configuration is accomplished by loading the necessary Channel Control Programs (CCPs) and Line Control Tables (LCTs) into the 4K x 8-bit read/write RAM. Once the channel is configured, the MLCP is ready to be set up for data transfer on selected channels. This is accomplished by loading one or more Communication Control Blocks (CCBs) into the RAM. CCBs are basically data block descriptions containing the address, range, control, and status for each block transfer between the MLCP and either the Megabus or the CLA. CCBs are loaded via I/O instructions and are implemented by MLCP hardware rather than firmware. Only those I/O instructions not related to CCBs are implemented in firmware.

When the channel has been set up for data transfer, the firmware enters a routine which waits for a request from the Megabus or from one of the CLAs. When a request occurs, firmware will analyze status and priority and then proceed to the appropriate firmware operation for processing the request.

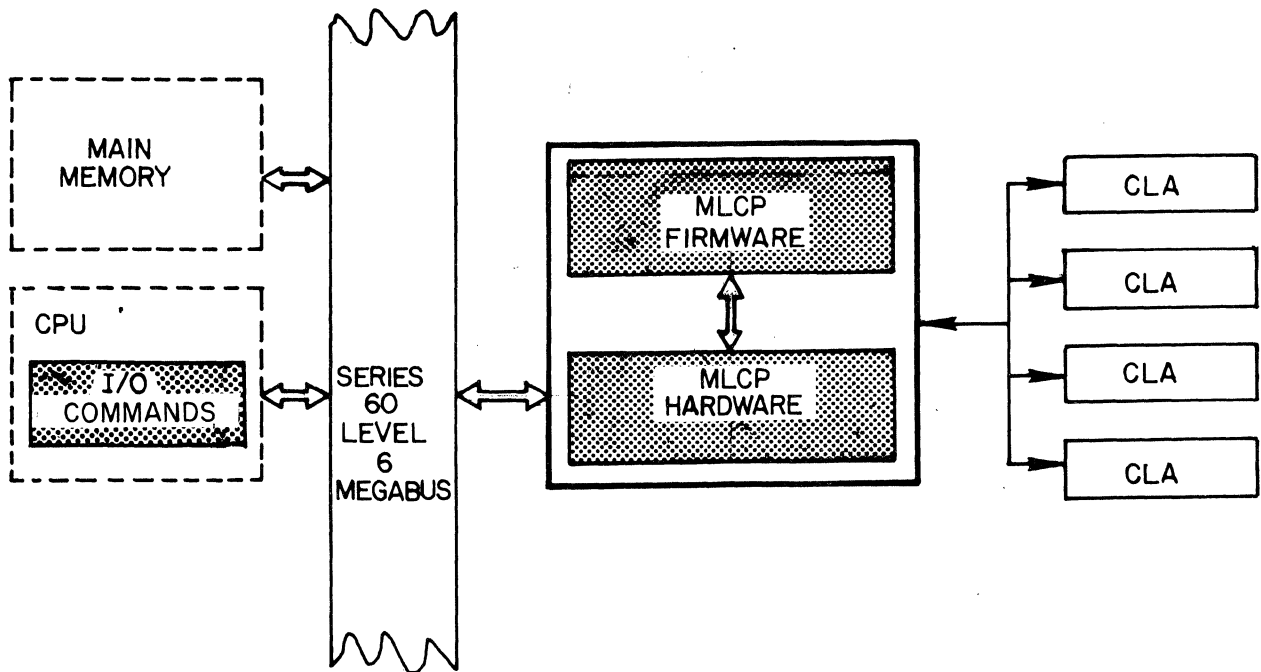
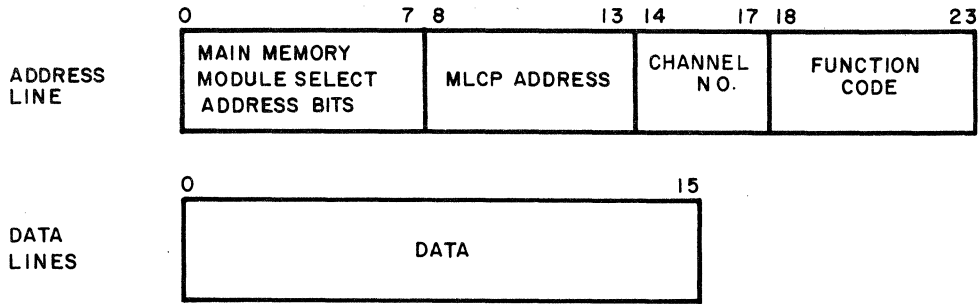


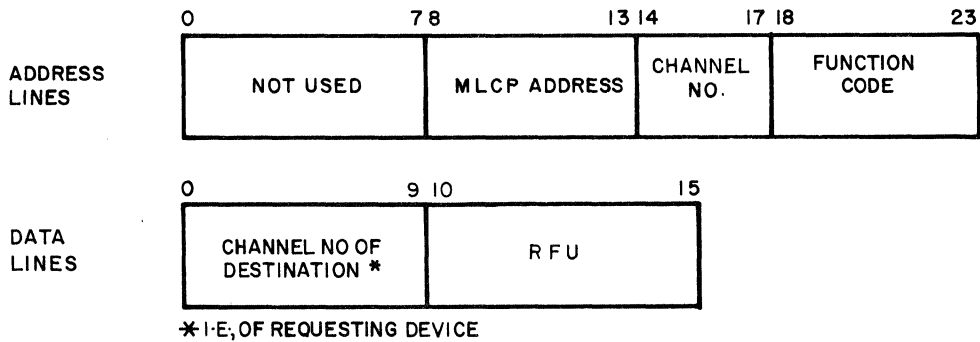
Figure 2-3 MLCP Functional Components

HONEYWELL PROPRIETARY AND CONFIDENTIAL

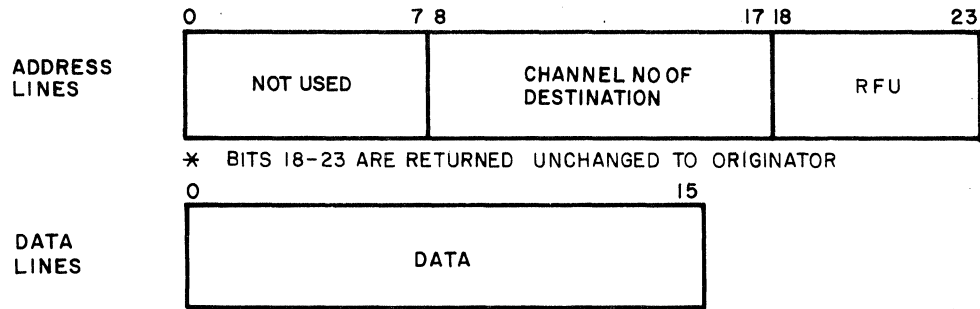
A. MLCP OUTPUT COMMANDS



B. MLCP INPUT COMMANDS



C. MLCP RESPONSE TO INPUT COMMANDS (CPU)



D. MLCP RESPONSE TO DIRECT MEMORY ACCESS COMMANDS (MAIN MEMORY)

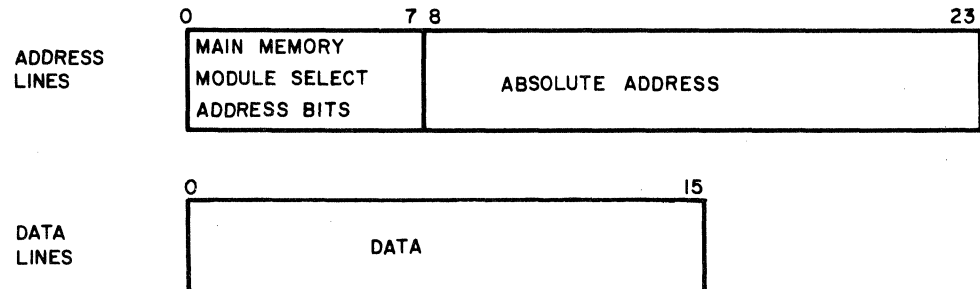


Figure 2-4 Megabus Line Configuration for MLCP Instructions

2.2.3 Hardware

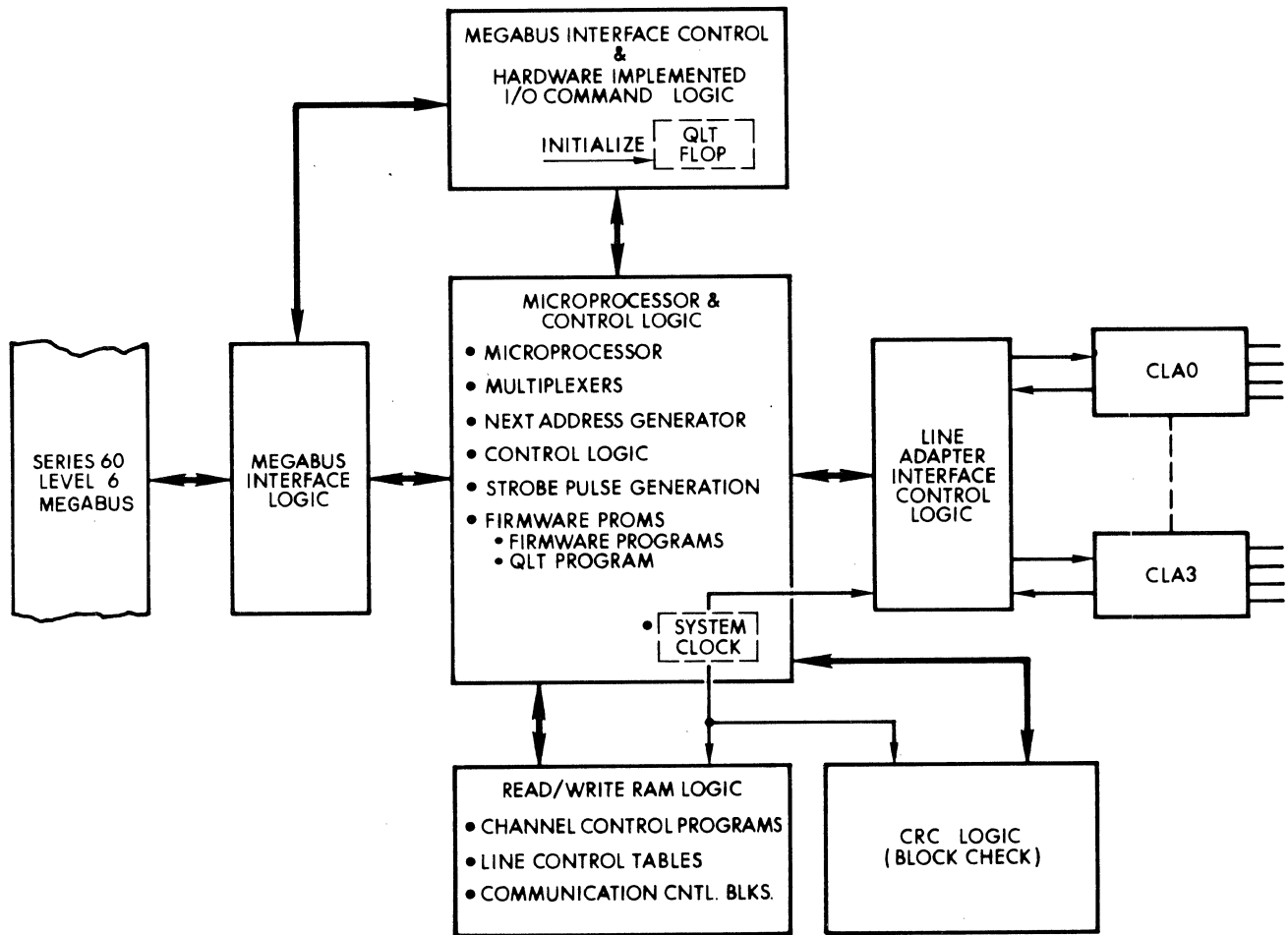
The MLCP hardware is organized into six fundamental logic areas as shown in Figure 2-5 and described in the following sub-paragraphs.

2.2.3.1 Line Adapter Interface Control Logic

This logic provides the MLCP with an interface for up to four line adapters. The major functions of this logic include resolution of conflicting CLA requests, data multiplexing, parity generation and checking, and special clock signal generation.

2.2.3.2 Megabus Interface Control Logic

This logic controls the transfer of data and instructions between the Megabus and the MLCP.



FOR NEXT LEVEL OF DETAIL,
SEE FIGURE 3-1

Figure 2-5 MLCP Hardware Major Functional Components

2.2.3.3 Read/Write RAM

The MLCP contains a 4096 x 8-bit random access memory, the content of which is used to direct the action of each MLCP channel. As shown in Figure 2-6, the read/write RAM is divided into three basic parts:

1. Communication Control Block (CCB) Area: Communication control blocks reside in a protected area of the RAM. These locations can be accessed only by I/O instructions or by a Block Read or Write.
2. Line Control Table (LCT) Area: Each of the eight lines in the MLCP has its own line control table, which contains status, configuration, setup and control information for the associated line.
3. Channel Control Program (CCP) Area: An area of R/W RAM is reserved for CCPs, which are lists of instructions used to process a channel's data communication character stream.

2.2.3.4 Cycle Redundancy Check (CRC) Logic

The block check hardware logic, available to each MLCP channel, enables either the generation (transmit) or residue accumulation (receive) of block check or CRC information. Residue accumulation is maintained in the channel line control table (LCT 3 and LCT 4 for a receive channel, and LCT 35 and LCT 36 for a transmit channel).

CRC residue during receive is checked by using the channel control program instructions to evaluate the content of the LCT CRC residue at the appropriate times. CRC residue may similarly be extracted from the LCT CRC residue at the appropriate time for data transmission. Reset and initialization of the CRC residue can be controlled as typical LCT entries either during the LCT loading or during CCP operation.

Data is added to the CRC residue only under control of the CCP instructions. Only the Receive, Send and CCH instructions can cause data to be added to the current CRC residue. The particular CRC polynomial used to accumulate the CRC residue is one of the four listed in Table 2-3. The polynomial is a definition of the configuration of the hardware which generates the check character.

The CRC selection field (cycle redundancy check polynomial) is stored in bits five and six of byte 2/34 of the line control table (LCT 2 for receive, LCT 34 for transmit).

When the CRC polynomial is set for a Longitudinal Redundancy Check (LRC), which is one byte, the LRC field is located in LCT 3 for a receive and LCT 35 for a transmit. LCT 4 and LCT 36 are set to Zero.

2.2.3.5 Microprocessor and Control Store Logic

The major functional elements of this area are the microprocessor, which performs arithmetic and logic operations; the microprogram control unit, which controls the sequence of microinstructions fetched from the control store; and the microprogram control store, which consists of Programmable Read-Only Memories (PROMs) containing the firmware for the MLCP.

2.2.3.6 Hardware-Implemented I/O Command Logic

This logic determines if the instruction is to be executed by MLCP firmware or hardware, and handles the loading of CCBs (which employ hardware-implemented instructions) into the R/W RAM.

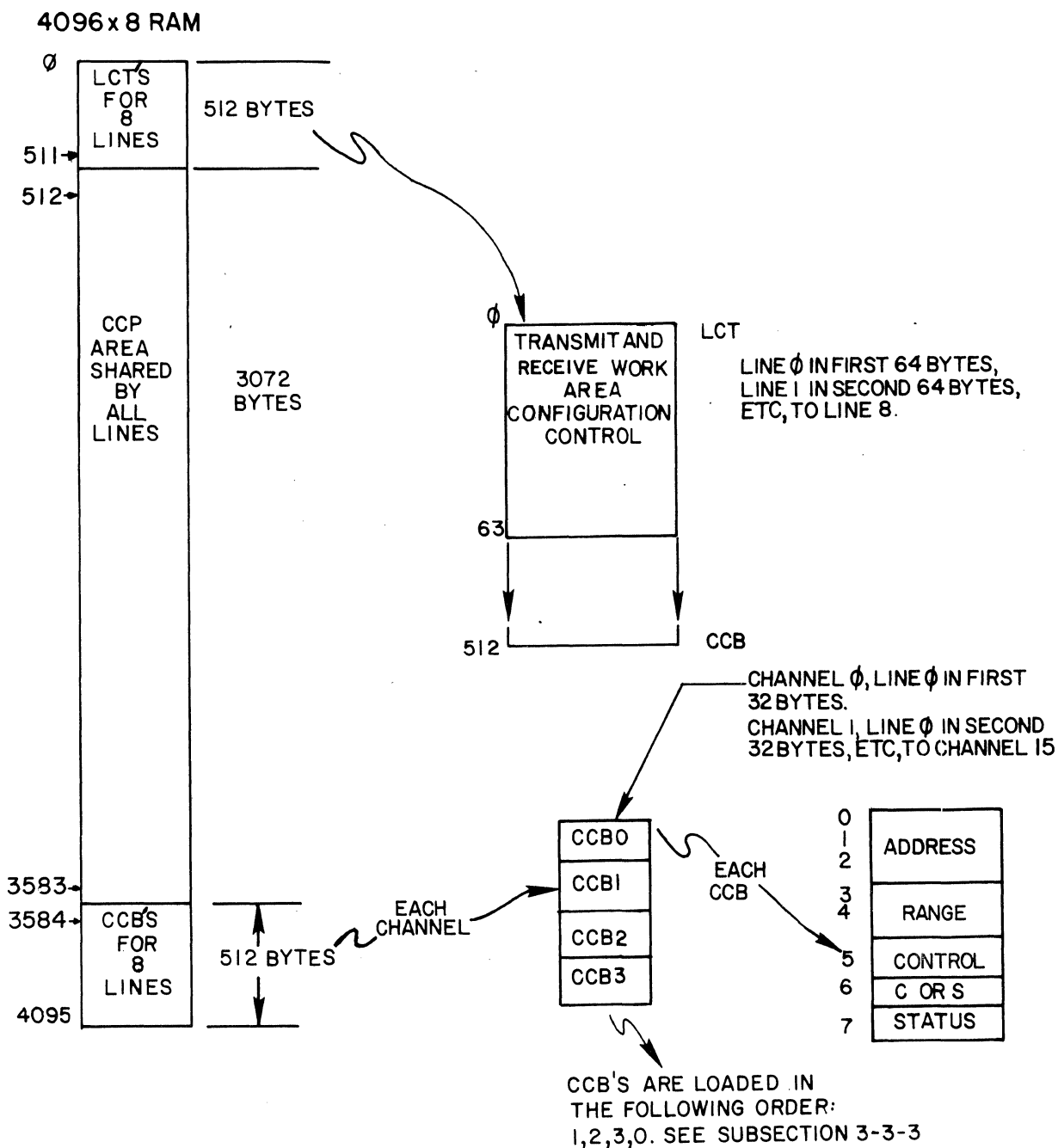


Figure 2-6 Read/Write RAM Organization

HONEYWELL PROPRIETARY AND CONFIDENTIAL

Table 2-3 CRC Polynomials

CRC SELECTION	CRC POLYNOMIAL	DESCRIPTION
01	$x^{16} + x^{12} + x^5 + 1$	CCITT Recommendation, HDLC, and Transparent Mode ASCII
11	$x^8 + 1$	LRC, Basic Mode ASCII
00	$x^{16} + x^{15} + x^2 + 1$	IBM BSC CRC-16
10	$x^{12} + x^{11} + x^3 + x^2 + x + 1$	CRC-12 IBM Transcode

2.3 SUMMARY OPERATIONAL DESCRIPTION

2.3.1 Operational Overview

The primary functions of the MLCP are to provide message delimiting, check character detection and generation, and performs minor edit functions. While providing this set of functions, the MLCP is also responsible for the transfer of data between main memory visible at the Megabus and communication lines or channels. This data may be viewed as a sequential data stream where the MLCP is providing the necessary control and transformation of that data into and out of the data formats of the connected lines and terminals.

Each character of byte that passes through the MLCP is handled individually by the MLCP microprocessor (see Figure 2-5). This microprocessor uses a combination of microprogram control, software-generated configuration parameters, and software-generated MLCP channel control programs to produce the appropriate user-defined action for each data element. These actions may consist of the transfer, translation, edit, deletion or addition of data elements, and the recognition of certain data elements or sequence of data elements as special conditions, message delimiters, DMA block boundaries, and block check characters.

At the Megabus side, data is arranged in DMA Communications Data Blocks (CDBs). The MLCP supports this interface via Communications Control Blocks (CCBs) which are set up and controlled by the MLCP I/O instructions. CCBs can be arranged in a list by channel (four CCBs maximum) to provide a larger timing window between the CPU servicing of each channel.

At the line side, the CLAs provide both the line interface and parallel/serial conversion of the byte data stream from the Megabus into bit stream serial form for the communication lines and vice-versa. The MLCP provides control of the line interface, and supplies bytes on transmit or byte buffers on receive to the CLAs as required.

As data passes through the MLCP from Megabus to CLA lines or vice-versa, the MLCP microprocessor exercises its control over the contents and formats of the data stream, generating appropriate status, interrupts, and control information.

HONEYWELL PROPRIETARY AND CONFIDENTIAL

Configuration data generated by CPU software is stored in the line control table in the RAM. The instructions to the MLCP processor also generated by CPU software are stored in the channel control program area in the RAM.

These CCPs contain byte-by-byte instructions designed to accept a communications data stream and to act on the elements of that data stream as previously noted. Initial startup, temporary status and results, CLA and MLCP channel configuration and control information, check character algorithms and results, and other data is maintained in the LCT for reference and is updated during CCP execution.

Figures 2-7 and 2-8 show pictorially how the MLCP provides the message-delimiting functions, such as looking for synchronization characters, Start of Header (SOH), Start of Text (STX), End of Text (ETX), block checks on receive, or sending sync, data and control characters. The support for different types of communication facilities and control procedures may require different CLAs to be used because of different character alignment or data clocking arrangements.

In Figure 2-7, MLCP Transmit Functions, two noncontiguous data blocks are shown which must be transmitted with the header followed by the text in the line format shown at the bottom of the figure. The data blocks are in the form of communication data blocks and therefore can be described by two CCBs. Characters are taken in turn as required by the CLA transmit channel of the line. For example, H3 is taken into the MLCP as the next character. The MLCP processor using the CCP table examines the character, adds the character to the block check accumulation, and then gives it to the CLA for transmission. If D3 happened to be a "Y" internally, and the terminal representation of "Y" was "X", the MLCP CCP procedure may be designed to provide the editing transformation instead of requiring the CPU software to provide the scan of the data for this function.

In Figure 2-8, MLCP Receive Functions, data coming from the CLA to the MLCP must be edited, delimited, and transferred into the two distinct data blocks shown in main memory, via use of CCBs. The MLCP microprocessor, using its CCP table, examines the character, adds the character to the block check accumulation for a later status report, causes any required transition to the next CCB, and does any required editing or status reporting.

In addition to the ability of the MLCP to accommodate data transfers and related message delimiting and editing for data communication, the CLA interface for each communication line is controlled by the MLCP processor. These functions are stored in the LCT, and reported in the status, but in addition can be modified and controlled through the CCP instructions as required.

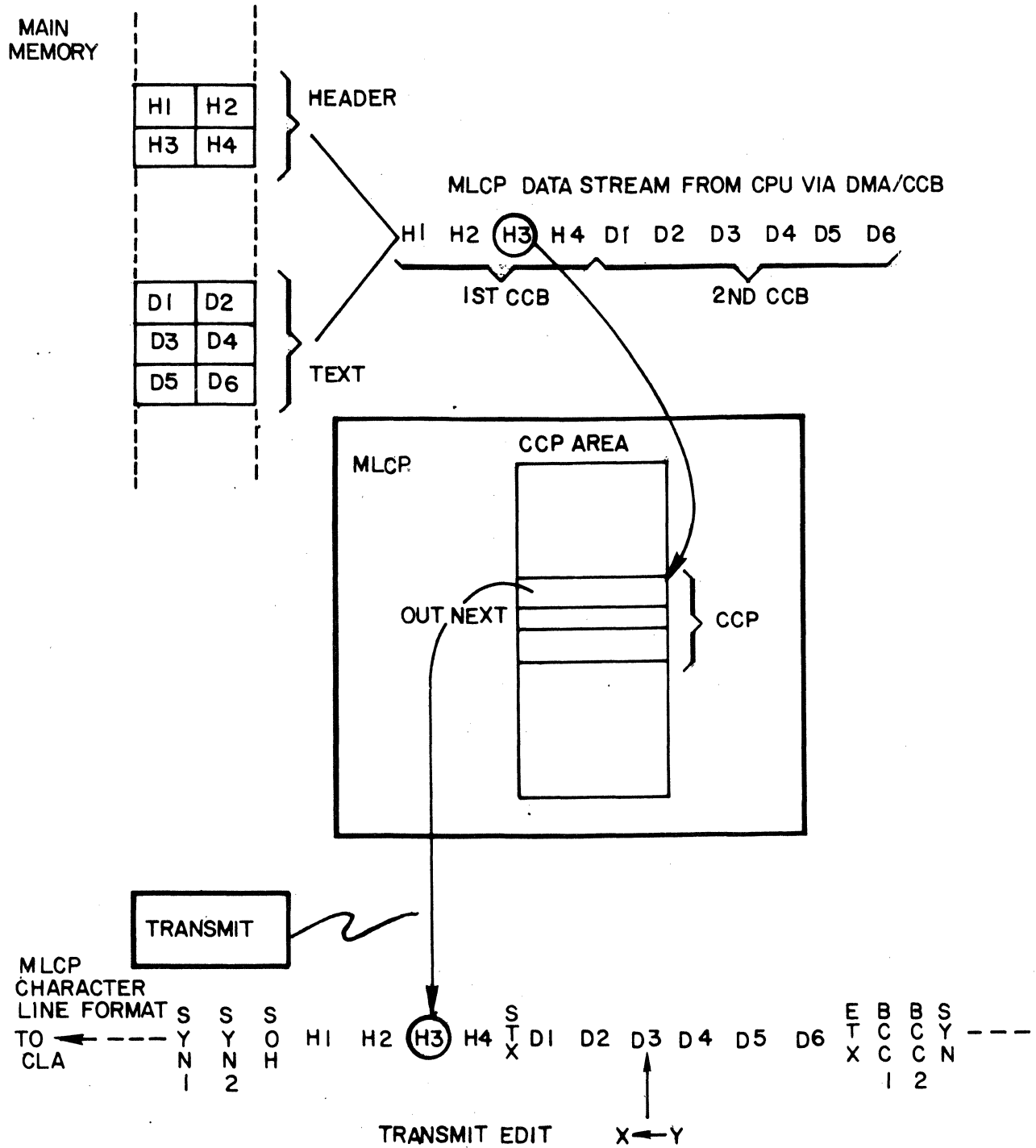


Figure 2-7 MLCP Transmit Functions

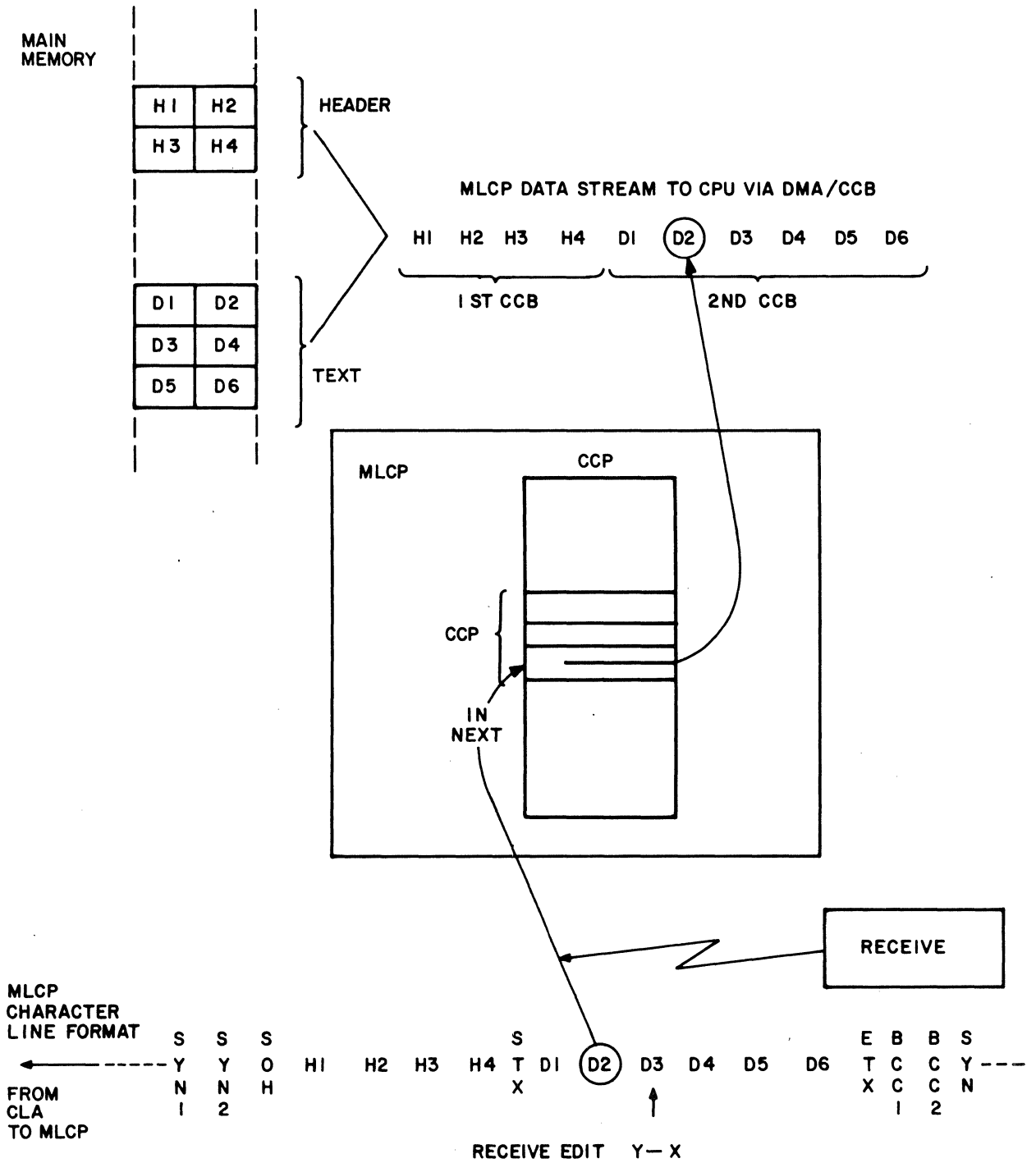


Figure 2-8 MLCP Receive Functions

2.3.2 Line Setup and Configuration

While channels are individually controlled, the typical requirement calls for setting up and configuring a line, or a pair of adjacent (even, odd) channels. These two channels must be configured as one transmit channel and one receive channel. Both channels in a given line must be of the same type, either synchronous or asynchronous.

Before being configured, the channels must be initialized via the Output Channel Control command with Channel Initialize. After initialization the channel control programs and line control tables are loaded with control and configuration parameters for the line. Channel control programs are loaded into the MLCP read/write RAM via a block mode write operation on the transmit channel. These CCPs are byte stream oriented and generally correspond to the byte-by-byte operations associated with analysis and identification of logical markers in the data stream.

CCPs are procedures for handling a data stream or some part of a data stream, with additional capability to manipulate and control status, communication control block transitions, communication devices attached to line adapters, line adapter configuration, etc.

The data stream to/from the line is manipulated by the channel control program, where special character detection/generation, CRC generation, residue accumulation, character sequence controls, data set control and character editing are performed. The channel control program utilizes the MLCP microprocessor to accomplish these functions.

The line control table areas of the read/write RAM contains line and line adapter configuration data, initial conditions, channel control program context and control, dynamic control, and status words for data transfer, and a channel control program work area for general application of the line. All basic line adapter configuration data and line operational data is stored in the line control table either during initialization or operation of the line. Line control tables contain pointers to the channel control programs being used by both the receive channel and the transmit channel. Line control tables are loaded by executing the Block Mode Write or the Output LCT Byte command.

Once the channel control programs and the line control tables are loaded, data transfer operations may begin.

2.3.3 Data Transfer

Data transfer on a channel may be set up once the line is configured for operation. This is done by loading one or more communications control blocks via the I/O commands. Communications control blocks are basically data block descriptors for data transfer operations. After one or more communications control blocks are loaded, the CPU may issue a Start I/O via the Channel Control instruction, which will initiate data transfer.

Each receive channel uses the CLA Channel Request interrupt to start the running of an input byte oriented channel control program. The channel control program extracts the byte from the line adapter, analyzes it, does any block check or CRC accumulation and check required, counts it, deletes it, makes decisions with regard to status, sequence detection, end of block, etc., then passes it on to the CPU via a Direct Memory Access (DMA) controlled by the communications control block. The channel control program suspends itself when it is finished processing a character.

Each transmit channel uses the CLA Channel Request interrupt to start the running of an output byte oriented channel control program. The channel control program takes the next byte given to it by the CCB-controlled DMA, analyzes it, adds it to the block check if required, counts it, substitutes or adds characters, then transfers the data to the line adapter.

As communications control blocks complete their transfers, CCBs configured to generate CPU interrupts indicate that status is complete. More CCBs can be loaded as desired.

The CPU can, at any time issue a Stop I/O command which will stop DMA transfers via the Megabus for that channel. When operation is complete, unused communications control blocks may be released from the MLCP by issuing a CCB List Reset command.

2.3.4 Pause Function

Firmware limits the number of contiguously executed CCP instructions to 31. If a specific channel attempts to execute more than 31 instructions, firmware generates a pseudo Wait (Pause) function which is transparent to the programmer and the CCP for all segments not containing timing loops. As a result, higher priority lines are given the chance to be serviced.

The pseudo Wait function differs from a normal Wait only in that the Channel Request Interrupt (CRI) from the channel is not reset. Therefore, if no channel of higher priority is requesting service, the current CCP is reentered experiencing only the delay of the context save and restore.

2.3.5 Data Transfer Clock

Data transfer for each channel relies on the availability of a clock to the line adapter.

For synchronous lines connected to DCE modems, this clock is generally supplied from the modem to the line adapter via the line adapter interface during normal modes of operation. During a test or a direct connect mode of operation, a data transfer clock supplied by the MLCP permits the line adapters to operate at the required bit transfer rates. This clock operates at a rate determined by the setting of a hex rotary switch, the output frequency being the product of a buad rate generator.

For asynchronous lines, the line adapter provides the clock for line adapter operation. The line adapters derive the actual data transfer clock for the channels and generate Channel Request Interrupts based on this clock.

HONEYWELL PROPRIETARY AND CONFIDENTIAL

Not all channels utilize the MLCP test/direct connect clock capability, as this feature is line adapter dependent.

2.3.6 Operational States (Figure 2-9)

The MLCP initialized state can be entered by depressing the system control panel MASTER CLEAR pushbutton, as a result of applying system power, or by a programmed master clear operation. The MLCP performs a soft initialize when the system control panel MASTER CLEAR pushbutton is depressed or when system power is applied. When initialized by a programmed Master Clear, a hard initialize is performed.

When a soft initialize is executed, the MLCP performs the following actions:

- Clears line register 2 of the line adapters, thus terminating all activity
- Clears LCT 8, 9, 40, and 41 of each channel
- Clears the MLCP control logic
- Runs a partial Quality Logic Test.

When a hard initialize is executed, the MLCP performs the following actions:

- Clears the line adapters
- Clears the MLCP control logic
- Runs the full Quality Logic Test
- Blocks MLCP interrupts (level = Zero)
- Clears the MLCP RAM
- Loads the current firmware revision level into absolute RAM address 1 (channel 0, LCT 1).

After the partial or full Quality Logic Test is executed, both the MLCP hardware and firmware enter an idle state.

In the firmware idle state, the firmware is running its Service Scan routine, which checks for line adapter requests or for the presence of an I/O command. Depending upon what the scan routine encounters, firmware may enter either the execution state or the background state.

The firmware execution state is the state in which I/O commands are decoded and line adapters are serviced. In the background state firmware checks for deferred DMA requests, and loads CCPs, CCBs, and LCTs, and checks for CLA status changes.

Once initiated, MLCP hardware enters its idle state. In this state, MLCP hardware remains quiescent until a request is received from the CP, after which the hardware enters the instruction execution state. Hardware-implemented commands are executed in this state. When the instruction includes a programmed master clear (function code = 01, bit one set), the hardware returns to the initialized state.

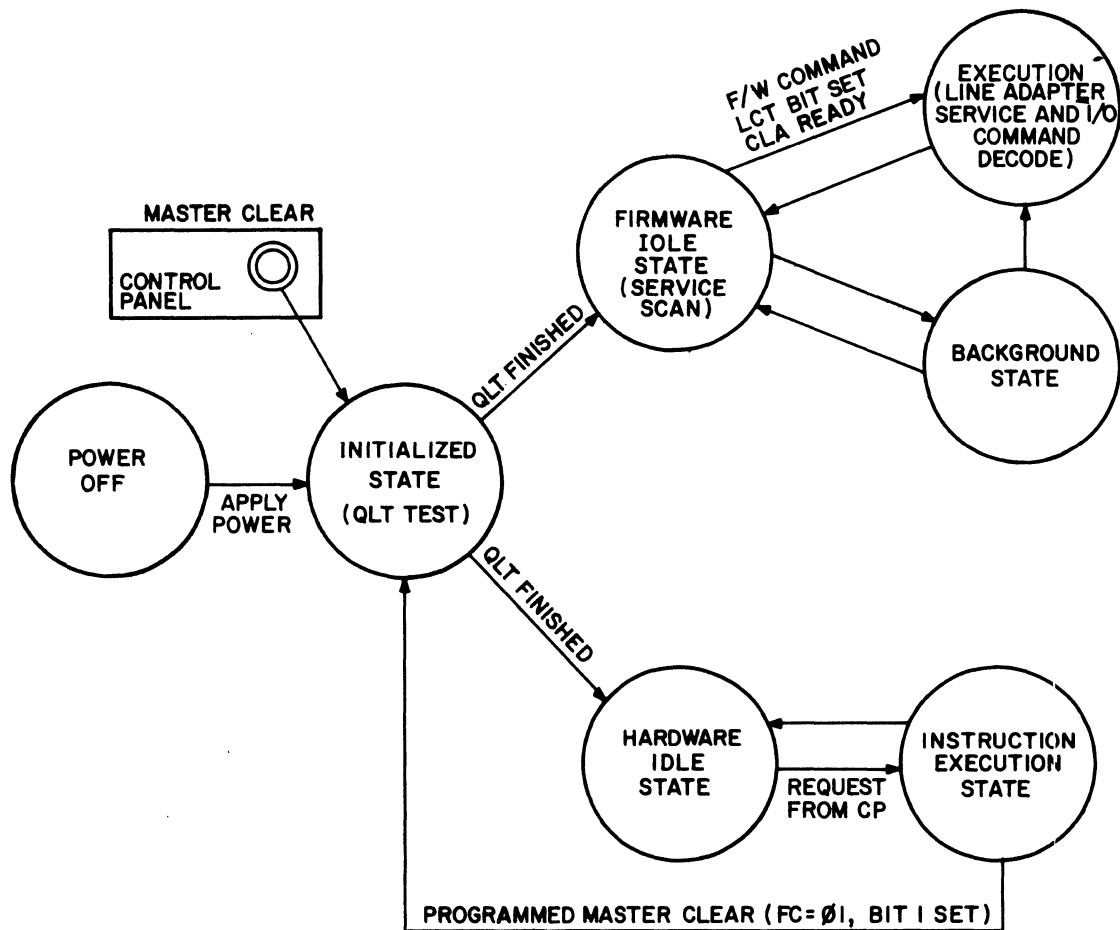


Figure 2-9 MLCP Operational States

2.3.7 Operational Modes

The MLCP has the two operational modes which are described in subsections 2.3.7.1 and 2.3.7.2.

2.3.7.1 Receive Mode

Incoming bit serial data from a channel configured to receive is converted by line specific CLAs into data elements of eight bits or less. Each time a data element is accumulated the CLA issues a Channel Request interrupt to the MLCP. This tells the MLCP that the CLA has a data element ready for transfer to the MLCP.

When the MLCP services this Channel Request interrupt, the MLCP must first load the CCP pointer and the program indicators into the MLCP microprocessor (do a context restore for this channel). The MLCP, using the CCP pointer as the next instruction location, then executes that CCP instruction on behalf of the channel. Since CCPs are loaded by the user, this MLCP channel is under complete user control at this point.

HONEYWELL PROPRIETARY AND CONFIDENTIAL

Typically the CCP will load the accumulated CLA data element, analyze it, and then take some specific action such as deleting it, converting it to some other data pattern, adding the data element to the CRC residue, or transmit it to main memory.

Storing a data element into the MLCP data buffer is essentially a request to transfer the data to the proper CDB via a CCB. The mechanics of this transfer is accomplished by MLCP hardware and is not visible to the CCP.

When the Channel Request interrupt has been serviced, the CCP relinquishes control for the next MLCP function by executing a Wait instruction.

The MLCP data buffer for each channel consists of two data elements in order to accommodate 16-bit DMA transfers on the Megabus.

Data transfers are monitored for various error conditions which might occur. These conditions are reported in the LCT and CCB status.

The CPU program communicates with the MLCP via the various channel-oriented I/O commands. Those commands which are associated with CCB setup, CCB execution, and CCB status input, control and monitor the data transfers. The actual data transfer is part of the CCB execution which uses a DMA Megabus transfer to fill a CCB-defined CDB in main memory.

2.3.7.2 Transmit Mode

Transmit mode is similar to the receive with the data flowing out instead of in. Line specific CLAs convert data elements of eight bits or less into Data Communications Equipment (DCE) compatible data (usually bit serial). Each time the CLA requires another data element, the CLA issues a Channel Request interrupt to the MLCP. This tells the MLCP that the CLA needs a data element for transmission.

When the MLCP services this Channel Request interrupt, the MLCP must first load the CCP pointer and the program indicators into the MLCP processor as in the receive mode.

Typically, the CCP loads the R-register with the next data element from the MLCP Data buffer. This data element was transferred from a CDB in main memory by a DMA operation during execution of a CCB, and placed in the MLCP data buffer (LCT 42 and 43) in anticipation of the CCP request. The loading of the data element into the R-register from the MLCP data buffer releases the storage for that data element in the buffer for use by additional data elements.

The CCP also has the option of loading R-register with CCP-generated data, and of extracting constants or data elements out of the LCT configuration data and LCT work area. In this way, special data elements, such as SYN, STX, ETX, BCC, FLAG, DLE, etc., can be inserted into the data stream by a CCP.

HONEYWELL PROPRIETARY AND CONFIDENTIAL

To output data to the CLA, the CCP issues an instruction which will transfer a data element in R-register into LR 1 of the CLA.

When the CCP has completed the operations for this Channel Request interrupt, the CCP relinquishes control for the next MLCP function by executing a Wait instruction.

The CLA accomplishes the data transmission of any data elements that it is given without any other MLCP action.

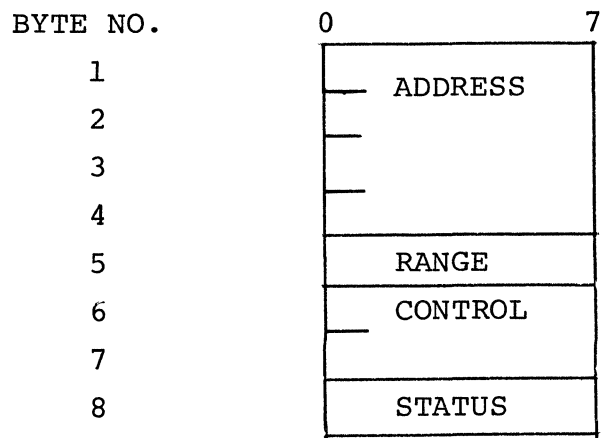
2.4 LINE CONTROL TABLE (LCT)

Each of the eight lines in the MLCP has assigned to it a unique and independent Line Control Table (LCT) in the MLCP read/write memory (RAM). All configuration, setup, status, and control information appears in the line control table during MLCP operation. The line control table is the key element in coordinating the MLCP operation on a line since this area is visible to the CPU programmer via I/O commands, to the MLCP programmer who generates procedures for the CCPs, and to the MLCP firmware.

Each line control table consists of a block of 64 contiguous bytes which can be broken down into the even (receive) channel and the odd (transmit) channel. The paired channels correspond to a line with one channel assigned for each direction of data transfer.

2.5 COMMUNICATION CONTROL BLOCKS (CCBs)

CCBs are defined and used by the MLCP to describe the address, range, control, and status for each block transfer between the MLCP and the rest of the system. Each CCB is set up through the use of I/O commands only. A CCB contains four basic elements of information in eight bytes as follows:



1. Address: the initial address is loaded into a CCB by the IOLD command. This 24-byte address is the starting address of a Communications Data Block (CDB) in main memory to be used in a MLCP/system data transfer.

2. Range: the initial range is also loaded into a CCB by the IOLD command. This 16-bit byte range defines the range count in bytes of the extent of a block or CDB which is to be read or written during the execution of this CCB.
3. Control: the control field is used to contain CCB specific control information.
4. Status: this area is used to store return status. The status word is set to Zero by execution of the IOLD command for this CCB.

For a detailed description of CCBs, refer to the MLCP Programmer's Reference Manual (Order Number AT97).

2.6 CHANNEL CONTROL PROGRAM (CCP)

A CCP is stored in the MLCP for application by one or more MLCP channels. The CCP pointer in the LCT (locations LCT 6 and 7 for receive, and LCT 38 and 39 for transmit) points to the next CCP location to be referenced by the channel when a CLA channel request interrupt must be serviced.

CCPs are lists of instructions used to process a channel's data communications character stream.

Each CCP is a procedure for handling a data stream or some part of a data stream, with additional capability to manipulate and control status, CCB transitions, DCE devices, CLA configuration, control, and status, and all related channel configurations. Each CCP contains a program of arbitrary length. The execution of a CCP is initiated by:

1. A channel request interrupt from the CLA (data driven).
2. A start I/O in the Channel command.
3. A data set control change condition for which the start CCP bit was set (bit three in LCT 8 on receive and LCT 40 on transmit).

For a detailed description of CCPs, refer to the MLCP Programmer's Reference Manual (Order Number AT97).

III THEORY OF OPERATION - INTERMEDIATE

3.1 HARDWARE MAJOR BLOCK DIAGRAM (Figure 3-1)

The MLCP hardware is composed of the major hardware functional areas summarized below and described in detail in the following subsections.

3.1.1 Functional Areas

3.1.1.1 Microprocessor Control Logic

This logic contains a control store which consists of a number of Programmable Read-Only Memories (PROMs) containing the firmware for the MLCP, a microprocessor, a next address generator for the control store, and a system clock which synchronizes MLCP operations.

3.1.1.2 Hardware Implemented I/O Command Logic

This logic is used to determine if the command to be executed by the MLCP is a hardware-implemented command (refer to list of commands in subsection 2.2.1) or a firmware-implemented command, to provide Megabus control signals, and to provide CCB address information to the read/write RAM.

3.1.1.3 Megabus Interface Register Logic

This logic contains the buffers and parity generator/checkers required to provide the MLCP with a data interface to the Megabus.

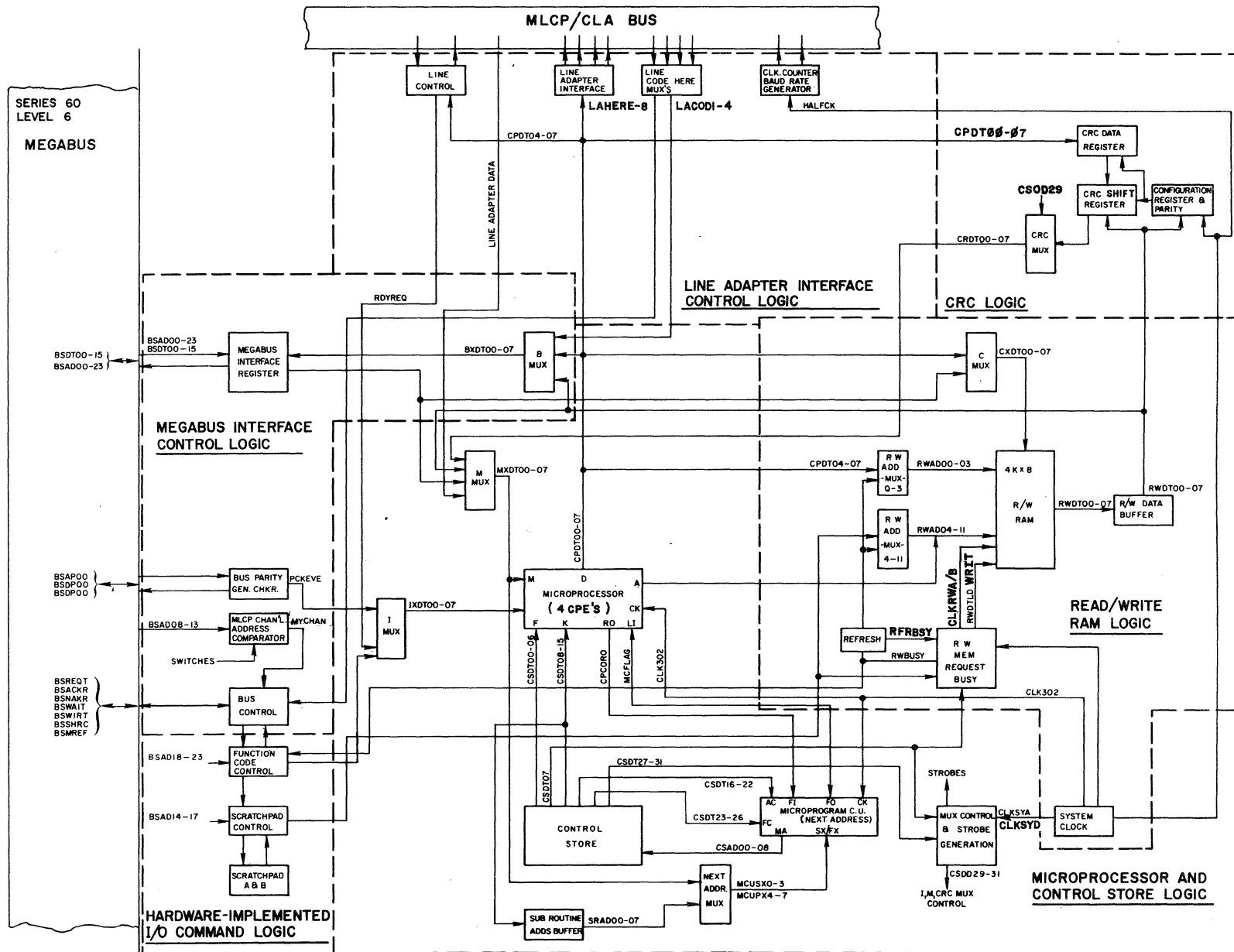


Figure 3-1 MLCP Hardware Major Block Diagram

3.1.1.4 Read/Write RAM

This logic contains storage for Channel Control Programs (CCP), Line Control Tables (LCT), and Communication Control Blocks (CCB), plus associated multiplexers, output buffering, and refresh circuitry.

3.1.1.5 Line Adapter Interface Control Logic

This logic interfaces the MLCP with up to four line adapters. It also includes control lines which tell the MLCP which function or instruction to perform, and address lines which tell the MLCP or line adapter which line is to perform the instruction.

3.1.1.6 CRC (Block Check) Logic

This logic, shared among all lines, is used to generate a CRC check character when that function is called for by the channel control program being run.

3.1.2 Data Flow and Control Paths

Data flow is normally through the microprocessor, which provides data to the read/write RAM, to the line adapters, to the Megabus (via the B-mux and the Megabus interface register), and to the CRC logic. Data from the Megabus interface register can be routed through the M-mux to the microprocessor, or through the C-mux to the read/write RAM. Multiplexer selection is determined by the state of certain bits in the control store word.

MLCP data paths are set up by the firmware resident in the control store. Each control store word includes a number of bits that are used by the MLCP to select multiplexer inputs, thereby establishing a unique data path for each microinstruction.

Example 1: Hardware-Implemented I/O Command

When data destined for the read/write RAM appears on the Megabus, the MLCP channel address comparator checks address bits 8 through 13 (BSAD08-BSAD13). If these bits contain the address of the MLCP, the MLCP checks itself to see if it is busy. If it is not busy, it returns an Acknowledge (ACK) signal to the Megabus, while at the same time the function code control logic checks address bits 18 through 23 (BSAD18-BSAD23) to determine the function code or command to be executed. Since this is a hardware-implemented command, information on the Megabus is loaded into the Megabus interface register. The Megabus interface register output is directed through the C-mux to the read/write RAM.

Example 2: Firmware-Implemented Command

During a firmware-executed command, address bits 8 through 13 and 18 through 23 are examined as described in Example 1. Since the instruction is to be executed by firmware, an input is generated to the I-mux. The firmware program is normally checking this I-mux bit. If this bit is true the firmware jumps to a routine to input data from the Megabus interface register to the M-mux. This data is

then sent to the microprocessor which determines the instruction to be executed. Firmware, having determined whether data in the Mega-bus interface register is to be written into the read/write RAM or whether data from the read/write RAM is to be loaded into the Mega-bus interface register, generates the request to the read/write RAM.

Example 3: Data Path to CRC Logic

The path from the microprocessor to the CRC logic is determined by the Channel Control Program instruction located in the read/write RAM. If the instruction calls for the generation of a CRC check character for the information on the microprocessor output data lines (CPDT00 through 07), that information will be loaded into the CRC data register, and from there through the CRC generator to develop the CRC check character. This check character is subsequently loaded into a line control table in the read/write RAM.

3.2 MICROPROCESSOR AND CONTROL STORE LOGIC

The microprocessor and control logic (Figures 3-2 and 3-3) consists of the following major elements:

1. A microprocessor consisting of an array of central processing elements
2. A microprogram control unit
3. A microprogram control store
4. A subroutine address buffer and multiplexer
5. A control store buffer
6. Strobe generation logic
7. MLCP multiplexers
8. Quality Logic Test (QLT) logic.

3.2.1 Microprocessor

Figure 3-3 is an intermediate level block diagram of the microprocessor and its inputs and outputs.

3.2.1.1 Central Processing Element

The microprocessor is composed of four 2-bit wide, central processing elements (CPEs). The data output of the microprocessor is sent to the B-mux, the C-mux, the R/W RAM address buffer, and the line adapter interface. The address output of the microprocessor is used to supply an address for the R/W RAM. The carry output from the microprocessor goes to the fast carry logic. The R output of the microprocessor is a right-shift output which goes to the microprogram control unit.

The I-mux and M-mux inputs to the microprocessor carry information to be used by the microprocessor. The F-input is used to specify the instruction and function to be performed by the microprocessor. The K-input is used to mask data on either the I- or on the

M-input. The LI-input is used for testing information from the microprogram control unit. The Clock input is used to clock the information. The CI-input is an input from the fast carry logic.

When four CPE chips are wired together in array, they provide the following functions:

1. Two's complement arithmetic
2. Logic AND, OR, NOT, and exclusive-OR
3. Incrementing and decrementing
4. Shifting left or right
5. Bit testing and zero detection
6. Multiple data and address buses.

3.2.1.2 Input and Output Functions (Figures 3-3 and 3-4)

The microprocessor input and output functions are summarized in Table 3-1.

3.2.1.3 Microfunction Decoder and Scratch Pad Registers

The function to be performed by the microprocessor is specified by the coding of control store word bits CSDT00 through CSDT06. These bits are decoded by the function decoder in the microprocessor. Bits CSDT00 through CSDT02 specify the function and select the operands for the Arithmetic Logic Section (ALS) by gathering them through the A- and B-multiplexers. Bits CSDT03 through CSDT06 select the desired scratch pad register (R0 through R9 and T) or Accumulator register (AC-register). The result of the ALS operation is deposited in either the AC-register or written into the selected scratch pad register, and where specified, developed address data is deposited into the Memory Address Register (MAR).

The decoded control store bits CSDT00 through CSDT02 define eight function groups (F-groups) as shown in Table 3-2. The decoded control store bits CSDT03 through CSDT06 select the scratch pad register as shown in Table 3-3. The scratch pad registers (and the AC-register) are organized into register groups (R-groups) 1, 2, and 3. The function performed by the microprocessor is a resultant of the particular combination of the F-group and R-group selected. A summary of functions is shown in Table 3-4.

Finally, the performance of the microprocessor is influenced by the clock pulse CLK302-00. This clock pulse is selectively gated into the microprocessor, and may therefore be omitted during a cycle. Since the carry, shift, and look-ahead circuits are not clocked, their outputs can be used to perform a number of nondestructive tests on data in the AC-register or in the scratch pad registers. No register contents are modified by the operation due to the absence of the clock pulse.

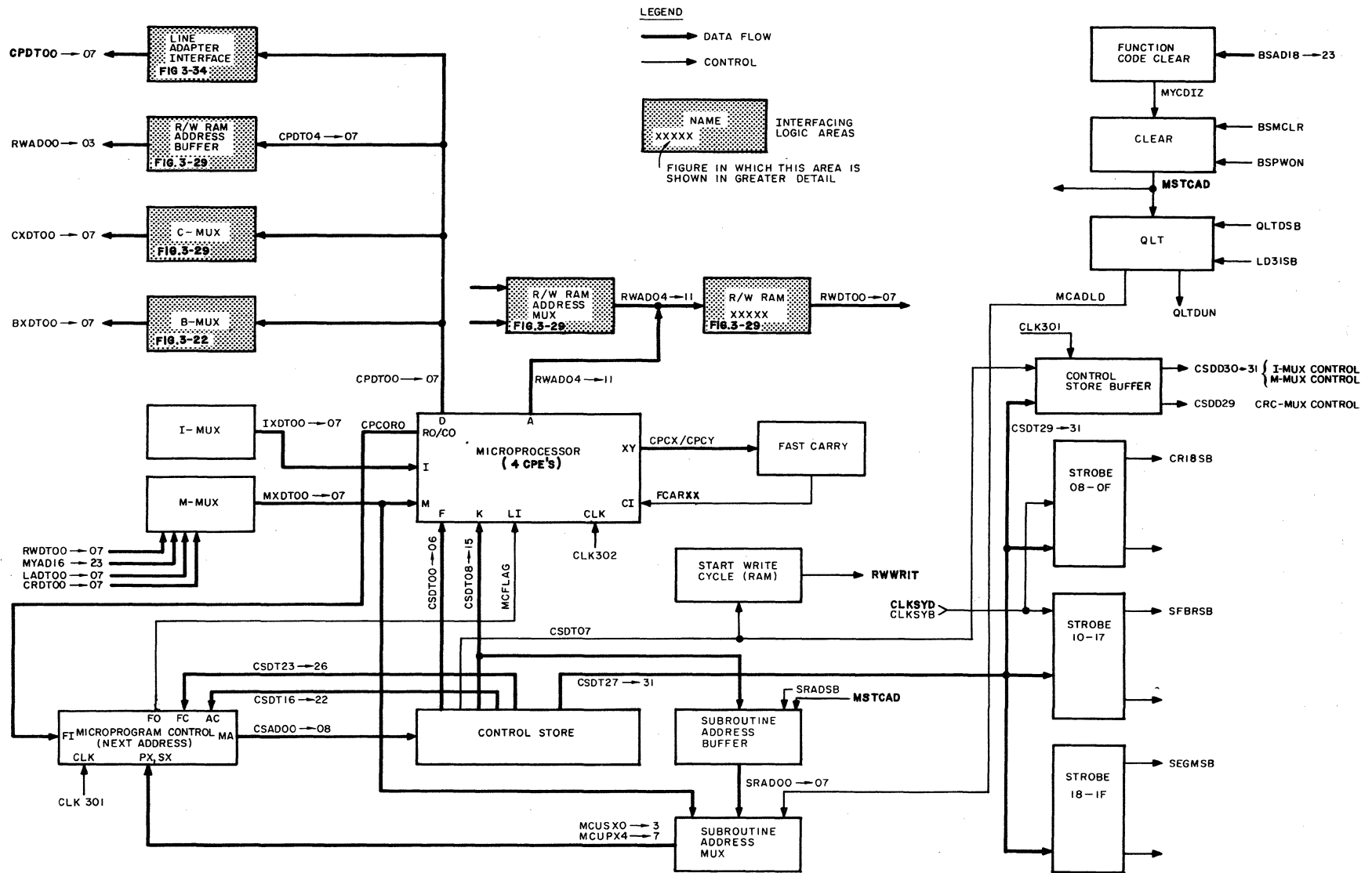


Figure 3-2 Microprocessor and Control Store Logic Functional Block Diagram

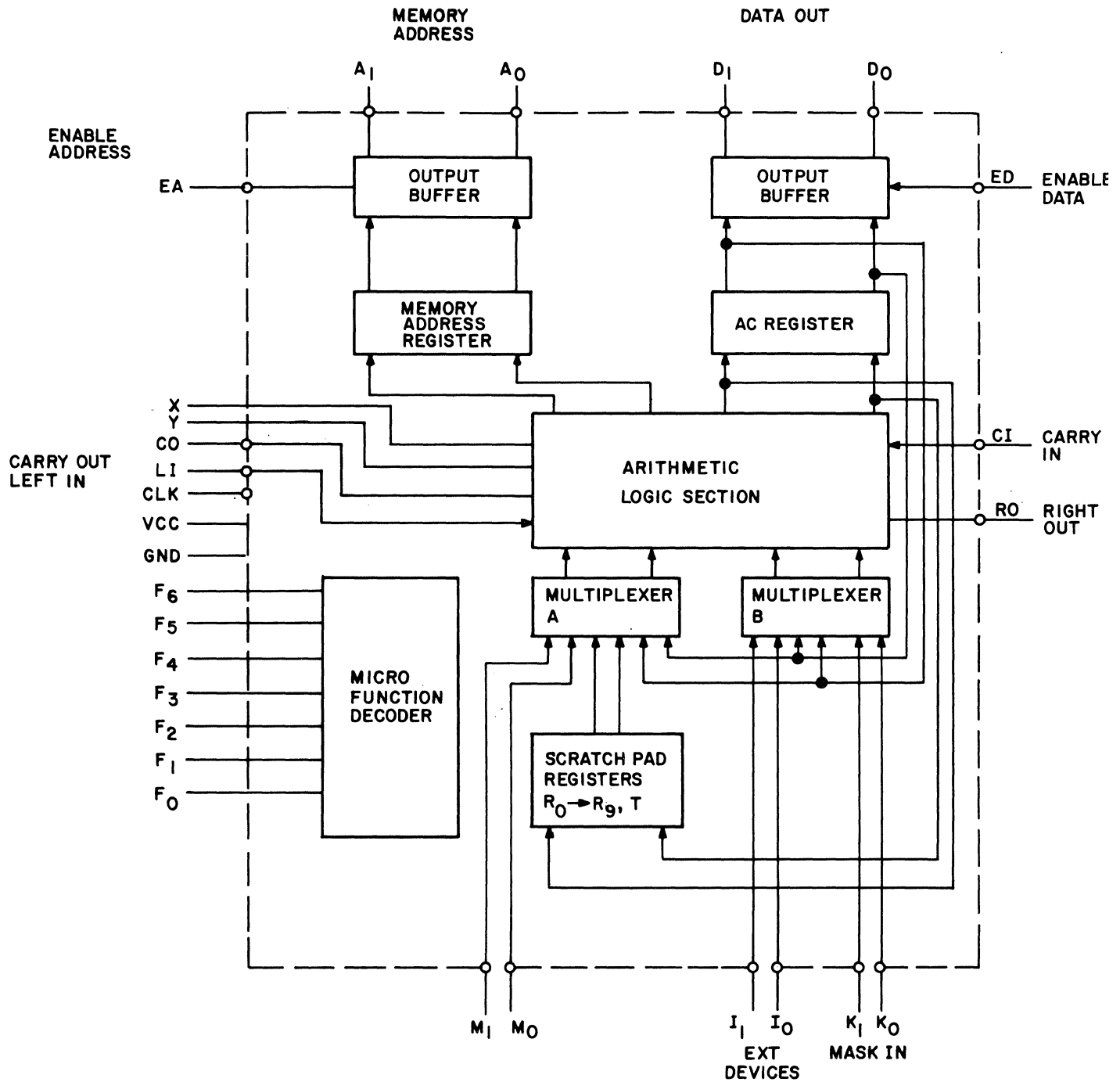


Figure 3-3 Microprocessor Intermediate Block Diagram

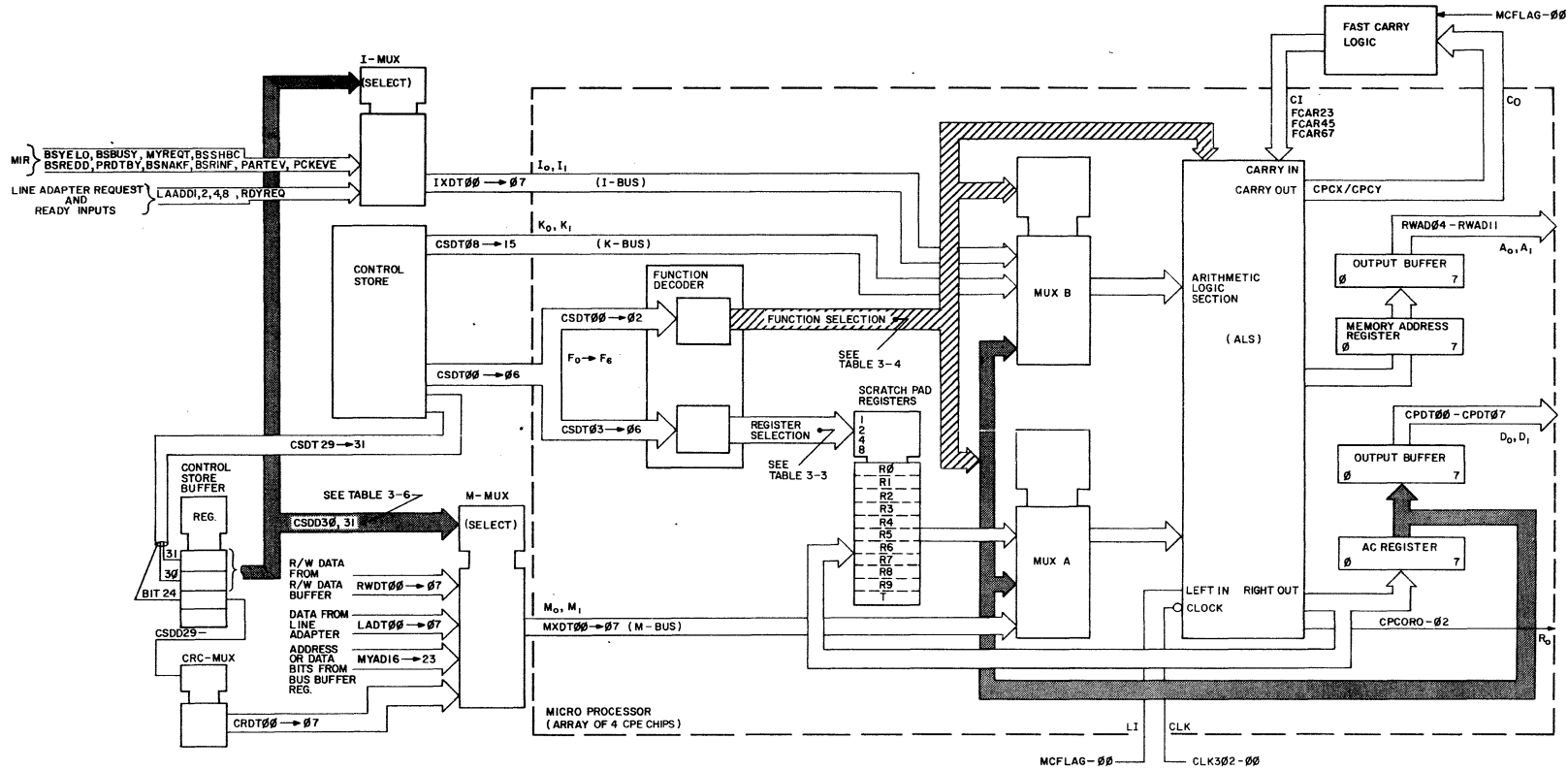


Figure 3-4 Microprocessor CPE Chip Functional Block Diagram

HONEYWELL PROPRIETARY AND CONFIDENTIAL

Table 3-1 Microprocessor Input/Output Functions

INDIVIDUAL CHIP		NAME, FUNCTION NAMES, AND DESCRIPTION	TYPE*
PIN	SYMBOL		
1,2	I_0-I_1	<u>I-Bus Data.</u> (IXDT00 thru IXDT07) I-bus inputs provide a separate input port for line adapter inputs and parity and MIR signals.	Active Low
3,4	K_0-K_1	<u>K-Bus inputs.</u> (CSDT08 thru CSDT15) These inputs provide a separate input port for the microprogram control store to allow mask or constant entry.	Active Low
5,6 7	X, Y CO	<u>Carry Look-Ahead Cascade outputs.</u> <u>Ripple Carry Output.</u> (CPCX/CPCY) This output is disabled only during shift right operations.	Active Low Three-State
8	RO	<u>Shift Right Output.</u> (CPCORO-02) This output is enabled only during shift right operations.	Active Low Three-State
9	LI	<u>Shift Right Input.</u> (MCFLAG-00)	Active Low
10	CI	<u>Carry Input.</u> (FCAR23/FCAR45/FCAR67)	Active Low
11	EA	<u>Memory Address Enable Input.</u> In the low state, this input enables the memory address outputs A_0-A_1 , i.e., output functions RWAD04 thru RWAD11.	Active Low
12-13	A_0-A_1	<u>Memory Address Bus Outputs.</u> (RWAD04 thru RWAD11) Buffered outputs of the Memory Address Register (MAR).	Active Low Three-State
14	GND	<u>Ground</u>	
15-17 24-27	F_0-F_6	<u>Microfunction Bus Inputs.</u> (CSDT00 thru CSDT06) These inputs control the microprocessor function and scratch pad register selection.	
18	CLK	<u>Clock Input.</u>	
19-20	D_0-D_1	<u>Memory Data Bus Outputs.</u> (CPDT00 thru CPDT07) These outputs are the buffered outputs of the full function Accumulator (AC) register.	Active Low Three-State
21-22	M_0-M_1	<u>Memory Data Bus Inputs.</u> (MXDT00 thru MXDT07) These inputs provide a separate input port for the selected M-Mux input (i.e., information from the R/W data buffer, bus buffer register, line adapter, or CRC generator.)	Active Low
23	ED	<u>Memory Data Enable Input.</u> In the low state, this input enables the data bus outputs (CPDTXX: permanently enabled in MLCP).	Active Low
28	V_{CC}	<u>+5 Volt Supply.</u>	

*Active high unless otherwise specified.

Table 3-2 Microprocessor
Function Selection

CONTROL STORE BIT			F-GROUP
00	01	02	
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

Table 3-3 Scratch Pad Register Selection*

CONTROL STORE BIT				REGISTER SELECTED	REGISTER GROUP
03	04	05	06		
0	0	0	0	R0	1
0	0	0	1	R1	
0	0	1	0	R2	
0	0	1	1	R3	
0	1	0	0	R4	
0	1	0	1	R5	
0	1	1	0	R6	
0	1	1	1	R7	
1	0	0	0	R8	
1	0	0	1	R9	2
1	0	1	0	T	
1	0	1	1	AC	
1	1	0	0	T	1
1	1	0	1	AC	
1	1	1	0	T	3
1	1	1	1	AC	

*Note: See Table 4-1 for register applications

HONEYWELL PROPRIETARY AND CONFIDENTIAL

Table 3-4 Microprocessor Function Summary

F-GROUP (CSDT00-CSDT02)	R-GROUP (CSDT03-CSDT06)	MICROFUNCTION
0	1	$R_n + (AC \wedge K) + CI \rightarrow R_n, AC$
	2	$M + (AC \wedge K) + CI \rightarrow AT$
	3	$AT_L \wedge (I_L \wedge K_L) \rightarrow RO$ $LI \vee [(I_H \wedge K_H) \wedge AT_H] \rightarrow AT_H$ $[AT_L \wedge (I_L \wedge K_L)] \vee [AT_H \wedge (I_H \wedge K_H)] \rightarrow AT_L$
1	1	$K \vee R_n \rightarrow MAR$ $R_n + K + CI \rightarrow R_n$
	2	$K \vee M \rightarrow MAR$ $M + K + CI \rightarrow AT$
	3	$(\overline{AT} \vee K) + (AT \wedge K) + CI \rightarrow AT$
2	1	$(AC \wedge K) - 1 + CI \rightarrow R_n$
	2	$(AC \wedge K) - 1 + CI \rightarrow AT$
	3	$(1 \wedge K) - 1 + CI \rightarrow AT$

(see Note 1)

- NOTE
- 2's complement arithmetic adds 111...11 to perform subtraction of 000...01.
 - R_n includes T and AC as source and destination registers in R-group 1 microfunctions.
 - Standard arithmetic carry output values are generated in F-group 0, 1, 2, and 3 instructions.

LEGEND

SYMBOL	MEANING	MICROPROCESSOR INPUT OR OUTPUT FUNCTION
I	Data on the I-Bus	IXDT00-IXDT07
K	Data on the K-Bus	CSDT08-CSDT15
M	Data on the M-Bus	MXDT00-MXDT07
CI	Data on the Carry Input	FCAR23/45/67
CO	Data on the Carry Output	CPCX/CPCY
LI	Data on the Left Input	MCFLAG-00
RO	Data on the Right Output	CPCORO-02
R_n	Content of Register n including T and AC	-----
AC	Content of the AC Register	CPDT00-CPDT07
AT	Content of AC or T as specified	-----
MAR	Content of Memory Address Register	RWAD04-RWAD11
L H	Subscripts designating low and high order bits.	-----
+	2's complement addition	-----
-	2's complement subtraction	-----
\wedge	Logical AND	-----
\vee	Logical OR	-----
\oplus	Exclusive NOR	-----
+	Deposit into	-----

3.2.1.4 M-Bus and I-Bus Inputs (M-, I-, and CRC Mux's)

The M-bus inputs are arranged to bring data from the R/W RAM, Megabus interface register, CRC logic, or from a line adapter into the microprocessor. These inputs arrive on the M-bus via the M-mux, gated through an exclusive OR gate. Data on the M-bus is multiplexed by the microprocessor A-multiplexer for input to the ALS.

The I-bus inputs are arranged to bring information from the Megabus interface control logic or a line adapter ready or request into the microprocessor. These inputs arrive on the I-bus via the I-mux. Data on the I-bus is also multiplexed within the microprocessor, although independently of the M-bus, for input to the ALS.

Control over the M- and I-multiplexers is provided by control store bits CSdT30 and CSdT31. Due to timing constraints, these two bits cannot select the multiplexer inputs in time for them to be effective for the microinstruction in which the bits occur. These bits are therefore latched into the control store buffer for use during a subsequent microinstruction. The multiplexer inputs selected by these control store bits are defined in Table 3-5.

One input to the M-mux is from the CRC mux, whose inputs are selected by control store bit CSdT29. This bit is similarly latched into the control store buffer and used during a subsequent microinstruction to select either bits one through eight or 9 through 16 of the CRC shift register for transfer to the M-mux. This input to the M-mux is selected during block check operations and is used as the transfer path by which a developed CRC check character is stored in a selected line control table in the R/W RAM.

3.2.1.5 Accumulator Register

As independent register called the Accumulator (AC) register is available for storing the result of an ALS operation. The output of the AC-register is multiplexed within the microprocessor for input back to the ALS and is also available via a three-state output buffer on the CP data bus (CPDT00 through CPDT07).

3.2.1.6 A- and B-Multiplexers

The A- and B-multiplexers select the two inputs to the ALS specified by control store bits CSdT00 through CSdT02. Inputs to the A-multiplexer include the AC-register, the scratch pad registers and the M-mux, while inputs to the B-multiplexer include the AC-register, the K-bus (control store bits 8 through 15) and the I-bus. The selected B-multiplexer input is always logically ANDed with the data on the K-bus (refer to subsection 3.2.1.7) to provide a flexible masking and bit testing capability. Tables 3-4 and 3-6 reflect multiplexer input selection within the microfunction equations.

3.2.1.7 Arithmetic Logic Section and K-Bus

The ALS is capable of a variety of arithmetic and logic operations, including two's complement addition and subtraction, incrementing, decrementing, logical AND, inclusive-OR, exclusive-NOR, and logical complement. The result of an ALS operation can be stored in

the AC-register or one of the scratch pad registers. Separate left input and right output lines, designated LI and RO, are available for use in right shift operations. Carry input and carry output lines, designated CI and CO, are provided for normal ripple carry propagation. CO and RO data are brought out via alternately enabled tristate buffers. In addition, standard look-ahead carry outputs, designated X and Y, are available for full carry look-ahead across any word length.

The ability of the K-bus to mask inputs to the ALS greatly increases the versatility of the microprocessor. During nonarithmetic operations in which carry propagation has no meaning, the carry circuits are used to perform a word-wise inclusive-OR of the bits, masked by the K-bus, from the register or bus selected by the function decoder. Thus, the microprocessor has a flexible bit-testing capability. The K-bus is also used during arithmetic operations to mask portions of the field being operated upon. An additional function of the K-bus is that of supplying constants to the microprocessor from the microprogram resident in the control store.

3.2.1.8 Memory Address Register and Bus

A separate ALS output is also available to the Memory Address register (MAR) and bus via a three-state output buffer (RWAD04 through RWAD11). This buffer provides the address when the microprocessor is accessing R/W RAM.

Table 3-5 M-, I-, and CRC Mux Input Selection

CONTROL STORE BIT			M-MUX INPUT	I-MUX INPUT	CRC MUX INPUT
29	30	31			
X	0	0	LADTXX+10 (0-7)	LAADDX(4) 1, 2, 3, 8/RDYREQ, BSRINF, PARTEV	-
X	0	1	MYAD16-23	-	-
X	1	0	CRDTXX+00 (0-7)	BSNAKF, BSBUSY, PRDTB4, MYREQT/BSYELD, BSSHBC, PCKEVE, BSREDD	-
X	1	1	RWDTXX+10 (0-7)	-	-
0	X	X	-	-	CRCR01-08
1	X	X	-	-	CRCR09-16

NOTE: To load the Control Store Buffer CSDT07 must = 1 and CSDT28 must = 0.

Table 3-6 Microprocessor Function with K-Bus = 0000000 or 1111111

F GROUP (CSDT00-CSDT02)	R GROUP (CSDT03-CSDT07)	K-BUS = 0000000 FUNCTION	MNEMONIC	K-BUS = 1111111 FUNCTION	MNEMONIC
0	1	$R_n + CI \rightarrow R_n, AC$	ILR	$AC + R_n + CI \rightarrow R_n, AC$	ALR
	2	$M + CI \rightarrow AT$	ACM	$M + AC + CI \rightarrow AT$	AMA
	3	$AT_L \rightarrow RO \quad AT_H \rightarrow AT_L \quad LI \rightarrow AT_H$	SRA	-	-
1	1	$R_n \rightarrow MAR \quad R_n + CI \rightarrow R_n$	LMI	$11 \rightarrow MAR \quad R_n - 1 + CI \rightarrow R_n$	DSM
	2	$M \rightarrow MAR \quad M + CI \rightarrow AT$	LMM	$11 \rightarrow MAR \quad M - 1 + CI \rightarrow AT$	LDM
	3	$\overline{AT} + CI \rightarrow AT$	CIA	$AT - 1 + CI \rightarrow AT$	DCA
2	1	$\left. \begin{array}{l} CI - 1 \rightarrow R_n \\ CI - 1 \rightarrow AT \end{array} \right\} \text{ See Note 1}$ (See CSA above)	CSR	$\left. \begin{array}{l} AC - 1 + CI \rightarrow R_n \\ AC - 1 + CI \rightarrow AT \end{array} \right\} \text{ See Note 1}$ $I - 1 + CI \rightarrow AT$	SDR
	2		CSA		SDA
	3		-		LDI
3	1	$R_n + CI \rightarrow R_n$	INR	$AC + R_n + CI \rightarrow R_n$	ADR
	2	(See ACM above)	-	(See AMA above)	-
	3	$AT + CI \rightarrow AT$	INA	$I + AT + CI \rightarrow AT$	AIA
4	1	$CI \rightarrow CO \quad 0 \rightarrow R_n$	CLR	$CI \vee (R_n \wedge AC) \rightarrow CO \quad R_n \wedge AC \rightarrow R_n$	ANR
	2	$CI \rightarrow CO \quad 0 \rightarrow AT$	CLA	$CI \vee (M \wedge AC) \rightarrow CO \quad M \wedge AC \rightarrow AT$	ANM
	3	(See CLA above)	-	$CI \vee (AT \wedge I) \rightarrow CO \quad AT \wedge I \rightarrow AT$	ANI
5	1	(See CLR above)	-	$CI \vee R_n \rightarrow CO \quad R_n \rightarrow R_n$	TZR
	2	(See CLA above)	-	$CI \vee M \rightarrow CO \quad M \rightarrow AT$	LTM
	3	(See CLA above)	-	$CI \vee AT \rightarrow CO \quad AT \rightarrow AT$	TZA
6	1	$CI \rightarrow CO \quad R_n \rightarrow R_n$	NOP	$CI \vee AC \rightarrow CO \quad R_n \vee AC \rightarrow R_n$	ORR
	2	$CI \rightarrow CO \quad M \rightarrow AT$	LMF	$CI \vee AC \rightarrow CO \quad M \vee AC \rightarrow AT$	ORM
	3	(See NOP above)	-	$CI \vee I \rightarrow CO \quad I \vee AT \rightarrow AT$	ORI
7	1	$CI \rightarrow CO \quad \overline{R_n} \rightarrow R_n$	CMR	$CI \vee (R_n \wedge AC) \rightarrow CO \quad R_n \oplus AC \rightarrow R_n$	XNR
	2	$CI \rightarrow CO \quad \overline{M} \rightarrow AT$	LCM	$CI \vee (M \wedge AC) \rightarrow CO \quad M \oplus AC \rightarrow AT$	XNM
	3	$CI \rightarrow CO \quad \overline{AT} \rightarrow AT$	CMA	$CI \vee (AT \wedge I) \rightarrow CO \quad I \oplus AT \rightarrow AT$	XNI

Note 1: 2's complement arithmetic adds 111...11 to perform subtraction of 000...01.

3.2.2 Control Multiplexers

The MLCP contains a number of controllable multiplexers used to establish data paths among the different hardware functional areas. Because these multiplexers are utilized in various operations and by various hardware elements (such as the microprocessor, control store, etc.), specific applications are described in the appropriate subsections of this manual. For reference purposes, all MLCP multiplexers are identified in Table 3-7, together with their input and output function names, select line names, and select line coding.

3.2.3 Fast Carry Logic (Figure 3-5)

The fast carry logic consists of three high-speed look-ahead generators which anticipate the carry across the array of four CPE chips which comprise the microprocessor.

3.2.4 Microprogram Control Store

The microprogram control store is an array of programmable read-only memories containing the firmware for the MLCP. Control store output functions are summarized in Table 3-8. Figure 3-6 portrays the control store word format. Details regarding specific usage of control store bits are provided in the subsections and Tables referenced in Table 3-8. Figure 3-7 illustrates the control store logic.

3.2.5 Microprogram Control Unit (Figure 3-8)

3.2.5.1 MCU Element

The Microprogram Control Unit (MCU) controls the sequence in which microinstructions are fetched from the control store. Basically, its functions include the following:

1. Maintenance of its own internal microprogram address register.
2. Selection of the next microinstruction in the control store (based on the content of the microprogram address register).
3. Decoding and testing the data supplied via input buses from the control store and from the address multiplexer in order to determine the microinstruction execution sequence.
4. Saving and testing the carry output data from the microprocessor.
5. Control of carry/shift input data to the microprocessor.

The MCU performs two major control functions. First, it controls the sequence in which microinstructions are fetched from control store. For this purpose, the MCU contains a Microprogram Address Register (MAR) and the associated logic for selecting the next microinstruction address. Second, it controls the two flag flip-flops that are included for interaction with the carry input and carry output logic of the microprocessor. A functional block diagram of the MCU chip is shown in Figure 3-8.

HONEYWELL PROPRIETARY AND CONFIDENTIAL

Table 3-7 MLCP Control Multiplexers

MULTIPLEXER NAME OUTPUT SIGNAL NAMES	INPUT SIGNAL NAMES	SELECT LINES
<p align="center"><u>B-MUX</u></p> <p>(Megabus Interface Register Multiplexer) BXDT00 through BXDT07</p>	<p>CPDT00+ through CPDT07+ LACOD1-4 PLUPAA/ZGND RWDT00+ through RWDT07+</p>	<p><u>BXSELB</u> <u>BXSELA</u></p> <p>0 0 0 1 1 0 1 1</p>
<p align="center"><u>C-MUX</u></p> <p>(R/W RAM Data Mux) CXDT00 through CXDT07</p>	<p>CPDT00+ through CPDT07+ MYAD16+ through MYAD23+</p>	<p><u>IOSELT</u></p> <p>1 0</p>
<p align="center"><u>CRC-MUX</u></p> <p>(Block Check Logic Mux) CRDT00 through CRDT07</p>	<p>CRCR01 through CRCR08 CRCR09 through CRCR16</p>	<p><u>CSD29</u></p> <p>0 1</p>
<p align="center"><u>CLA ADD MUX</u></p> <p>(Line Adapter Address) LAADDn(1,2,4,8)</p>	<p>RDYADn(1,2,4),RDYRQ1 (Line Adapter Ready) CPDT00 through CPDT03 (CP Data)</p>	<p><u>CSDT07</u> <u>LAADSB</u></p> <p>0 1 1 1</p>
<p align="center"><u>I-MUX</u></p> <p>(Microprocessor Input Mux) IXDT00 through IXDT07</p>	<p>LAADDn(4)1,2,4,8/RDYREQ,BSRINF,PARTEV BSNAKF,BSBUSY,PRDTB4,BSYELO,BSREDD, MYREQT/BSSHBC,PCKEVE</p>	<p><u>CSDT31</u> <u>CSDT30</u></p> <p>0 0 0 1</p>
<p align="center"><u>M-MUX</u></p> <p>(Memory Data Mux) MXDT00 through MXDT07</p>	<p>RWDT00 through RWDT07 MYAD16 through MYAD23 (Bus Bfr. Ref. 0-7) CRDT00 through CRDT07 LADT00 through LADT07</p>	<p><u>CSDT31</u> <u>CSDT30</u></p> <p>1 1 1 0 0 1 0 0</p>
<p align="center"><u>NEXT ADD MUX</u></p> <p>(Next Address Logic Mux) MCUSX0 - MCUSX3 and MCUPX4 - MCUPX7</p>	<p>MXDT00 through MXDT07 SRAD00 through SRAD07</p>	<p><u>MCADLD</u></p> <p>0 1</p>
<p align="center"><u>LA COD MUX</u></p> <p>(Line Adapter Code Mux) LACOD1 through LACOD4</p>	<p>LACOD1(4) through LACOD4(4) from CLAs 1-4</p>	<p><u>SCPAD3</u> <u>SCPAD2</u></p> <p>See Figure 3-35</p>
<p align="center"><u>LA HERE MUX</u></p> <p>(Line Adapter Here Mux) LAHERE</p>	<p>LAHERE-01 through 08 from CLAs 1-4</p>	<p><u>BSAD14</u> <u>BSAD15</u> <u>BSAD16</u></p> <p>See Figure 3-35</p>
<p align="center"><u>CCB ADD MUX</u></p> <p>(R/W RAM Address Bits) CBAD07 and CBAD08</p>	<p>STCNT1 and STCNT2 LDCNT1 and LDCNT2</p>	<p><u>PRAD23</u></p> <p>0 1</p>
<p align="center"><u>R/W ADD MUX</u></p> <p>(R/W RAM Address Mux) RWAD00 through RWAD11</p>	<p>RFAD00 through RFAD05 PLUPAB,SCPAD0-3,CBAD07-08,PRDTB0-2</p>	<p><u>CPRWBS</u>¹ <u>RFBUSY</u>²</p> <p>0 0 0 1</p>
<p align="center"><u>XOR MUXES (3)</u></p> <p>(CRC Exclusive-OR Muxes) DTCR08 and 15,SCRC12,SCCITT/ DTEX01-03,05,12, and 15, DTCR06 and 10.</p>	<p>CRCR15,EXOR16/CRCR05 and 12,EXOR02 and 15/CRCR01,03,06 and 10. DTEX15,CRCR08,SLCRC0 and C1/EXOR05 and 12,CRCR02 and 15/EXOR01 and EXOR03,CRCR06</p>	<p><u>SLRC08</u> <u>SCCITT</u> <u>SCRC12</u></p> <p>0 0 0 1 1 1</p>
<p align="center"><u>CRC SHREG MUXES (4)</u></p> <p>(CRC Shift Register Muxes) CRCR01 through CRCR16</p>	<p>EXOR16,DTEX01-03/CRCR04,06,07,DTEX05/ DTCR08,10,CRCR09,11/DTEX12,CRCR13,14 DTCR15 RWDT00-03 through RWDT07</p>	<p><u>SHECRC</u> <u>CL108R</u> <u>CL916R</u></p> <p>0 1 1 1 1 1</p>

NOTE: 1. CPRWBS = CP Memory Busy
2. RFBUSY = Refresh Memory Busy.

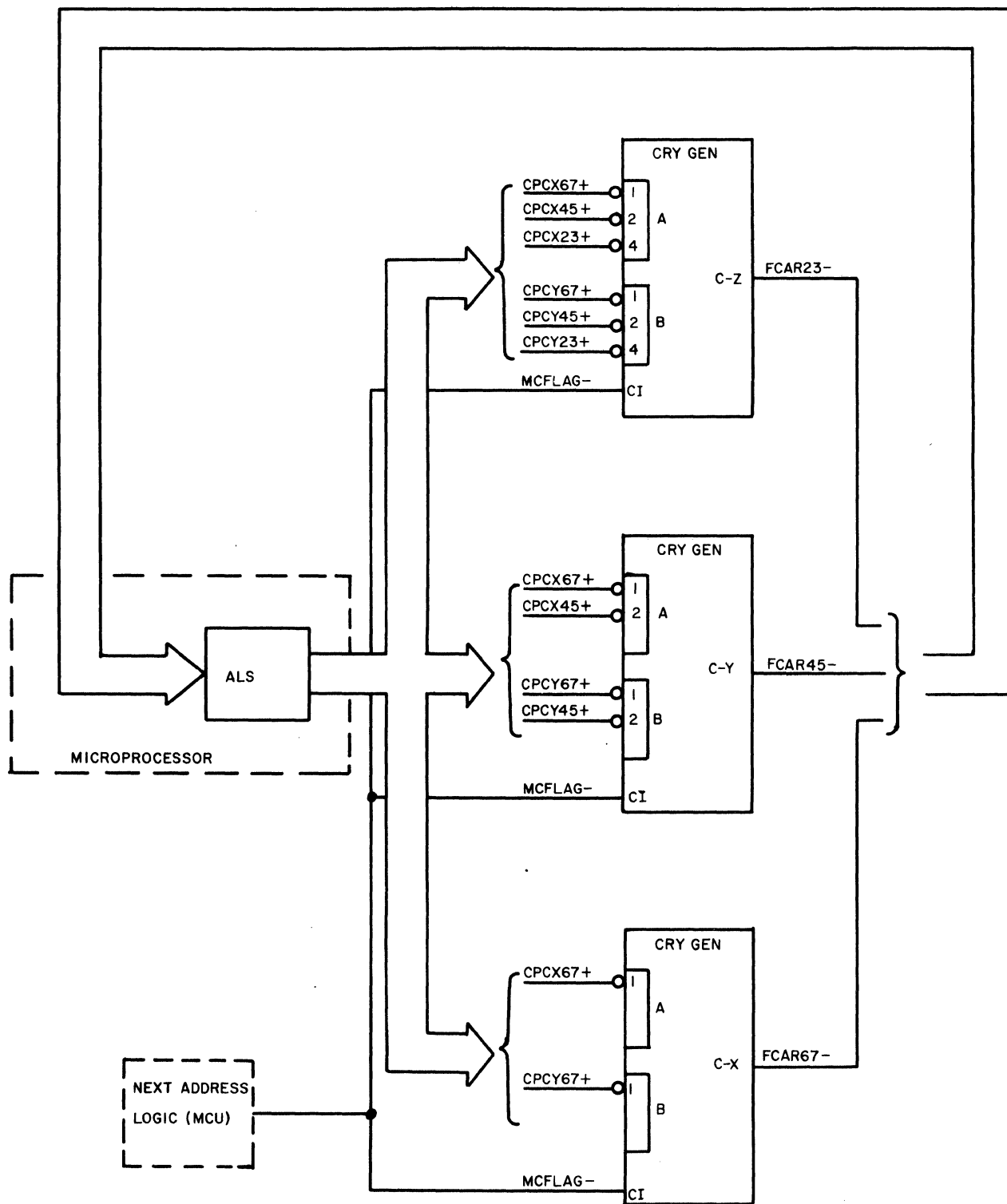


Figure 3-5 Fast Carry Logic

Table 3-8 Control Store Output Functions

CONTROL STORE BITS CSDTXX	FUNCTION	FOR DETAILED DESCRIPTION OF APPLICATION
0 through 6	Used by the microprocessor to specify the function which it is to perform. Bits 00 through 02 are decoded to specify functionality while bits 03 through 06 are decoded to select the desired microprocessor scratch pad register.	3.2.1.3 Table 3-2 Table 3-3
7	<p>This is a dual-purpose bit.</p> <ol style="list-style-type: none"> 1. For R/W RAM access bit 7 is set to One for write or to Zero for read operations 2. Determines usage of control store bits 27 through 31. If bit 7 is One, bits 27 through 31 are used to select the inputs for the M-, I-, and CRC-multiplexers. If bit 7 is Zero, bits 27 through 31 are used to generate a number of MLCP strobe signals. 	3.5.3 3.2.1.4 Table 3-6 Table 3-7
8 through 15	These bits are used as masking bits (K-bits) by the microprocessor.	Table 3-1 3.2.1.7
16 through 26	These bits are used by the microprogram control unit as inputs to the next address generator (bits 16 through 22) and to the flag control logic (bits 23 through 26).	Table 3-9
27 through 31	These bits can be used for multiplexer input selection or for strobe pulse generation, according to the state of control store bit 7 (see above).	3.2.6 3.2.7 3.2.8 Tables 3-7, 3-10

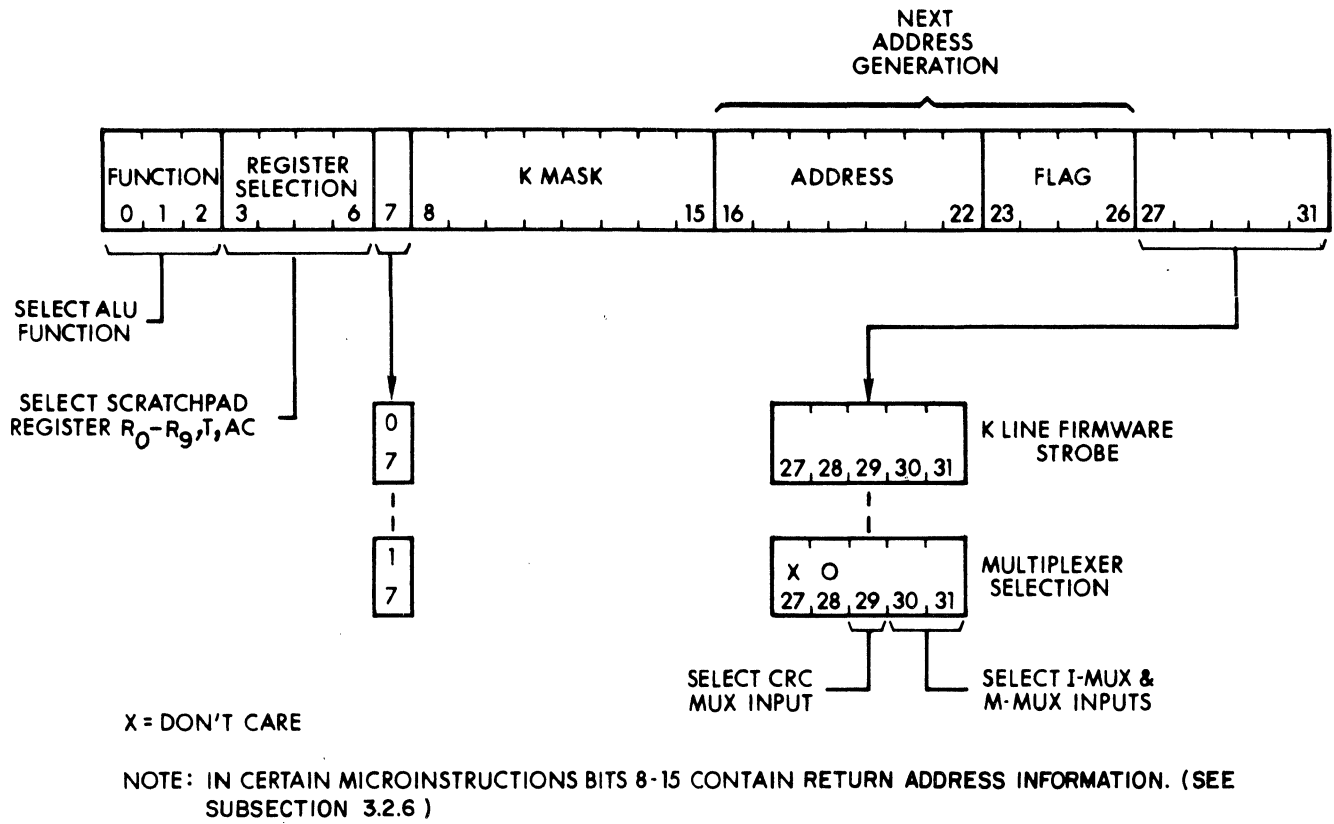


Figure 3-6 Control Store Word Format

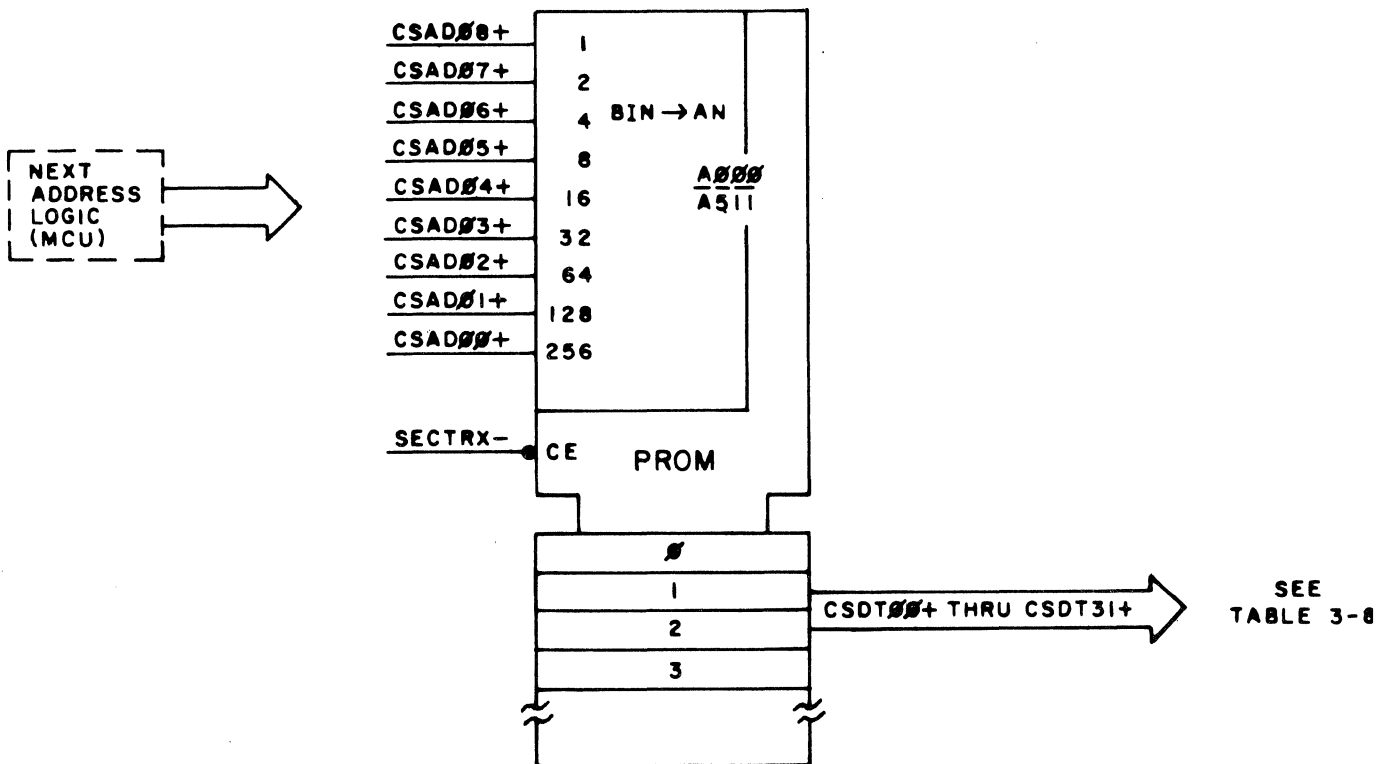


Figure 3-7 Control Store Logic

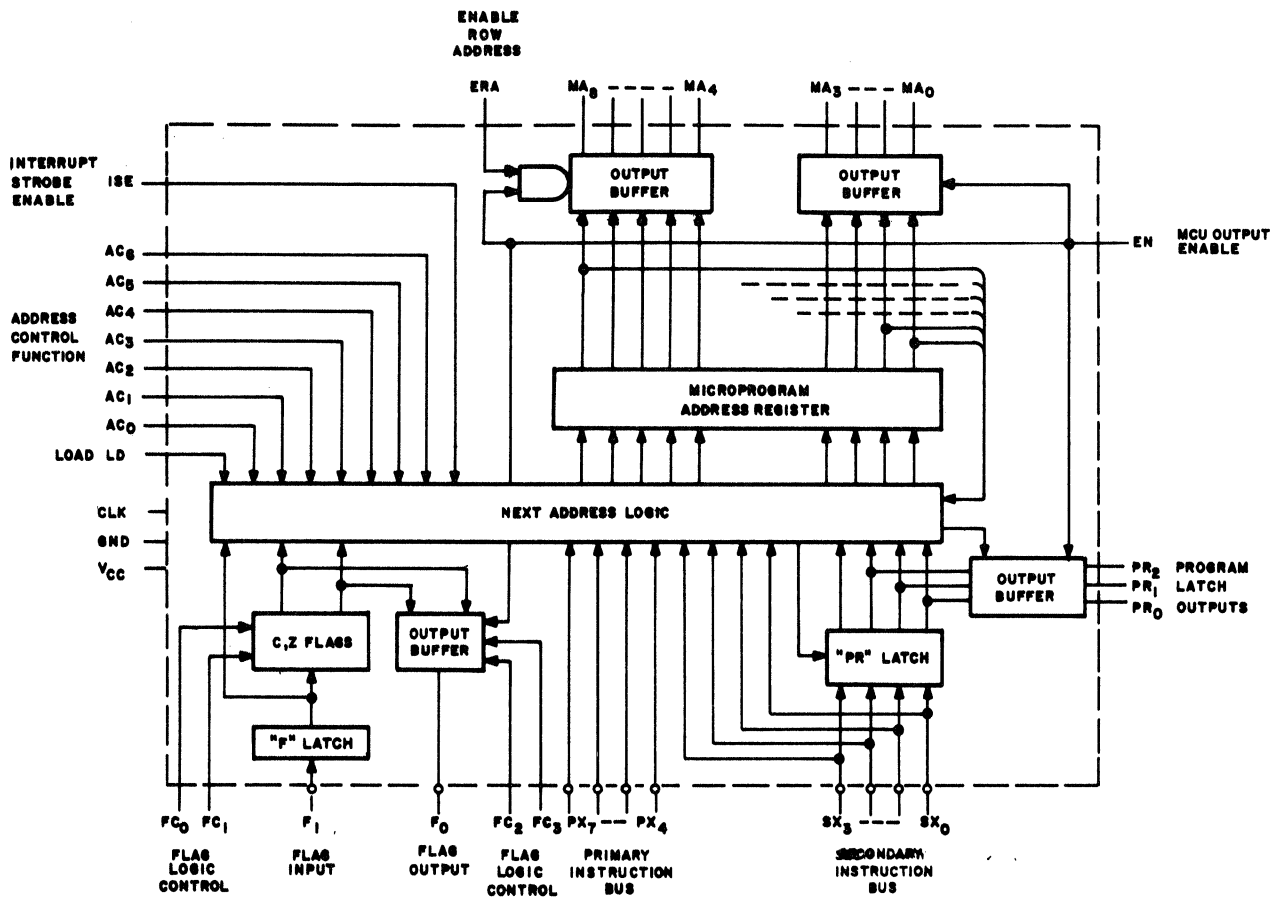


Figure 3-8 Microprogram Control Unit Functional Block Diagram

3.2.5.2 Input and Output Functions (Figures 3-8 and 3-9)

The microprogram control unit input and output functions are summarized in Table 3-9.

3.2.5.3 Next Address Logic

The next address logic of the MCU chip provides a set of conditional and unconditional address control functions. These functions are used to implement a jump or jump/test operation (refer to Table 3-10) as part of every microinstruction; that is, each microinstruction typically contains a jump operation field that specifies the address control function and, hence, the next microprogram address. The next address logic addresses a maximum of 512 microinstruction control store locations.

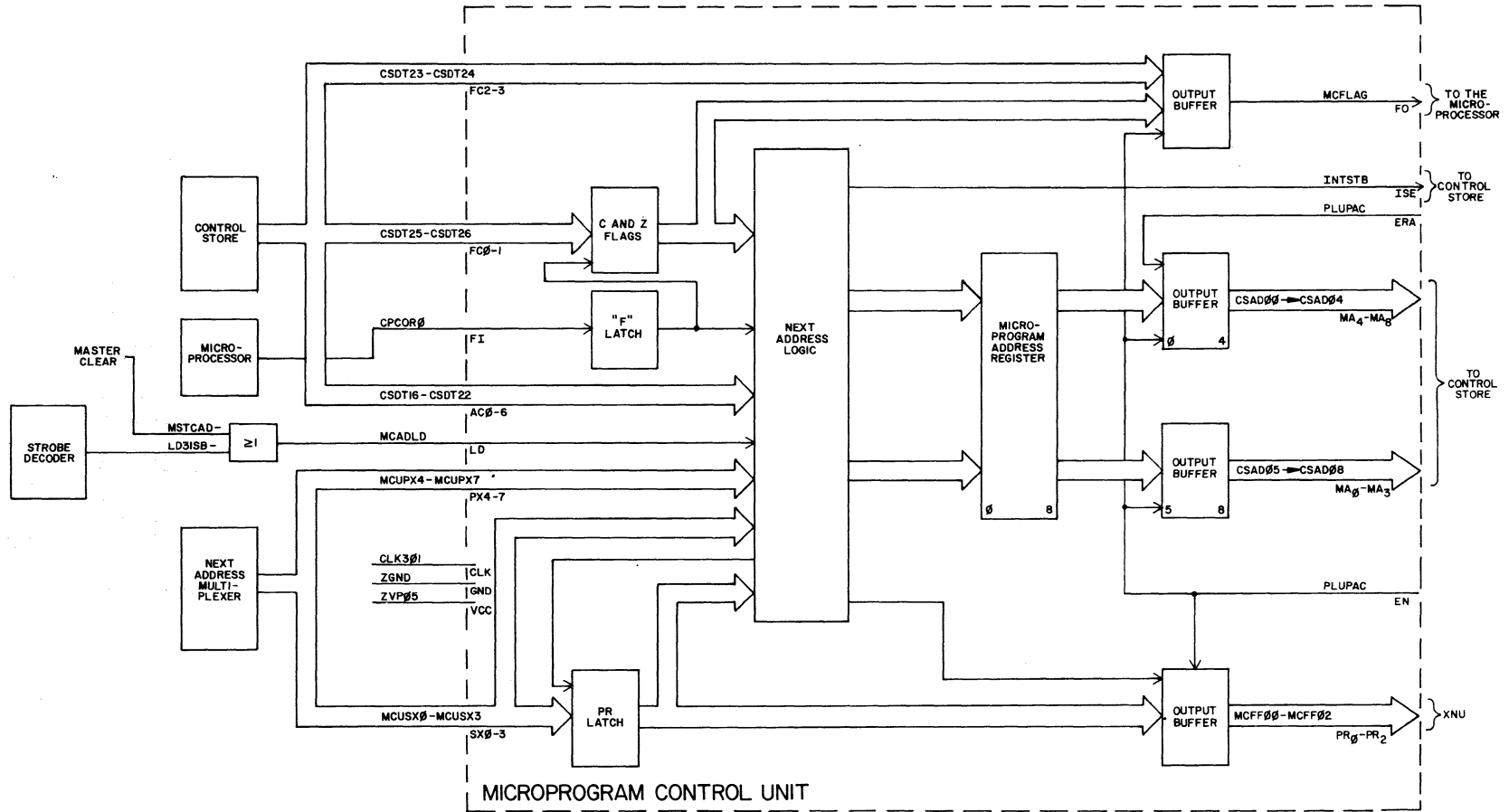


Figure 3-9 Microprogram Control Unit (MCU) Chip Functional Block Diagram

HONEYWELL PROPRIETARY AND CONFIDENTIAL

Table 3-9 MCU Input/Output Functions

PIN	SYMBOL	NAME, FUNCTION NAMES, AND DESCRIPTION	TYPE*
1-4	PX ₄ -PX ₇	Primary Instruction Bus Inputs. (MCUPX4 thru MCUPX7). Data on this bus is tested by the JPX function to determine the next control store address.	Active Low
5,6 8,10	SX ₀ -SX ₃	Secondary Instruction Bus Inputs. (MCUSX0 thru MCUSX3). Data on this bus is synchronously loaded into the PR-latch while the data on the PX-bus is being tested. During a subsequent cycle, the contents of the PR-latch may be tested by the JPR, JLL, or JRL functions to determine the next control store address.	Active Low
7,9,11	PR ₀ -PR ₂	PR-Latch Outputs.	Open Collector
12,13 15,16	FC ₀ -FC ₃	Flag Logic Control Inputs. (CSDT23 thru CSDT26). These inputs are used to cross-switch the flags (C and Z) with the flag logic input (FI) and the flag logic output (FO).	
14	FO	Flag Logic Output. (MCFLAG). The outputs of the flags (C and Z) are multiplexed internally to form the common flag logic output. MCFLAG may also be forced to a logic Zero or logic One.	Active Low Three-State
17	FI	Flag Logic Input. (CPCORO). This signal is demultiplexed internally and applied to the inputs of the flags (C and Z). (Note that the flag input data is saved in the F-latch when the clock input (CLK) is low.)	Active Low
18	ISE	Interrupt Strobe Enable Output. (INTSTB). Goes to logic One when one of the JZR functions is selected.	
19	CLK	Clock Input. (CLK301).	
20	GND	Ground.	
21-24 and 37-39	AC ₀ -AC ₆	Next Address Control Function Inputs. (CSDT16 thru CSDT22). All jump functions are selected by these control lines.	
25	EN	Enable Input. (PLUPAC). When at a logic One (high) this signal enables the microprogram address, PR-latch and flag outputs.	
26-29	MA ₀ -MA ₃	Microprogram Column Address Outputs. (CSAD08,07,06,05). Control store address.	Three-State
30-34	MA ₄ -MA ₈	Microprogram Row Address Outputs. (CSAD04,03,02,01,00) Control store address.	Three-State
35	ERA	Enable Row Address. (PLUPAC). When high, this signal independently enables the low address outputs.	
36	LD	Microprogram Address Load Input. (MCADLD) When high, this input inhibits all jump functions and synchronously loads data from the primary and secondary instruction buses (MCUSX/MCUPX) into the microprogram address register. This signal is brought high during master clear operations (to force a beginning control store address of all Zeros, where the QLT test begins), or following return from a subroutine for which a return address had been loaded into the subroutine address buffer. Operation of the PR-latch and generation of the interrupt strobe enable are not inhibited.	
40	VCC	+5 Volt Supply.	

*Active high unless otherwise specified.

HONEYWELL PROPRIETARY AND CONFIDENTIAL

Table 3-10 Address Control Functions

MNEMONIC	DESCRIPTION	FUNCTION								NEXT ROW					NEXT COL			
		AC ₆	5	4	3	2	1	0	MA ₈	7	6	5	4	MA ₃	2	1	0	
JCC	Jump in current column	0	0	d ₄	d ₃	d ₂	d ₁	d ₀	d ₄	d ₃	d ₂	d ₁	d ₀	m ₃	m ₂	m ₁	m ₀	
JZR	Jump to zero row	0	1	0	d ₃	d ₂	d ₁	d ₀	0	0	0	0	0	d ₃	d ₂	d ₁	d ₀	
JCR	Jump in current row	0	1	1	d ₃	d ₂	d ₁	d ₀	m ₈	m ₇	m ₆	m ₅	m ₄	d ₃	d ₂	d ₁	d ₀	
JCE	Jump in column/enable	1	1	1	0	d ₂	d ₁	d ₀	m ₈	m ₇	d ₂	d ₁	d ₀	m ₃	m ₂	m ₁	m ₀	
JFL	Jump/test F-latch	1	0	0	d ₃	d ₂	d ₁	d ₀	m ₈	d ₃	d ₂	d ₁	d ₀	m ₃	0	1	f	
JCF	Jump/test C-flag	1	0	1	0	d ₂	d ₁	d ₀	m ₈	m ₇	d ₂	d ₁	d ₀	m ₃	0	1	c	
JZF	Jump/test Z-flag	1	0	1	1	d ₂	d ₁	d ₀	m ₈	m ₇	d ₂	d ₁	d ₀	m ₃	0	1	z	
JPR	Jump/test PR-latches	1	1	0	0	d ₂	d ₁	d ₀	m ₈	m ₇	d ₂	d ₁	d ₀	p ₃	p ₂	p ₁	p ₀	
JLL	Jump/test left PR bits	1	1	0	1	d ₂	d ₁	d ₀	m ₈	m ₇	d ₂	d ₁	d ₀	0	1	p ₃	p ₂	
JRL	Jump/test right PR bits	1	1	1	1	1	d ₁	d ₀	m ₈	m ₇	1	d ₁	d ₀	1	1	p ₁	p ₀	
JPX	Jump/test PX-bus	1	1	1	1	0	d ₁	d ₀	m ₈	m ₇	m ₆	d ₁	d ₀	x ₇	x ₆	x ₅	x ₄	

SYMBOL	MEANING
d _n	Data on address control line n
m _n	Data in microprogram address register bit n
p _n	Data in PR-latch bit n
x _n	Data on PX-bus line n (active Low)
f,c,z	Contents of F-latch, C-flag, or Z-flag, respectively.

3.2.5.4 Flag Logic

The flag logic of the MCU provides a set of functions for saving the current value of the carry output of the microprocessor and for controlling the value of the carry input to the microprocessor (refer to Table 3-11). The flag logic is used in conjunction with the carry and shift logic of the microprocessor chips to implement a variety of shift/rotate and arithmetic functions.

3.2.5.5 Load Function

When the load function is active at the rising edge of the clock, the data on the primary and secondary instruction buses is loaded into the microprogram address register (refer to Table 3-11). The load function overrides the next address control functions but does not override the latch enable or load sub-functions of the JCE or JPX instructions. In addition, the load function does not inhibit the interrupt strobe enable or any of the flag control functions.

HONEYWELL PROPRIETARY AND CONFIDENTIAL

Table 3-11 Flag Control and Load Functions

TYPE	MNEMONIC	DESCRIPTION	FC ₁	0
Flag Input	SCZ	Set C-flag and Z-flag to f	0	0
	STZ	Set Z-flag to f	0	1
	STC	Set C-flag to f	1	0
	HCZ	Hold C-flag and Z-flag	1	1
TYPE	MNEMONIC	DESCRIPTION	FC ₃	2
Flag Output	FF0	Force FO to Zero	0	0
	FFC	Force FO to C-flag	0	1
	FFZ	Force FO to Z-flag	1	0
	FF1	Force FO to One	1	1

LOAD FUNCTION	NEXT ROW	NEXT COL
LD	MA ₈ 7 6 5 4	MA ₃ 2 1 0
0	See Table 3-10	See Table 3-10
1	0 x ₃ x ₂ x ₁ x ₀	x ₇ x ₆ x ₅ x ₄

SYMBOL	MEANING
F	Contents of the F-latch
X _n	Data on PX- or SX-bus line n (active Low)

3.2.6 Subroutine Address Buffer/Address Multiplexer (Figure 3-10)

The subroutine address buffer enables storage of the address of a subroutine. This is a firmware-controlled operation in which control store bits CSDT08 through CSDT15 (otherwise used as the K-mask field for the microprocessor) contain the return address of a firmware instruction in control store. When the firmware program jumps to a subroutine, the Save Return Address Strobe (SRADSB-) is raised to latch the return address into the buffer. Upon termination of the subroutine, firmware drives strobe signal LD31SB- low. This strobe raises the Master Clear Address Load function (MCADLD-) to select the stored address through the next address multiplexer (MCUSX0 through X3, MCUPX4 through X7) and into the next address logic of the microprogram control unit. This address is passed without modification to the Control Store Address (CSAD00 through 08) bus to select the next microinstruction word from the control store.

The next address multiplexer is also used to select the output of the M-mux as one of its inputs, thereby allowing the input to the next address logic to be selected from other sources (Megabus interface register, line adapter, etc.). The particular input selected through the M-mux is determined by the relative states of control store bits CSDT30 and CSDT31 as defined in Table 3-6.

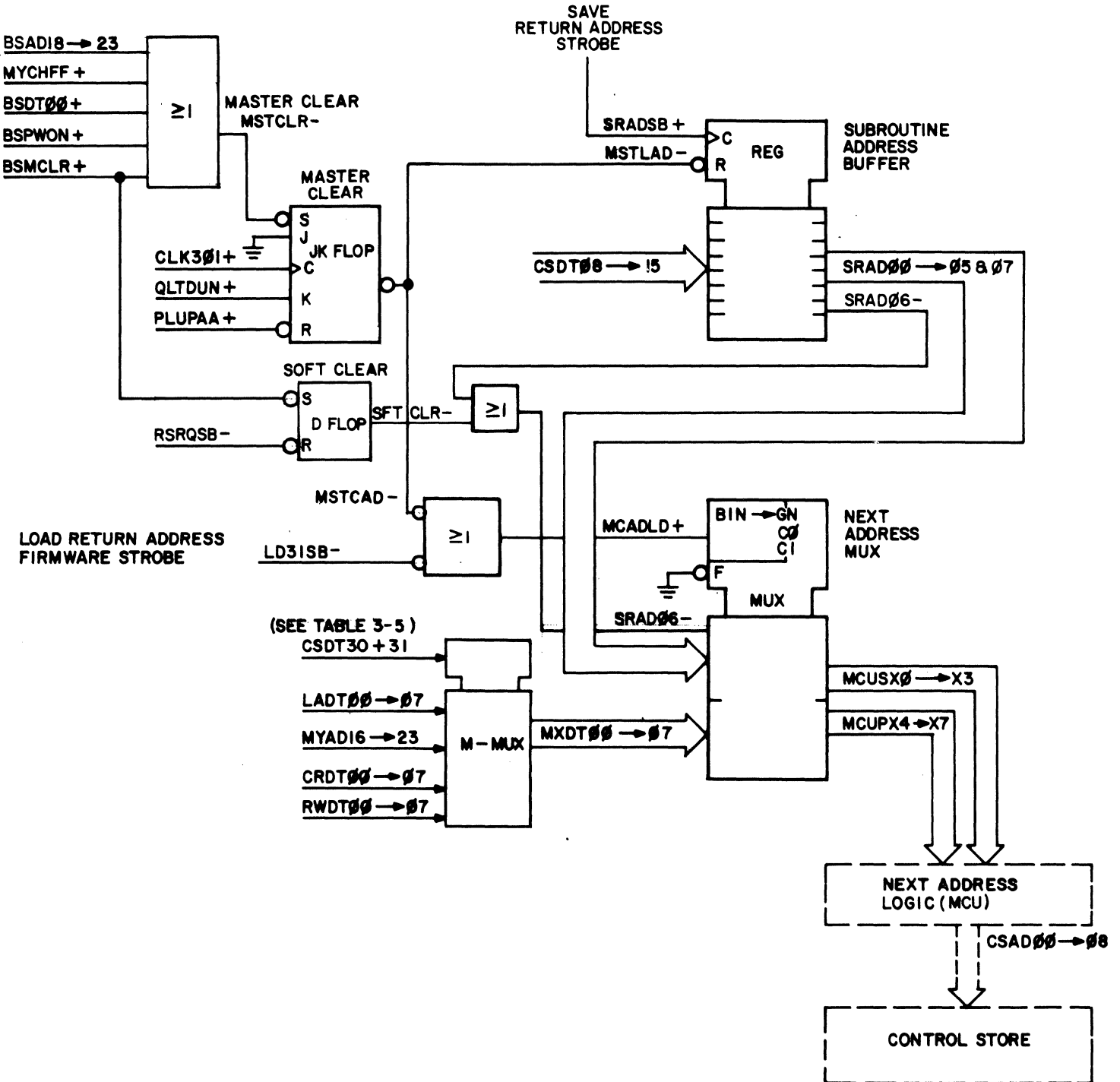


Figure 3-10 Subroutine Address Buffer/Address Multiplexer Logic

3.2.7 Control Store Buffer (Figure 3-11)

This buffer is used to latch control store data bits CSDT29 through CSDT31 for subsequent selection of inputs to the CRC mux (bit 29) and I- and M-mux's (bits 30 and 31). The I- and M-mux's are inputs to the microprocessor. Due to timing constraints, bits 30 and 31 cannot perform the multiplexer input selection in time to be effective for the current microinstruction. The same timing constraint is applicable to selection of the CRC mux input, and similarly requires latching of its selection bit 29 until a later time.

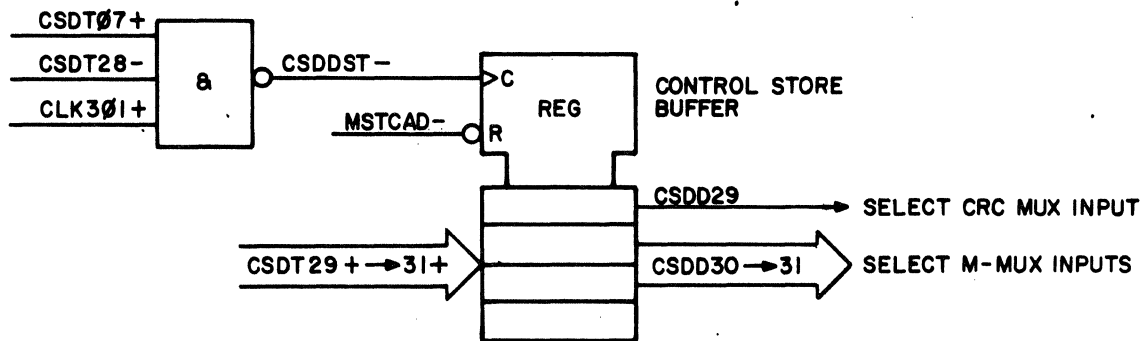


Figure 3-11 Control Store Buffer Logic

3.2.8 Strobe Generation Logic (Figure 3-12)

Control store bits CSDT27 through CSDT31 are decoded by the three strobe decoders (Figure 3-12) to generate the MLCP strobe signals defined in Table 3-12. The first two decoders generate strobe pulses, while the third decoder generates pulses that are used to select inputs for several of the MLCP's multiplexers (e.g., M-mux, I-mux, etc.), or are used as inputs to certain MLCP flip-flops.

Bits CSDT27 and CSDT28 are applied to enable gates of all three decoders, and the relative One/Zero states of these two bits determine which of the decoders is enabled. The Strobe Enable function (STBENB-) completes the enable gating for the first two decoders. This function is true (low) as long as the clock is running (CLKSYD+ and CLKSYP-) and there is no communication going on with either the line adapter (STOBLA-) or the memory (CPRWBS- and CPRWRQ-). The Step Mode pulse (SSTPEN+) is normally held in the true (high) state.

HONEYWELL PROPRIETARY AND CONFIDENTIAL

Should a test panel be attached to the MLCP, however, SSTPEN+ can be raised or lowered by activation of a test panel switch or pushbutton. When enabled, the step mode pulse advances the strobe decoders once per activation of the switch or pushbutton.

NOTE

Control store bit CSDT07 has several applications:

1. During microprocessor memory cycles CSDT07=One signifying a write.
2. When CSDT28=Zero and CSDT07=One it means load control store buffer.
3. Selects input to line adapter mux when LAADSB is used to load.

3.2.9 Master Clear and QLT Logic (Figures 3-10 and 3-13)

The MLCP can be cleared by a soft or hard initialize and can run either a partial or full Quality Logic Test (QLT). The execution of the initialize function determines which QLT is to be executed.

3.2.9.1 Soft Initialize (Figure 3-10)

When the control panel CLEAR pushbutton (CLR) is depressed, the MLCP performs a soft initialize. The function BSMCLR sets the Soft Clear (SFTCLR) flip-flop, which is ORed with bit six of the subroutine address buffer. This causes the buffer output to be selected through the address multiplexer and into the next address logic. This action forces firmware to begin at control store address location 00000010, thereby running only a partial QLT. This initialize does not clear the RAM.

3.2.9.2 Hard Initialize (Figure 3-13)

A hard initialize is executed when an output MLCP Control Command is initiated with bit 0 set to a One. This command clears the RAM to all Zeros.

Any of the conditions shown in Figure 3-13 will generate the Master Clear (MSTCLR+) function and set the Master Clear (MSTCAD) flip-flop. Master clear resets the subroutine address buffer to all Zeros and then selects the buffer through the next address mux to the next address logic, thereby forcing control store to the start address 00000000 for the beginning of the full QLT firmware routine.

The QLT flip-flop, set by master clear, is reset when the firmware issues the Quality Logic Test Done (QLTDSB-) function. This raises the K-input to the master clear flip-flop, which is then reset by the next clock pulse CLK301+.

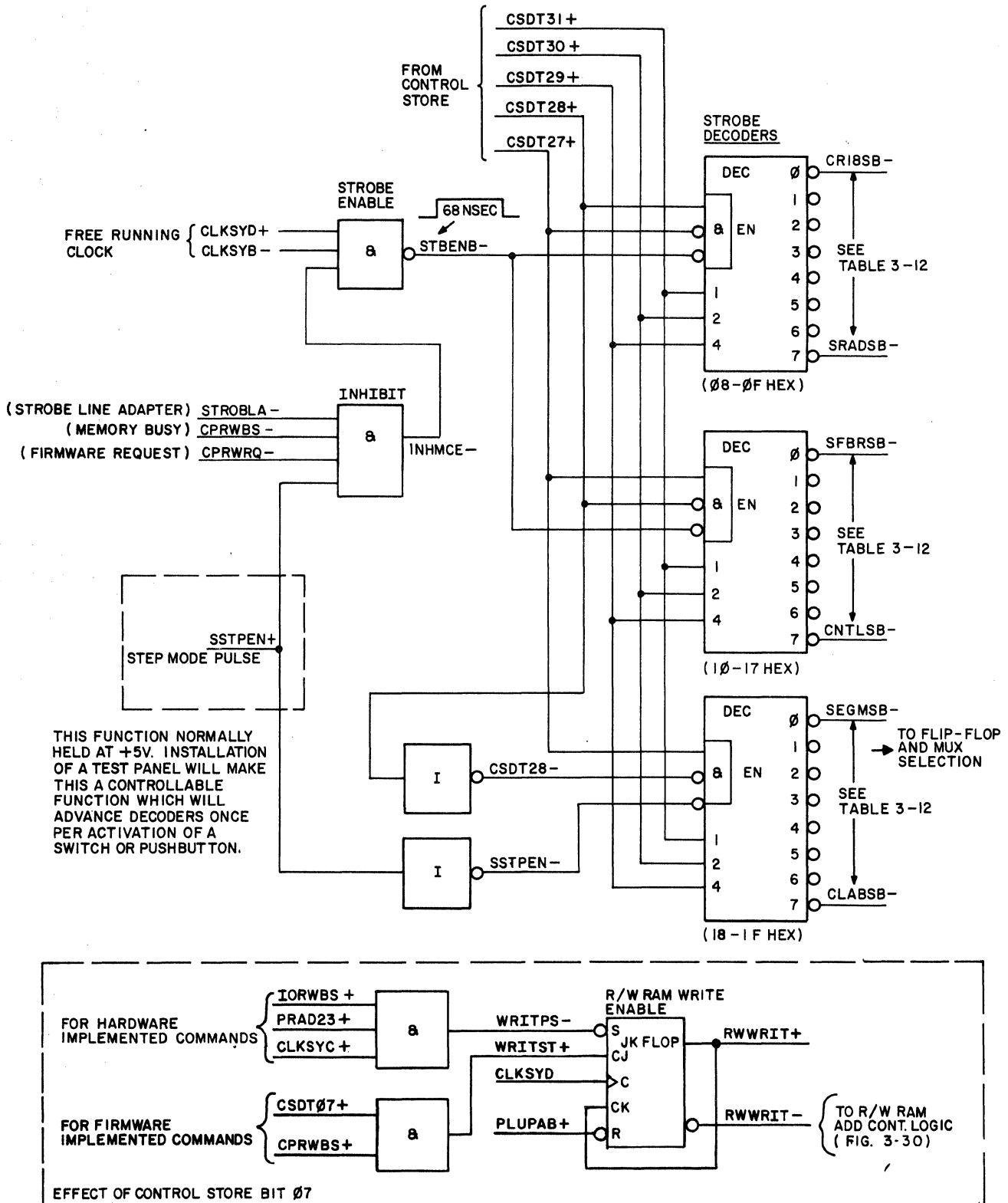


Figure 3-12 Strobe Generation Logic

HONEYWELL PROPRIETARY AND CONFIDENTIAL

Table 3-12 Strobe Functions

CONTROL STORE BIT					HEX CODE	FIRMWARE STROBE	DESCRIPTION
27	28	29	30	31			
0	1	0	0	0	08	CR18SB-	Load CRC remainder - byte 1
0	1	0	0	1	09	CR96SB-	Load CRC remainder - byte 2
0	1	0	1	0	0A	CRCSSB-	Start CRC computation
0	1	0	1	1	0B	LDCFSB-	Load configuration in CRC generator
0	1	1	0	0	0C	CRCPSB-	Load data byte for CRC
0	1	1	0	1	0D	RSIOSB-	Reset I/O special strobe
0	1	1	1	0	0E	LADDSB-	Latch CLA address (priority encoder)
0	1	1	1	1	0F	SRADSB-	Load subroutine return address
1	0	0	0	0	10	SFBRBSB-	Shift Megabus interface register 1 byte
1	0	0	0	1	11	STRQSB-	Request access Megabus interface register
1	0	0	1	0	12	RSRQSB-	Reset Megabus cycle request
1	0	0	1	1	13	CYRQSB-	Set Megabus cycle request
1	0	1	0	0	14	STBRBSB-	Load misc. Megabus interface register signals
1	0	1	0	1	15	QLTDSB-	End of QLT
1	0	1	1	0	16	SPCLSB-	Reset load/status - CCB pointers
1	0	1	1	1	17	CNTLSB-	Set CLA control lines
1	1	0	0	0	18	SEGMSB-	Select upper 512 PROM address
1	1	0	0	1	19	CPRWSB-	R/W RAM access request by microprocessor
1	1	0	1	0	1A	3002SB-	Block clock (CPE microprocessor)
1	1	0	1	1	1B	LD31SB-	Load starting address of next firmware routine
1	1	1	0	0	1C	LDPCSB-	Load ext. (high order) R/W address bits
1	1	1	0	1	1D	SWCSSB-	Select upper 1K of PROM address
1	1	1	1	0	1E	CLAASB-	CLA strobe (not data)
1	1	1	1	1	1F	CLABSBS-	CLA strobe - data (enable parity generator)

*CSDT07 must = ONE.

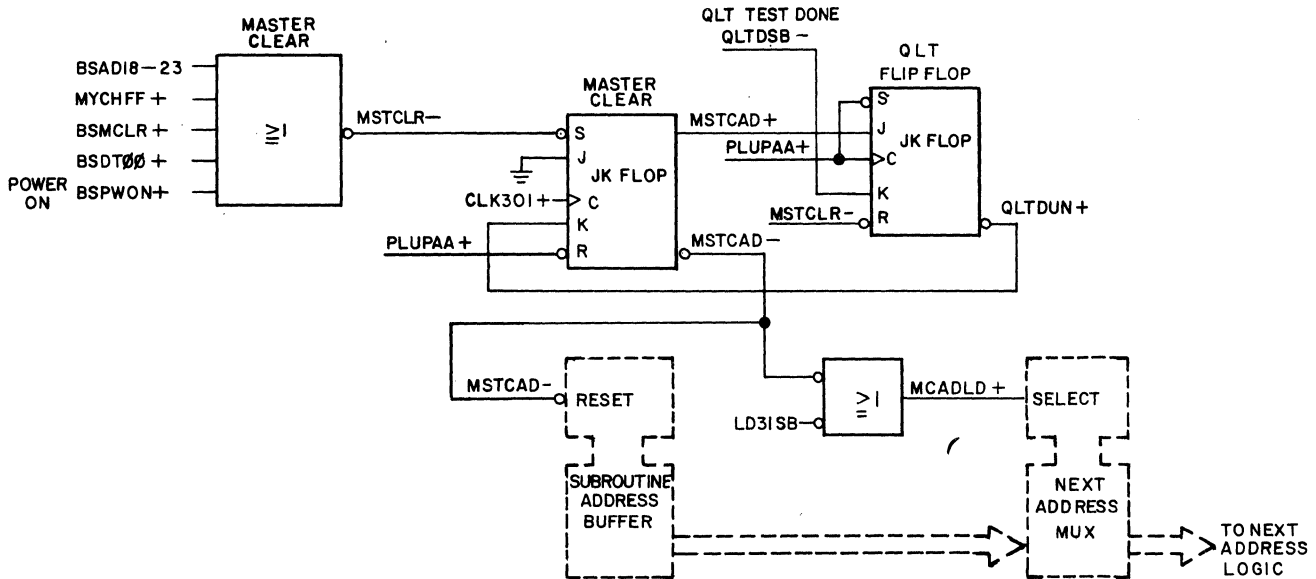


Figure 3-13 Master Clear and QLT Logic

3.3 MEGABUS CONTROL AND HARDWARE-IMPLEMENTED
I/O COMMAND LOGIC

Figure 3-14 portrays the major hardware functional areas of this logic. The basic functions of this logic are:

1. Compare the address on the Megabus with the state of the MLCP channel address switches to ascertain whether or not the MLCP is being addressed.
2. Monitor the MLCP to ascertain whether or not it is busy.
3. Determine whether the MLCP is to be acting as master or slave.
4. Generate ACK, NAK, or WAIT signal according the MLCP condition.
5. Analyze the function code on the Megabus address lines to ascertain the function to be performed, and whether the command is to be implemented by MLCP hardware or firmware.
6. Analyze the line number on the Megabus address lines and manipulate, via scratch pad memories and counters, the addresses of CCBs located in the R/W RAM.
7. Transfer address, range, flag and control information from the Megabus address and data lines into the appropriate CCB area in R/W RAM.
8. Count the number of CCBs per line and store a NAK indication in scratch pad memory so that the MLCP may NAK any attempt to load more than the maximum number of CCBs.

As shown in Figure 3-14, the logic which accomplishes the functions listed above is divided into the following functional areas:

1. Megabus Control Logic
 - a. Master Cycle Logic
 - b. Slave Response Logic
2. CCB Address Control Logic.

Activity of the MLCP logic shown in Figure 3-14 is defined by Tables 3-13, 3-16, and 3-17. The description in this subsection is based primarily upon typical examples of command execution, and makes frequent reference to these tables. Hardware areas are described as they are encountered during command execution. These examples provide sufficient understanding of hardware functionality to enable extrapolation of details for the implementation of commands not described herein.

3.3.1 Firmware and Hardware Implementation
of I/O Commands

The MLCP executes certain commands by resident firmware (see Section IV for details) and other instructions by hardware activity. Regardless of the command type (i.e., input or output, hardware- or firmware-implemented), the initial steps of addressing the MLCP and

HONEYWELL PROPRIETARY AND CONFIDENTIAL

loading the Megabus Interface Register (MIR) are the same as shown in Figure 3-15. The method of implementation is determined by bit four of the output function code translator (PRDTB4). If this bit is a logic One, the command is to be implemented by firmware; if it is a logic Zero, MLCP hardware handles the implementation under control of the function code translator.

Megabus address bits BSAD18 through BSAD23 (the function code) supply part of the address for the function code translator and are loaded into the function code latch. The remainder of the translator address is supplied by the function code counter which, because it is periodically implemented, allows different function code outputs to be generated as required to perform different functions during execution. It is the output of the translator which controls R/W RAM addressing for loading of CCBs, requests R/W memory cycles, shifts the MIR, etc.

The same address bits (BSAD18 through BSAD23) which address the translator are also loaded into the MIR, together with all other information on the Megabus address and data lines. If the command is to be implemented by firmware, the firmware program must extract this function code from the MIR and analyze it to determine what action to take. As stated above, the function code for all commands requiring firmware implementation generates a function code translator output having bit four (PRDTB4) set to a logic One. This bit is sent to the I-mux, where it is examined by firmware to determine whether the command is to be implemented by hardware or firmware.

Table 3-13 identifies Megabus address and data line usage and gives the function code bit structure for all input and output instructions.

3.3.1.1 Firmware-Implemented Commands (See Table 3-14)

This subsection typifies implementation of commands by firmware via discussion of MLCP action in response of an Output Interrupt Control (FC - 03) command. First, the channel address (BSAD08 through BSAD13) is compared against the setting of the channel address switches by the channel address comparator. An equal comparison sets the My Channel Flip-Flop (MYCHFF). If the MLCP is busy, MYCHFF causes a Wait response to be sent to the Megabus via the ACK/NAK/Wait Response logic. If the MLCP is not busy, MYCHFF causes an acknowledgement (MYACKR) to be sent to the Megabus. The content of the Megabus address and data lines is then stored in the MIR.

At the same time, address bits BSAD18 through BSAD23 (the function code) are stored in the function code latch (PRAD18 through PRAD23). These bits form part of the address for the function code translator. In this example the function code (03) generates an address of (decimal) 24 (refer to Table 3-15). The output of this location has bit four (PRDTB4) set to a logic One, which is sent to the I-mux. When firmware, which periodically tests the I-mux, detects this bit, it examines the content of the MIR (which contains the function code), determines the nature of the command to be executed, and executes it accordingly.

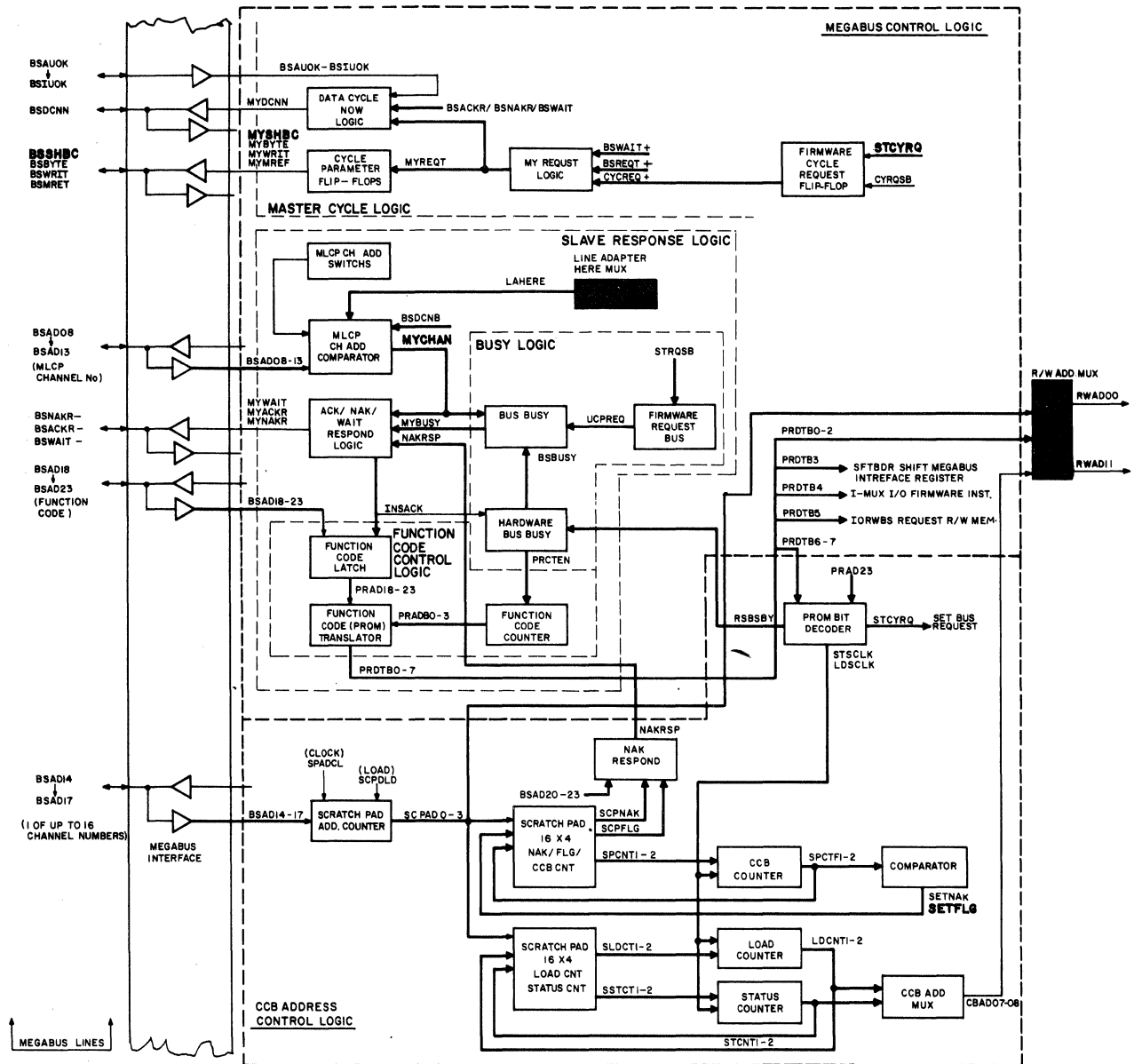


Figure 3-14 Megabus Control and Hardware-Implementation of I/O Instructions

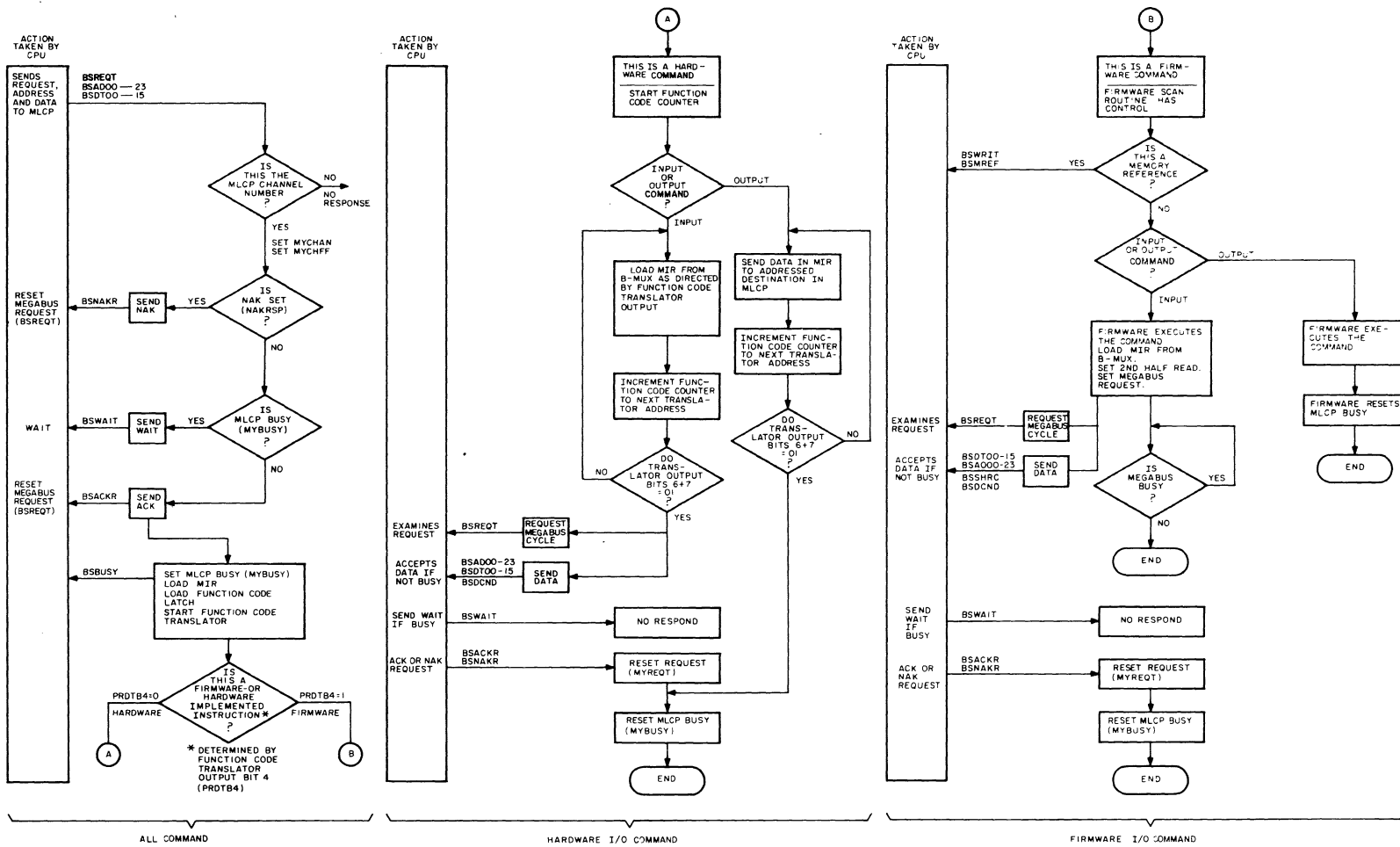


Figure 3-15 Hardware and Firmware Implementations

Table 3-13 Command Function Codes

MEGABUS ADDRESS BITS BSAD00-BSAD23																								FUNCTION CODE (HEX)						
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23							
NORMALLY = 0								MLCP CHANNEL NO.				CHANNEL NUMBER				FUNCTION CODE														
IOLD								(Output Add)								0	0	1	0	0	1									09
								(Output Range)								0	0	1	1	0	1									0D
Output CCB Control																0	0	1	1	1	1									0F
Output Channel Control (FW)																0	0	0	1	0	1									05
Output LCT Byte (FW)																0	0	1	0	1	1									0B
Output Interrupt Control (FW)																0	0	0	0	1	1									03
Output MLCP Control (Master Clear) (FW)																0	0	0	0	0	1									01
Input Range																0	0	1	1	0	0									0C
Input Status																0	1	1	0	0	0									18
Input Status (Next)																0	1	1	0	1	0									1A
Input Device Identification																0	0	0	1	1	0									26
Input LCT Byte (FW)																0	1	1	1	1	0									1E
Input Data Set Status (FW)																0	1	1	1	0	0									1C
Input Extended Device ID Number (FW)																0	0	1	0	0	0									08

FW = Firmware Implemented Command

Table 3-14
Firmware-Implemented Commands

FUNCTION CODE	INSTRUCTION
05	Output Channel Control
03	Output Interrupt Control
0B	Output LCT Byte
1C	Input Data Set Status
1E	Input LCT Byte
01	Output MLCP Control
08	Input Extended Device ID Number

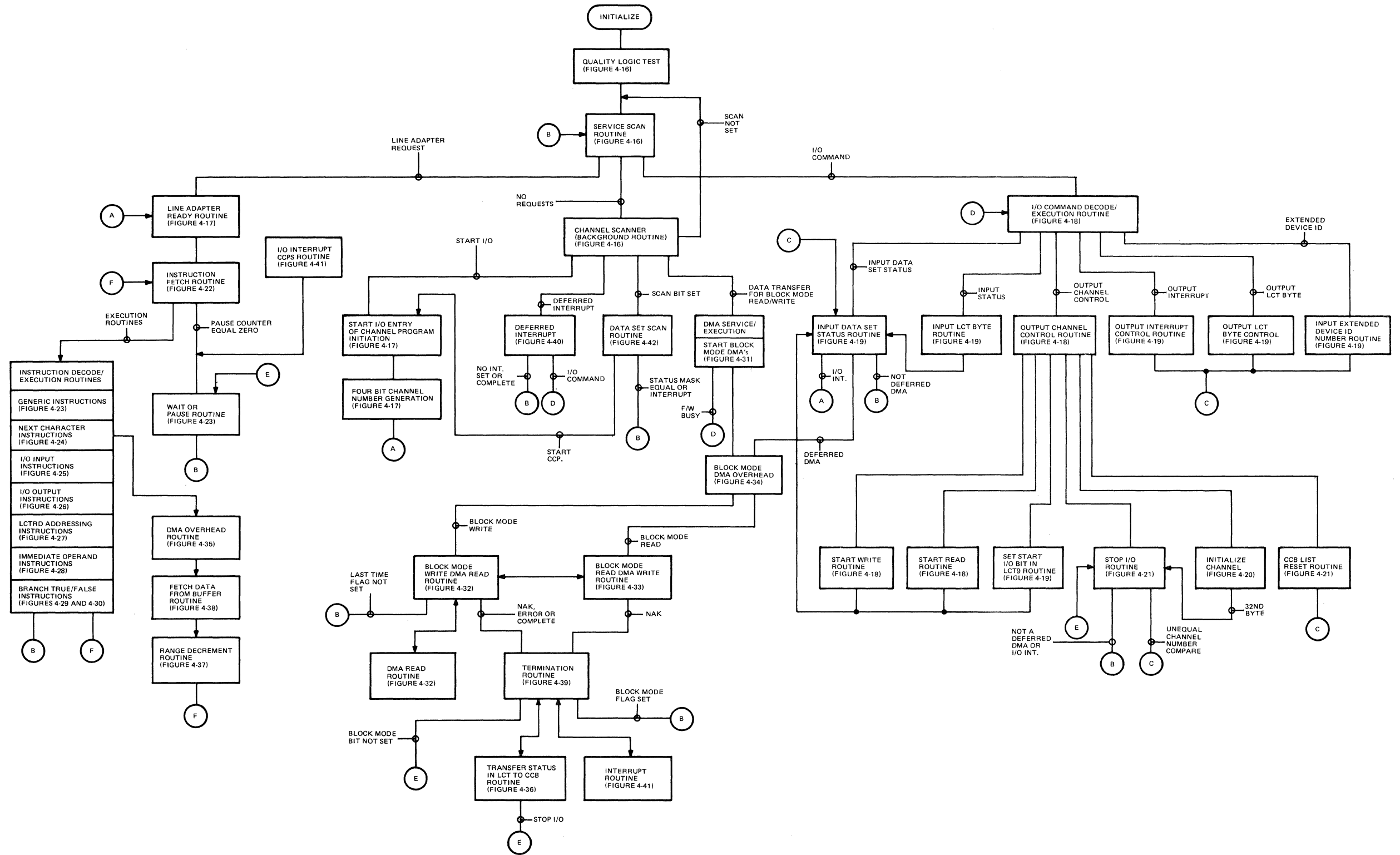


Figure 4-15 MLCP Firmware Overview Flow Chart

HONEYWELL PROPRIETARY AND CONFIDENTIAL

Table 3-15 Function Code Translation for Firmware-Implemented Commands

FUNCTION CODE TRANSLATOR (PROM) DECIMAL ADDRESS	FUNCTION CODE TRANSLATOR OUTPUT PRDTB0-7 (PROM)							FUNCTION CODE TRANSLATOR ADDRESS PRAD19-23, PRADB2-0 (PROM)								FUNCTION CODE HEX	FIRMWARE COMMAND	
	7	6	5	4	3	2	1	0	129	64	32	16	8	4	2			1
08	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	01	MLCP Control
24	0	0	0	1	0	1	1	0	0	0	0	1	1	0	0	0	03	Output Interrupt Control
40	0	0	0	1	0	1	1	0	0	0	1	0	1	0	0	0	05	Output Channel Control
64	0	0	0	1	0	1	1	0	0	1	0	0	0	0	0	0	08	Input Extended Device ID Number
88	0	0	0	1	0	1	1	0	0	1	0	1	1	0	0	0	0B	Output LCT Byte
224	0	0	0	1	0	1	0	0	1	1	1	0	0	0	0	0	1C	Input Data Set Status
240	0	0	0	1	0	0	0	0	1	1	1	1	0	0	0	0	1E	Input LCT Byte

Firmware Bit → bit 7
 B-Mux Select → bit 6
 From Bus Address Bits 19-23 → bits 129-16
 From Address Counter Bits 0-2 → bits 8-1

3.3.1.2 Hardware-Implemented Commands
 (See Table 3-16)

This subsection typifies implementation of commands by hardware via discussion of MLCP action in response to an Input ID (FC - 26) command. The initial MLCP action is as described in subsection 3.3.1.1, with the channel address comparator checking for the correct address and an acknowledge (MYACKR) being sent to the Megabus (assumes that MLCP is not busy). The MIR is loaded from the Megabus address and data lines, and the function code is loaded into the function code latch.

Table 3-17 shows that function code (FC - 26) generates a function code translator address of (decimal) 48. In this location the outputs have bits one and two (PRDTB1,2) set to Zero and One, respectively. These two bits select the B-mux input to the MIR via BXSELA and BXSELB (refer to Table 3-7). Bit three is set to One, and is used to shift the MIR. The selected B-mux input data is loaded into the MIR.

The function code counter increments to translator address (decimal) 49. Since bit three (PRDTB3) is a One, the MIR is shifted once again. Bits one and two (PRDTB1,2) are set to One and Zero, respectively, and are used to select the next B-mux input, which is then loaded into the MIR.

HONEYWELL PROPRIETARY AND CONFIDENTIAL

The function code counter is incremented again to translator address (decimal) 50. In this location bit seven (PRDTB7) is set to a One, which causes the MLCP to generate a request for a Megabus cycle (i.e., the identification data requested by the current instruction is now in the MIR and ready for transfer via the Megabus). When the Megabus responds with an acknowledge, the MLCP busy logic is reset.

Table 3-16 Hardware-Implemented Commands

FUNCTION CODE	INSTRUCTION
09	IOLD (Output Address)
0D	IOLD (Output Range)
0F	Output CCB Control
0C	Input Range
18	Input Status
1A	Input Next Status
26	Input Device Identification

Table 3-17 Function Code Translation for Hardware-Implemented Commands (Sheet 1 of 2)

FUNCTION CODE TRANSLATOR (PROM) DECIMAL ADDRESS	FUNCTION CODE TRANSLATOR (PROM) OUTPUT PRDTB0-7							FUNCTION CODE TRANSLATOR (PROM) ADDRESS							FUNCTION CODE (HEX)	COMMAND (Hardware Action)		
	7	6	5	4	3	2	1	0	128	64	32	16	8	4			2	1
																	09	IOLD (Output Address)
72	0	1	0	0	0	0	0	0	0	1	0	0	1	0	0	0		Increment CCB Counter Address
73	0	0	0	0	1	0	0	0	0	1	0	0	1	0	0	1		Shift MIR
74	0	0	0	0	1	0	0	0	0	1	0	0	1	0	1	0		Shift MIR
75	0	0	1	0	1	0	0	0	0	1	0	0	1	0	1	1		Shift MIR; Address R/W Memory; Store Data Bits 8-15
76	0	0	1	0	1	1	0	0	0	1	0	0	1	1	0	0		Shift MIR; Address R/W Memory; Store Data Bits 0-7
77	0	0	1	0	1	0	1	0	0	1	0	0	1	1	0	1		Shift MIR; Address R/W Memory; Store Address Bits 0-7
78	1	0	0	0	0	0	0	0	0	1	0	0	1	1	1	0		Reset Bus Busy
																	0D	IOLD (Output Range)
104	0	1	0	0	0	0	0	0	0	1	1	0	1	0	0	0		Increment CCB Counter Address
105	0	0	0	0	1	0	0	0	0	1	1	0	1	0	0	1		Shift MIR
106	0	0	0	0	1	0	0	0	0	1	1	0	1	0	1	0		Shift MIR
107	0	0	1	0	1	1	1	0	0	1	1	0	1	0	1	1		Shift MIR; Address R/W Memory; Store Data Bits 8-15
108	0	0	1	0	1	0	0	1	0	1	1	0	1	1	0	0		Shift MIR; Address R/W Memory; Store Data Bits 0-7
109	1	0	0	0	0	0	0	0	0	1	1	0	1	1	0	1		Reset Bus Busy
																	18	Input Status
192	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0		
193	0	0	1	0	1	0	1	1	0	1	1	0	0	0	0	1		Shift MIR; Address R/W Memory
194	0	0	1	0	1	1	1	1	0	1	1	0	0	0	1	0		Shift MIR; Address R/W Memory
195	1	0	0	0	0	0	0	0	0	1	1	0	0	0	1	1		Set Request

FROM BUS
ADDRESS BITS
19-23

FROM
ADDRESS
COUNTER
BITS
0-2

Table 3-17 Function Code Translation for Hardware-Implemented Commands (Sheet 2 of 2)

FUNCTION CODE TRANSLATOR (PROM) DECIMAL ADDRESS	FUNCTION CODE TRANSLATOR (PROM) OUTPUT PRDTB0-7								FUNCTION CODE TRANSLATOR (PROM) ADDRESS								FUNCTION CODE (HEX)	COMMAND (Hardware Action)
	7	6	5	4	3	2	1	0	128	64	32	16	8	4	2	1		
208	0	1	0	0	0	0	0	0	1	1	0	1	0	0	0	0	1A	Input Next Status
209	0	0	1	0	1	0	1	1	1	1	0	1	0	0	0	1		Increment Status CCB
210	0	0	1	0	1	1	1	1	1	1	0	1	0	0	1	0		Shift MIR; Address R/W Memory;
211	1	1	0	0	0	0	0	0	1	1	0	1	0	0	1	1		Read Status
212	1	0	0	0	0	0	0	0	1	1	0	1	0	1	0	0		Shift MIR; Address R/W Memory;
																		Read Status
																		Write PROM (Update)
																		Set Request
48	0	0	0	0	1	1	0	0	0	0	1	1	0	0	0	0	26	Input ID
49	0	0	0	0	1	0	1	0	0	0	1	1	0	0	0	1		Shift MIR; Select B-Mux
50	1	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0		Shift MIR; Select B-Mux
																		Reset Bus Busy
96	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0C	Input Range
97	0	0	1	0	1	1	1	0	0	1	1	0	0	0	0	1		Shift MIR; Address R/W Memory;
98	0	0	1	0	1	0	0	1	0	1	1	0	0	0	1	0		Read Range
99	1	0	0	0	0	0	0	0	0	1	1	0	0	0	1	1		Shift MIR; Address R/W Memory;
																		Read Range
																		Reset Bus Busy
120	0	1	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0F	Output CCB Control
121	0	0	0	0	1	0	0	0	0	1	1	1	1	0	0	1		Increment CCB LDAD Counter
122	0	0	0	0	1	0	0	0	0	1	1	1	1	0	1	0		Shift MIR
123	0	0	1	0	0	1	0	1	0	1	1	1	1	0	1	1		Shift MIR
124	0	0	1	0	1	1	1	1	0	1	1	1	1	1	0	0		Store Data 8-15; Address R/W Memory
125	0	0	1	0	0	0	1	1	0	1	1	1	1	1	0	1		Shift MIR; Address R/W Memory;
126	1	1	0	0	0	0	0	0	0	1	1	1	1	1	1	0		Store Data 8-15
127	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1		Store Data 0-7
																		Write PROM (Update)
																	Reset Bus Busy	

FROM BUS ADDRESS BITS 19-23
FROM ADDRESS COUNTER BITS 0-2

3.3.2 Megabus Control Logic

As shown in Figure 3-14, the Megabus control logic is composed of a number of smaller, interrelated functional logic areas. For ease of discussion the control logic is divided into:

1. Master Cycle Logic
2. Slave Response Logic
3. Function Code Control Logic.

3.3.2.1 Master Cycle Logic (Figure 3-16)

When the MLCP wants to transfer over the Megabus to main memory or to the CPU, it must set the Megabus Request flip-flop UCPREQ for firmware commands; BSBUSY for hardware commands (see Figure 3-18) and load the MIR with address and data. If the command is to be executed by firmware, the MLCP sets the appropriate Megabus function (i.e., BSBYTE, BSMREF, or BSWRIT) for the transfer.

Once the MIR is loaded and the control functions have been set, the Cycle Request flip-flop (CYCREQ) is set, thereby initiating the Megabus request. When the cycle request flip-flop is set, and when the Megabus is not busy (BUSBSY), the Set Request signal becomes active, causing the request flip-flop (MYREQT+) to set. The request flip-flop goes to the Megabus (BSREQT) driver and also to the Set Data Cycle Now (SETDCN-) gate. This Megabus request inhibits other units on the Megabus from initiating a new request, and causes the MLCP priority network (BSMYOK) output to go low.

With the Megabus Request (MYREQT) flip-flop set and with the completion of any previous Megabus (BSDCNB) cycle, the MLCP will set its Data Cycle Now (MYDCNN) flip-flop provided it has the highest priority on the Megabus. Priority is determined by the lines BSAUOK through BSIUOK being high, indicating that no unit on the Megabus with higher priority has a request active. When all conditions of the SETDCN gate are met, the flip-flop (MYDCNN+) sets until the slave unit responds. The response of the slave clears the MYDCNN flip-flop, resetting the request unless the response is a Wait, in which case the request flip-flop remains set.

For I/O input commands, a Second Half Bus Cycle (MYSHBC) is developed in the MLCP for a response cycle. The response may be either hardware (HDSHBC) or firmware (FWSHBC) generated dependent on the type of input command.

3.3.2.2 Slave Response Logic (Figures 3-17 through 3-19)

This logic generates the ACK/NAK/Wait signal as required (Figure 3-17), analyzes the address on the Megabus address lines to determine if the MLCP is being addressed (channel address comparator), generates the busy signal (MYBUSY, Figure 3-18) when required, and translates the function code on the Megabus address lines (BSAD18 through BSAD23) into meaningful bit configurations for MLCP control (via the function code control logic shown in Figure 3-19). The MLCP may respond to a request with any one of the following responses (Figure 3-17):

1. Wait: This response is generated whenever the MLCP is busy with a hardware, firmware, or DMA instruction (except Initialize).
2. NAK: This response is made to all commands when the MLCP is performing the QLT. This response is also made to IOLD (Output Address and Output Range) and Output CCB Control commands when the CCB list for that channel is full. NAK's are stored in scratch pad memory for each line (0-15). This response is made to an input next status as an indication that the CCB pointers cannot advance without causing an error.
3. ACK: This response is made to all commands that are neither NAK'd nor Wait'ed.
4. No Response: MLCP may make no response when the addressed channel has no communications line adapter installed.

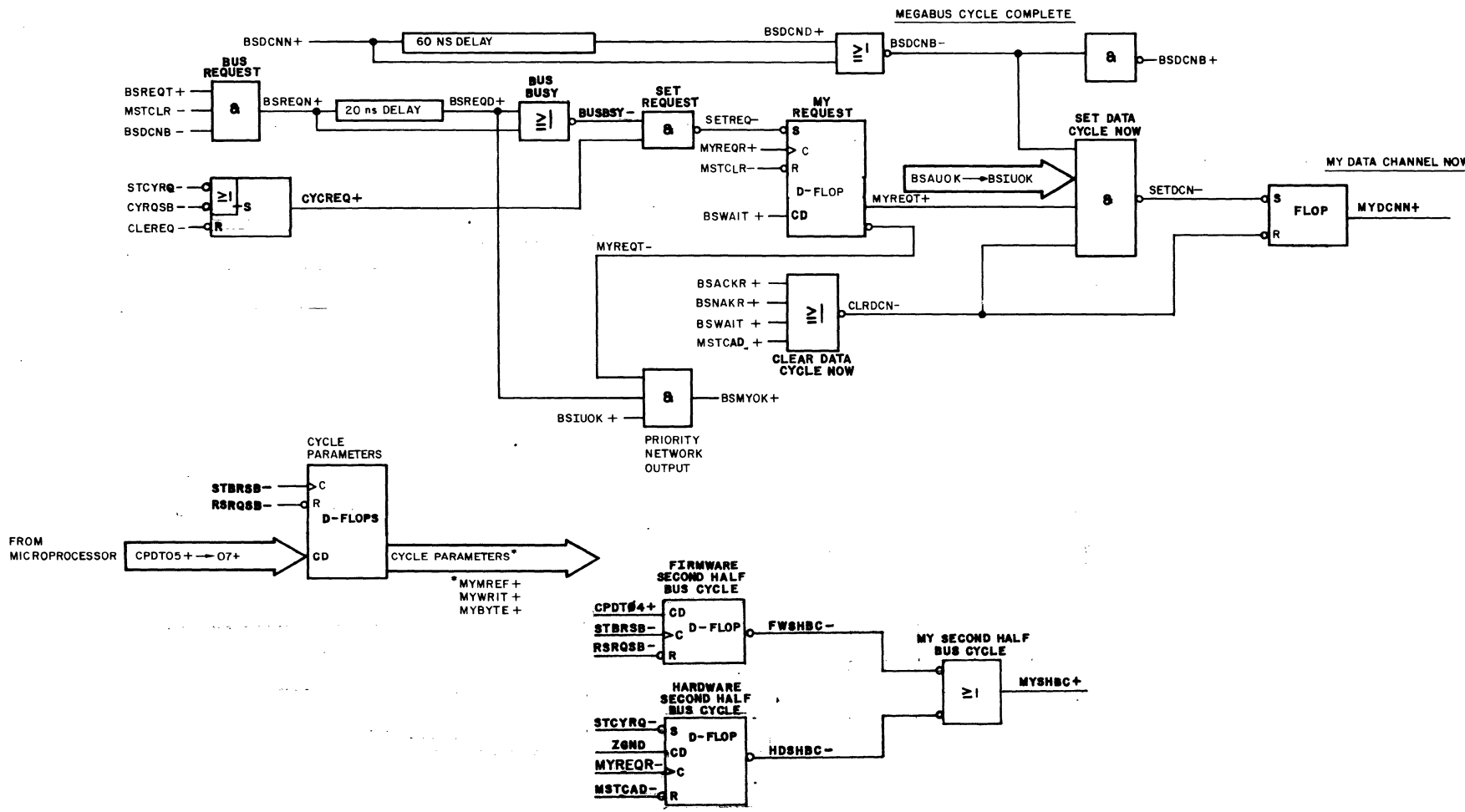


Figure 3-16 Master Cycle Logic

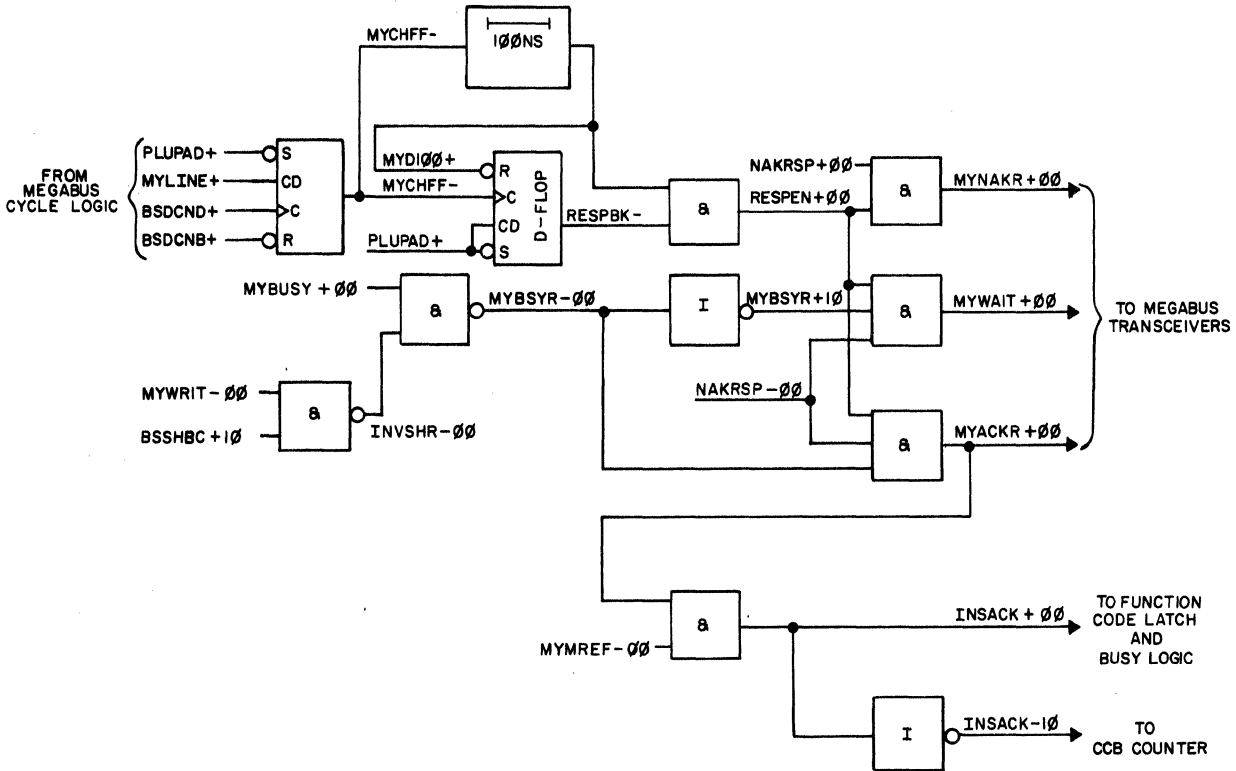


Figure 3-17 ACK/NAK/Wait/Respond Logic

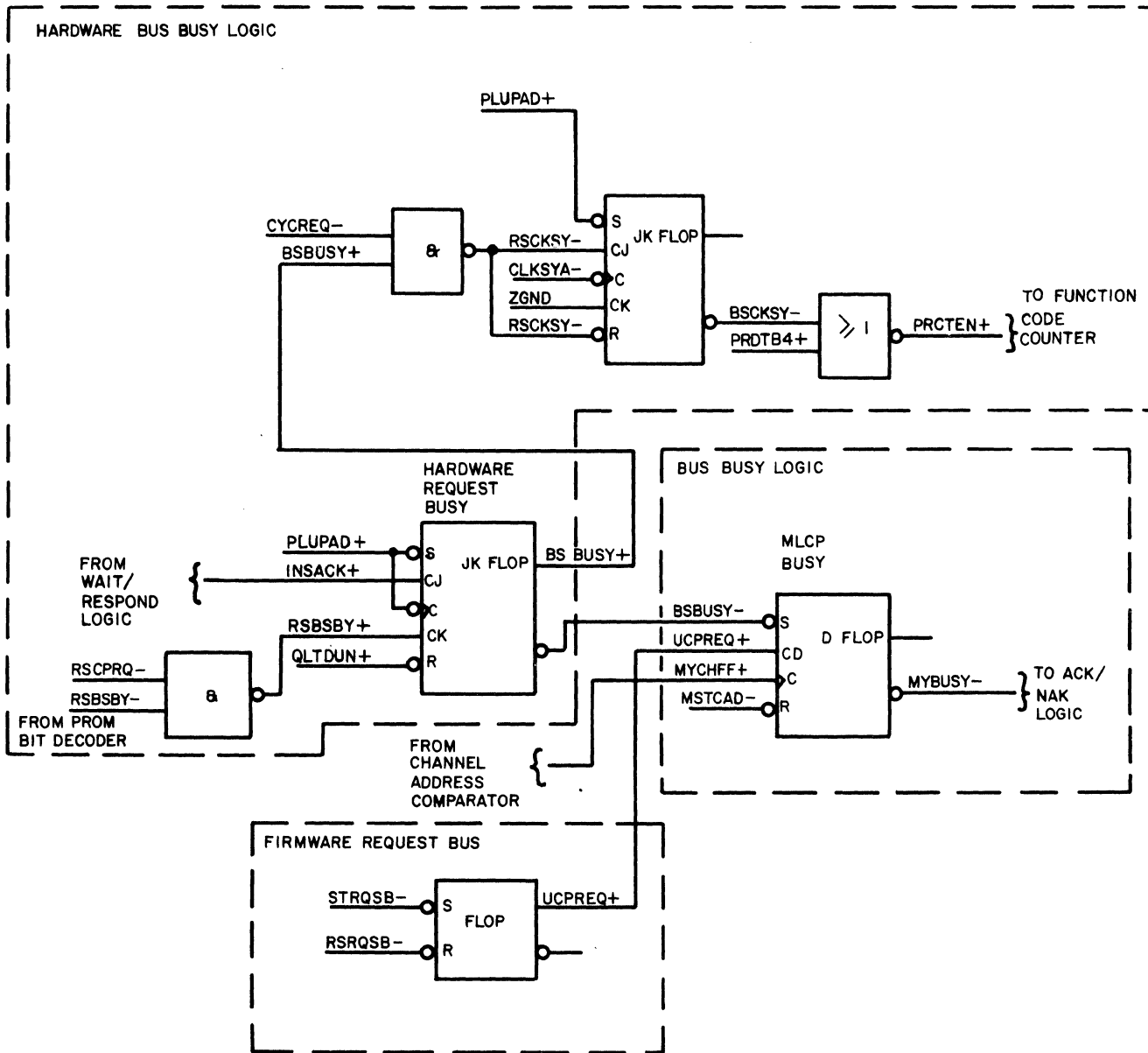


Figure 3-18 Busy Logic

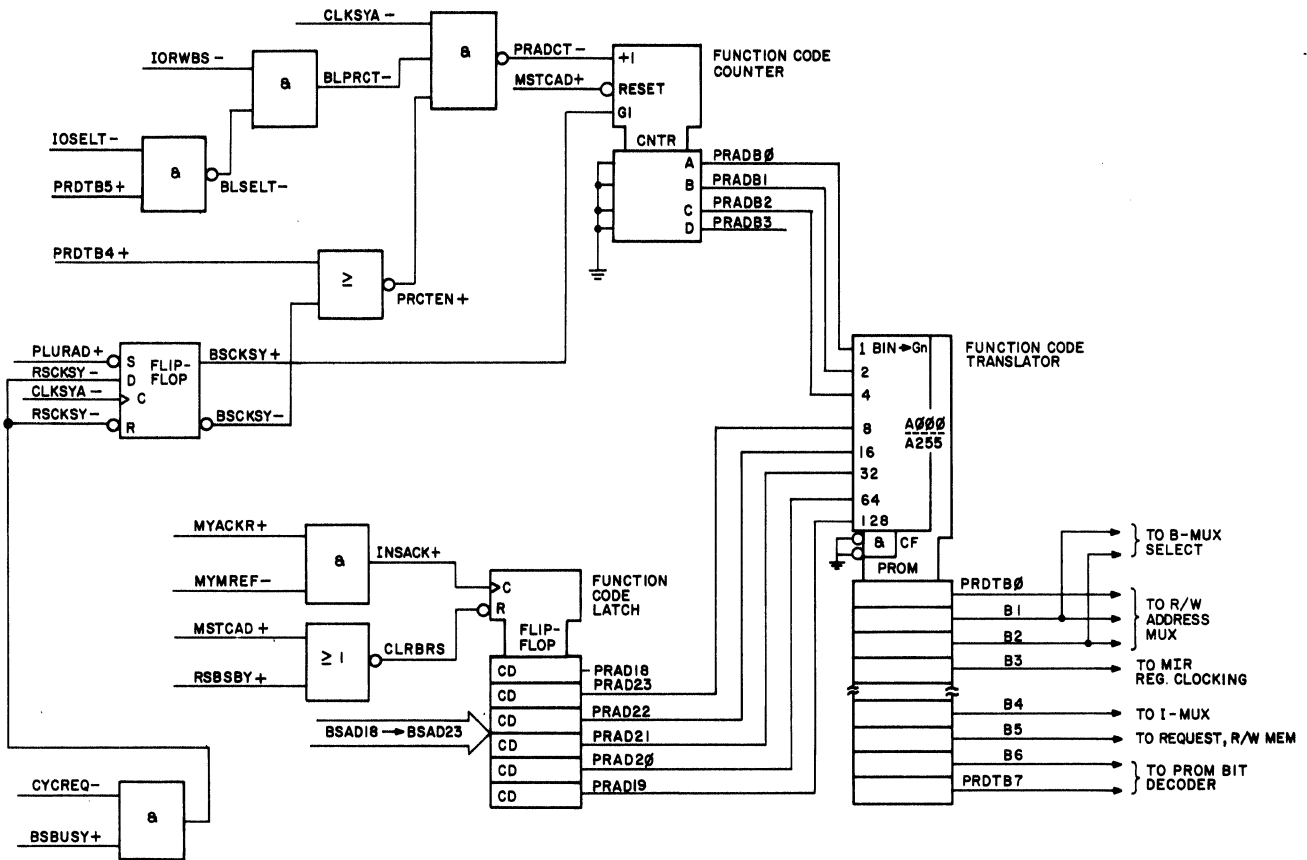


Figure 3-19 Function Code Control Logic

3.3.2.3 Function Code Control Logic

The function code translator output controls MLCP operation as follows (Figure 3-19):

- Bits 0-2: Supply part of R/W address
- Bits 1&2: Selection of B-mux inputs
- Bit 3: Shifts Megabus interface register (MIR)
- Bit 4: Goes to I-mux for firmware examination (identify command as firmware or hardware)
- Bit 5: Generates R/W memory cycle
- Bits 6&7: Controls operation of CCB address control counters and resets busy when bit seven is a One.

Tables 3-16 and 3-17 are maps of the function code translator (PROM) for hardware implemented commands. These tables show the bit format for each PROM address, and specify the MLCP hardware activity associated with each address.

3.3.3 CCB Address Control Logic

This subsection describes how a CCB is handled by the MLCP logic. As shown in Figure 3-20, functional logic areas involved in CCB handling are:

1. A scratch pad address counter
2. Two scratch pad memories (16 locations with four outputs each)
3. The CCB counter
4. The load counter
5. The status counter
6. The comparator
7. The CCB address mux
8. The PROM bit decoder
9. The NAK response logic.

3.3.3.1 Loading a Single CCB

An IOLD command is used as an example. To the MLCP an IOLD command causes two distinct Megabus transfers. During the first transfer the Megabus loads the 24-bit byte address into the first CCB location in R/W RAM via the Megabus interface register and C-mux. During the second transfer the Megabus loads a 16-bit range quantity into the same CCB area.

Each of the 16 possible lines has four associated CCBs. This discussion assumes that this is the first IOLD command, and that CCB number 01 for line number 00 is being loaded. Table 3-18 shows the R/W RAM addresses associated with each CCB location. Table 3-16 shows the sequential steps involved in the execution of an IOLD command. It is assumed that the system has been initialized, leaving the scratch pad memory locations, load counter, status counter, and CCB counter at Zero (Zeroing occurs during the QLT test which occurs in initialization).

Loading of a CCB requires two commands:

- IOLD Instruction: As described above, this command occurs in two parts: Load Output Address (FC = 09) and Load Output Range (FC = 0D).
- Output CCB Control Instruction: (FC = 0F).

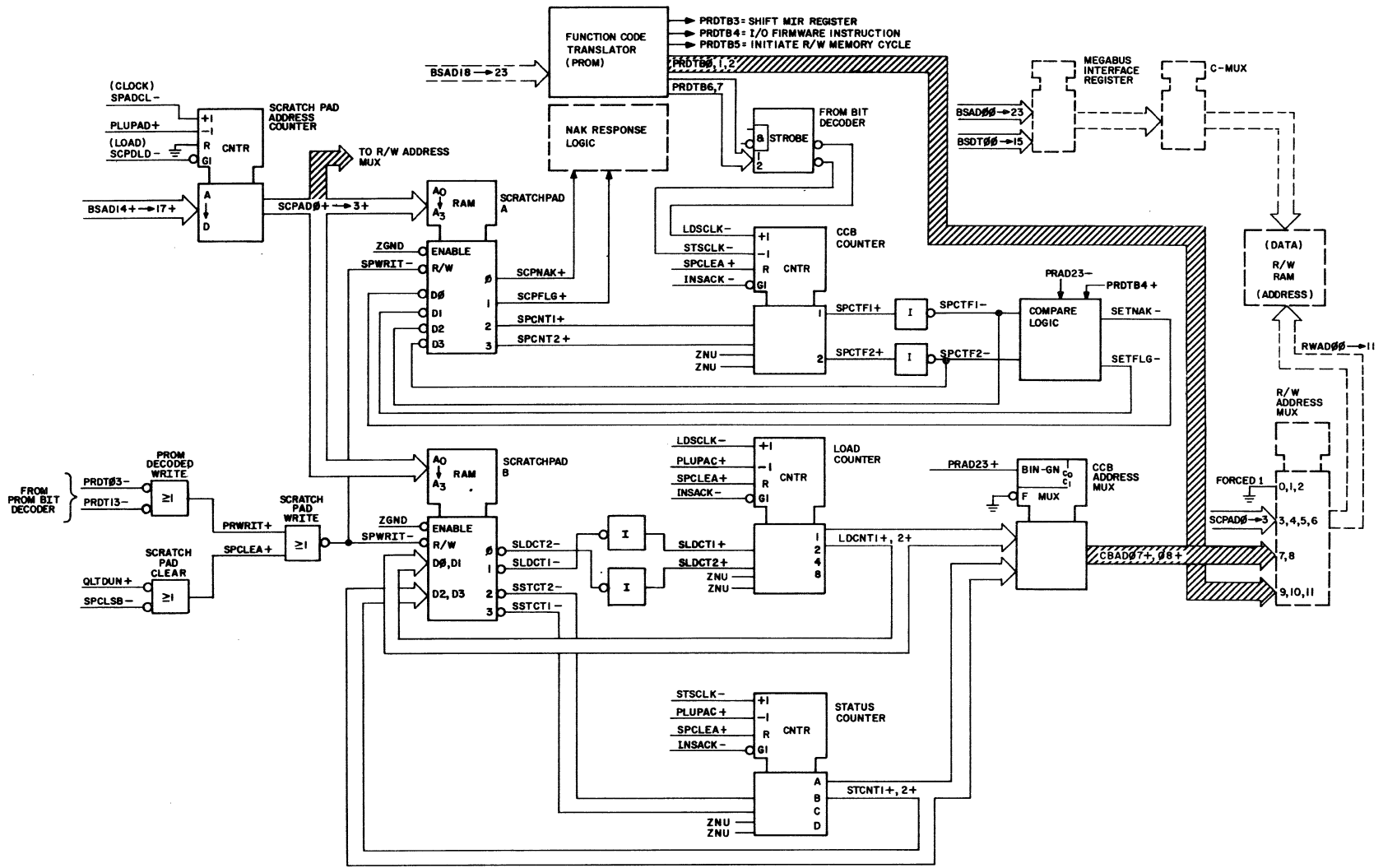


Figure 3-20 CCB Address Control Logic

HONEYWELL PROPRIETARY AND CONFIDENTIAL

Table 3-18 CCB Locations in R/W RAM

R/W RAM ADDRESS BITS RWAD00-RWAD11												INFORMATION LOADED INTO THE CCB	CCB, LINE
FORCED			LINE NO.				CCB NO.		FUNCTION CODE (PROM)				
0	1	2	3	4	5	6	7	8	9	10	11		
1	1	1	0	0	0	0	0	0	0	0	0	Address 16-23	CCB 0, Line 0
1	1	1	0	0	0	0	0	0	0	0	1	Address 8-15	
1	1	1	0	0	0	0	0	0	0	1	0	Address 0-7	
1	1	1	0	0	0	0	0	0	0	1	1	Range 8-15	
1	1	1	0	0	0	0	0	0	1	0	0	Range 0-7	
1	1	1	0	0	0	0	0	0	1	0	1	Flag	
1	1	1	0	0	0	0	0	0	1	1	0	Status	
1	1	1	0	0	0	0	0	0	1	1	1	Status	
1	1	1	0	0	0	0	0	1	X	X	X	Address, range, flag and status loaded into each CCB as above	CCB 1, line 0
1	1	1	0	0	0	0	1	0	X	X	X		CCB 2, line 0
1	1	1	0	0	0	0	1	1	X	X	X		CCB 3, line 0
1	1	1	0	0	0	1	0	0	X	X	X	Address, range, flag, and status loaded into each CCB as above	CCB 0, line 1
1	1	1	0	0	0	1	0	1	X	X	X		CCB 1, line 1
1	1	1	0	0	0	1	1	0	X	X	X		CCB 2, line 1
1	1	1	0	0	0	1	1	1	X	X	X		CCB 3, line 1
1	1	1	0	0	1	0	Y	Y	X	X	X	Address, range, flag, and status loaded into each CCB as above.	CCB 0, line 2
1	1	1	1	1	1	1	Y	Y	X	X	X		CCB 3, line 15

XXX = Function code increments from 000 to 111 as above.

YY = CCB number increments from 00 to 11 for each line.

Status is loaded by firmware upon completion of a CCB.

HONEYWELL PROPRIETARY AND CONFIDENTIAL

IOLD: Load Output Address (FC = 09)

Loading the output address is performed as follows:

1. The starting address of the CCB block in R/W RAM is established first. This address, RWAD00- RWAD11, is the R/W address mux, and is developed as follows:
 - a. Bits 0-2 are hard-wired at the mux input to force all Ones.
 - b. Bits 3-6 are generated by the Scratch Pad Address counter (SCPAD00 through SCPAD03). Bus address lines BSAD14 - BSAD17 load the line number (0000) into the scratch pad address counter.
 - c. Bits 7 and 8 are taken from the CCB address mux. The scratch pad address counter outputs address scratch pad memories A and B, which immediately load the CCB counter, the load counter, and the status counter with Zeros.
 - d. Bits 9-11 are taken from the function code translator.

For this portion of the IOLD command, the function code translator address causes the translator to produce an output of PRDTB7 - PRDTB00 = 01000000. PRDTB6 and PRDTB7 are taken to the PROM bit decoder where they are decoded to generate the function LDSCLK. This signal increments the load counter from 00 (which had been loaded from the scratch pad memory) to 01 (which is equivalent to CCB number 1). The load counter output feeds back as input to the scratch pad and is directed through the CCB address mux (CBAD07, CBAD08) to produce R/W address bits seven and eight.

Scratch pad address bits SCPAD00 - SCPAD03 address scratch pad A memory, which feeds the CCB counter a value of 00 (indicating that no CCBs have been loaded). Output from the CCB counter is taken into the compare logic, and is fed back as input to scratch pad A. Scratch pad A therefore has the count functions SPCNT1 and SPCNT2 set at 00 initially and at 01 at the end of the first IOLD command. During the second IOLD command, the value of 01 will be loaded into the CCB counter, which will then be incremented to a value of 10 (i.e., indicating the second CCB to be loaded for a given line). This process is repeated for each CCB that is loaded, up to a total of four CCBs per line. (Refer to subsection 3.3.3.2).

2. The starting address (RWAD00 - RWAD11) of the CCB clock in the R/W RAM has now been established. Next, the address data to be loaded into that address must be fetched from the Megabus interface register and directed through the C-mux and into the R/W RAM.

HONEYWELL PROPRIETARY AND CONFIDENTIAL

- a. The function code translator is incremented twice. (Refer to Table 3-17.) Locations 73 and 74 have bit three equal to a One. This bit shifts the Megabus interface register twice (once per location), thereby placing Megabus address bits 16-23 at the input to the C-mux (which feeds the R/W RAM). When the MIR is shifted twice, without initiating a R/W cycle (PRDTB5 low), the information at the output of the MIR is lost (BSAD08-BSAD15 and BSAD16-BSAD23).
- b. The function code translator is incremented to location 75. Bit five (PRDTB5) initiates a R/W memory cycle during which Megabus address bits 16-23 (containing BSDT08-15) are loaded into the CCB data block. Bit three (PRDTB3) is a One, and causes the Megabus interface register to be shifted once, thereby placing Megabus address bits 8-15 (containing BSDT00-07) at the C-mux.
- c. The function code translator is incremented to location 76. Bits 0-2 (PRDTB0-PRDTB2), having changed from 000 to 100, alter the R/W RAM address so that Megabus address bits 8-15 can be loaded into the proper location. Bit five (PRDTB5) initiates a second R/W memory cycle during which Megabus address bits 8-15 (containing BSDT00-07) are transferred from the Megabus interface register into the proper area of the CCB block via the C-mux. Bit three (PRDTB3) shifts the Megabus interface register at the end of the memory cycle, thereby placing Megabus address bits 0-7 at the input to the C-mux.
- d. The function code translator is incremented to location 77. Bits 0-2 (PRDTB0-PRDTB2), having changed from 100 to 010, alter the R/W RAM address so that Megabus address bits 8-15 can be loaded into the proper location. Bit five (PRDTB5) initiates a third R/W memory cycle during which Megabus address bits 8-15 (containing BSAD00-07) are transferred from the Megabus interface register into the proper area of the CCB block via the C-mux. Bit three (PRDTB3) shifts the Megabus interface register at the end of the memory cycle.

NOTE

During steps (a) through (d) above, function code translator bits PRDTB7 and PRDTB6 are both Zero. For this reason the PROM bit decoder does not generate LDSCLK, and the CCB counter and load counter have not been incremented (both are now at 01).

- e. The function code translator is incremented to location 78. Bit seven (PRDTB7), which is a One, resets the bus busy logic via the signal RSBSBY.

The output address portion of the IOLD command is now complete.

HONEYWELL PROPRIETARY AND CONFIDENTIAL

IOLD: Load Output Range (FC - 0D)

Following termination of the first Megabus transfer, the CPU automatically places the output range on the Megabus data lines (BSDT00 - BSDT15) and initiates the second Megabus transfer. During this transfer the output range, a quantity which defines the starting location of a Communication Data Block (CDB), is loaded into the Megabus interface register, and from there into the CCB in R/W RAM. Two consecutive read/write memory cycles are required.

1. The R/W RAM address into which the output range is to be loaded is established first. This address, RWAD00 - RWAD11, is directed through the R/W address mux, and is developed as follows:
 - a. Bits 0-2 are hard-wired at the mux input to force all Ones.
 - b. Bits 3-6 are generated by the scratch pad address counter (SCPAD00 through SCPAD03). Bus address lines BSAD14 - BSAD17 load the line number (0000) into the scratch pad address counter.
 - c. Bits 7-8 are taken from the CCB address mux. The scratch pad address counter outputs load scratch pad memories A and B, which immediately load the CCB counter, the load counter and the status counter with Zeros. Note that bits seven and eight define the CCB number, and that at this point CCB number 00 is indicated (it is CCB number 1 that is being loaded). This number will be changed to 01 when the load counter is incremented, as occurred during the output address portion of this IOLD command.
 - d. Bits 9-11 are taken from the function code translator.

For this portion of the IOLD command the function code translator address causes the translator to produce an output of PRDTB7-PRDTB0 = 01000000. PRDTB6 and PRDTB7 are decoded by the PROM bit decoder to generate the function LDSCLK. This signal increments the load counter from 00 to 01 to reflect the correct CCB number, which is 1.

2. Reference to Table 3-17 shows that the sequential steps required to load the output range are similar to the steps used to load the output address. Operation of the hardware is essentially the same as for loading output address. The bus busy logic is reset at location 109 by bit seven being a One, which raises the reset function RSBSBY.

The load output range portion of the IOLD command is now complete. The next and final step in loading the CCB is to load the control flag information and to store status of the CCB control logic in the scratch pads.

Output CCB Control (FC - 0F)

Loading of the CCB is completed by the Output CCB Control command, which loads flag and status information into the CCB and then stores CCB counter, load counter, status counter, and compare logic

HONEYWELL PROPRIETARY AND CONFIDENTIAL

status in scratch pads A and B. Table 3-17 shows the steps involved in the execution of this command.

1. Bus address bits BSAD14 - BSAD17 load the scratch pad address counter with 0000. SCPAD00 - SCPAD03 address scratch pads A and B, which immediately set both the CCB counter and the load counter to 00 (these will be incremented later to reflect CCB number 1, which is the CCB being loaded).
2. The first function code translator location addressed is decimal 120, in which bits PRDTB7 and PRDTB6 are a Zero and a One, respectively. As occurred during Output Address and Output Range operations, the CCB counter and the load counter are incremented from 00 to 01. The load counter output generates R/W RAM address bits seven and eight which are the CCB number.
3. The function code translator is then incremented twice, through decimal locations 121 and 122. In both of these locations bit PRDTB3 is a One causing the Megabus interface register to shift.
4. The function code translator is incremented to decimal location 123. Bit PRDTB5 is a One in this location, and it initiates a read/write memory cycle during which the flag information is loaded into the CCB area address in the R/W RAM (see Table 3-18).
5. The function code translator is incremented to decimal location 124. Another memory cycle is initiated (PRDTB5 = One) to load status information into the CCB. The same data from the last memory cycle is loaded into location seven of the R/W memory. Following the memory cycle the Megabus interface register is shifted again (PRDTB3 = One).
6. The function code translator is incremented to decimal location 125. This location causes the status information to be written into location six of the R/W memory.
7. The function code translator is incremented to decimal location 126. In this location the content of the CCB counter and compare logic are loaded into scratch pad A. The CCB counter at this time has the value 01, indicating completion of loading of CCB number 1. The content of the load counter, which at this time is 01, is loaded into scratch pad B. The content of the status counter, which at this time is 00, is also loaded into scratch pad B. At the beginning of the next IOLD instruction in which CCB number 2 will be loaded, these values will be loaded back into their respective counters, and then incremented by one to reflect CCB number 2.
8. The function code translator is incremented again to decimal location 127. Bit seven (PRDTB7), which is set to One, resets the bus busy logic via the signal RSBSBY.

Loading of CCB number 1, line 0 is now completed.

NOTE

Output Control Flag

1. Output address and range increment the CCB load pointer before use, but do not cause it to be stored back in the scratch pad. Therefore, the pointer is not permanently advanced. Because Output Control Flag does store the pointer, this command permanently advances the pointer.
2. This command, in normal mode, loads one byte of CCB flag information, but also clears the last two bytes of previous status. Therefore, software may test the status complete bit rather than rely on interrupts for notice of CCB completion.
3. In block mode this command loads two bytes of MLCP R/W RAM address and clears the last status as in note 2 above.
4. The pre-increment method used for the CCB load pointer makes the first CCB used after initialization CCB 1 (and not CCB 0). Thus the order of use of CCBs is 1, 2, 3, 0. The address of the first CCB for channel 0 is therefore E08 rather than E00.

3.3.3.2 Loading All CCBs in a Line

A maximum of four CCBs can be loaded per line, and the loading of each occurs as described in subsection 3.3.3.1. As each CCB is loaded, the content of the CCB counter is incremented and stored in scratch pad A. The value stored is loaded back into the CCB counter at the beginning of the subsequent IOLD command as previously described. The compare logic, which monitors the status of the CCB counter, generates the function SETNAK during loading of the fourth CCB. This is stored in scratch pad A. Should a fifth IOLD (Output Control) be issued, this function will, via the NAK respond logic, send a NAK response to the Megabus.

3.3.4 Input Status/Input Next Status

Table 3-17 describes the steps and hardware events which occur during Input Status and Input Next Status commands. Functional operation of the logic is essentially the same as occurred during the IOLD and Output CCB Control commands. Input Status, which begins at function code translator address (decimal) 192, generates a read/write memory cycle, fetches the status information from R/W RAM and loads it into the Megabus interface register, and sets a CPU request. Input Next Status, in addition to performing the same steps as Input Status, increments the CCB status pointer and writes this updated status pointer information into the scratch pad memory.

The normal sequence of events is to load a CCB or to test a CCB to perform an IOLD and an Output CCB control command, and then to perform an Input Next Status followed by an Input Status command.

HONEYWELL PROPRIETARY AND CONFIDENTIAL

The CCB address control logic handles Input Next Status such that data can never be read from a CCB that has not been loaded first. This guarantee is implemented by the status counter (STCNT1, 2) and the compare logic. Each time an Input Next Status is issued, the status counter is incremented and stored in scratch pad B. In this type of command it is the status counter that generates the CCB number from which data is to be fetched (STCNT1, 2 generate R/W RAM address bits seven and eight via the CCB address mux).

Meanwhile, the compare logic monitors the CCB counter, which is being decremented. If this counter decrements to 00, the compare logic issues the signal Set Flag (SETFLG), which is stored in scratch pad A. If another Input Next Status is issued, the MLCP responds with a NAK, due to SETFLG, which is stored in the scratch pad. SETFLG is removed from the scratch pad by the next IOLD command, which loads new values into the scratch pad via the scratch pad address counter.

3.3.5 PROM Bit Decoder Logic

PROM bit decoder logic is shown in Figure 3-21. The selected output is determined by the state of the function code translator bits six and seven. The outputs are gated to the indicated logic areas to control the hardware operation.

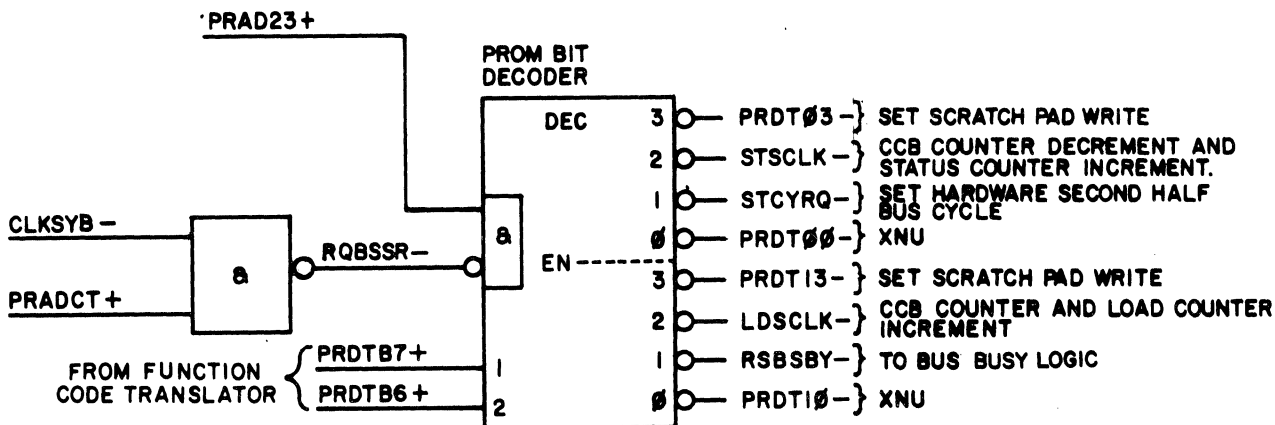


Figure 3-21 PROM Bit Decoder Logic

3.4 MEGABUS INTERFACE REGISTER AND CONTROL LOGIC

The Megabus Interface Register (MIR) and control logic provides the MLCP with an interface to the Series 60 Level 6 Megabus. Figure 3-22 illustrates the major hardware areas of this logic, which are:

1. A 40-bit Megabus interface register
2. A parity generator
3. A parity buffer
4. A parity checker.

3.4.1 MIR Organization and Control (Figures 3-22 through 3-25)

The MIR contains 24 bits of address (MYAD00 through MYAD23) and 16 bits of data (MYDT00 through MYDT15) information. It is composed of eight shift register chips arranged to provide an eight-bit wide byte stacked in five rows. These shift registers provide parallel-to-serial conversion when the MLCP is receiving data from the Megabus, or serial-to-parallel conversion when the MLCP is sending data to the Megabus.

For all firmware-implemented output instructions, the Megabus parity buffer and the Megabus parity checker (PCKEVE+) are used to compare the parity on selected address and data information in the MIR with the parity bit that was sent on the Megabus to accompany the same address or data as it was loaded into the MIR. Incorrect parity (PCKEVE+) is sent to the I-mux where it is checked by firmware.

For all input instructions a parity bit is generated for data being loaded into the MIR from the B-mux. This parity bit is developed by the B-mux parity generator for each eight bits loaded into the MIR, and is then stored in the Megabus parity buffer for subsequent transmission on the Megabus.

Figure 3-23 shows the functionality of the MIR and B-mux, together with their inputs, outputs, and controlling functions. Shifting of the MIR is controlled by bit three of the function code translator (PRDTB3) for hardware-implemented instructions, or by firmware (via control store bits CSDT27 through CSDT31 and the strobe decoder) for firmware-implemented instructions. The appropriate B-mux input is selected by function code translator bits one and two (PRDTB1 and PRDTB2) for input instructions and is loaded into the MIR via its shift (serial) inputs. During output instructions, information on the Megabus address and data line is loaded in parallel into the MIR.

Figure 3-24 shows the MIR organization when the MLCP is acting as master and is sending data to the Megabus. Row 5 of the MIR is loaded from the B-mux. The MIR is then shifted by shift signal SFTBDR, with row 5 going to row 4, row 4 going to row 3, etc. Row 5 is then loaded with the next eight bits of information and the MIR is again shifted by SFTBDR. If this were a firmware-implemented output instruction, the shift signal SFTBDR would be raised by the function SFBRBS-, an output from the strobe-decoding logic (i.e.,

HONEYWELL PROPRIETARY AND CONFIDENTIAL

the logic which decodes control store bits CSDT27 through CSDT31; for a detailed description of this logic, refer to subsection 3.2.8). If this were a hardware-implemented instruction, the shift signal SFTBDR would be raised by bit three of the function code translator, PRDTB3 (for a detailed description of this logic, refer to subsection 3.3.1.2 and Table 3-17).

Figure 3-25 shows the MIR organization when the MLCP is acting as a slave and is receiving data from the Megabus. The 40 input lines from the Megabus are loaded in parallel by the load (SETBDR) and shift (SFTBDR) signals. The MLCP then shifts the MIR one row at a time, and selects either the C-mux or the M-mux (or both) as a data path for the MIR output (MYAD16 through MYAD23). Mux selection is described in detail in Table 3-7. Control over the MIR shift signal SFTBDR is supplied by either control store bits 27-31 or by bit three of the function code translator, as described above.

3.4.2 MIR Operation When MLCP Is Slave

This subsection describes the operation of the MIR when the MLCP is acting as a slave and is receiving data from the Megabus. The example followed is that of a hardware-implemented instruction named IOLD (Output Address)/IOLD (Output Range) instruction, during which the output address and range portions of a CCB are loaded into the proper address in the R/W RAM. Only those aspects of this instruction which relate to the MIR are described here; for a detailed, step-by-step description of how this instruction is implemented, refer to subsection 3.3.3.

When the Megabus makes a request, address bits BSAD08 through BSAD13 are used to determine if the request is for the MLCP. These bits are compared by the channel address comparator against the MLCP channel address that is set into the channel address hex rotary switches (BSSW08 through BSSW13). (Refer to Figure 3-23.) If the two compare (MYCHSP+), and if there is a line adapter present (MYDL20), and if the MLCP is not busy (MYBSYR), the MLCP will acknowledge the bus request and will raise the load MIR function (SETBDR) so that the information on the address and data lines will be loaded into the MIR during the next clock cycle (SFTBDR). The content of the MIR following loading is as shown in Figure 3-26.

The MLCP now examines the first row of MIR data, which contains address bits BSAD16 through BSAD23. Since this is a hardware instruction, both the M-mux and C-mux are selected. These address bits are taken via the M-mux to the microprocessor, and via the C-mux to the R/W RAM. Meanwhile, BSAD18 through BSAD23 supply an address for the function code translator to determine which instruction to execute.

As the instruction is executed the function code translator shifts twice, and each time its output has bit three (PRDTB3) set to a logic One, which causes the MIR to shift. During the first MIR shift, BSAD16 through BSAD23 are lost and BSAD08 through BSAD15 are located in row 1 of the MIR. During the second shift, BSAD08 through BSAD15 are lost, and BSAD08 through BSAD15 are located in row 1 of the MIR. This data is loaded into the selected CCB address in the R/W RAM via the C-mux.

HONEYWELL PROPRIETARY AND CONFIDENTIAL

The function code translator shifts again, and PRDTB3 is a logic One, causing the MIR to shift. Following this shift, row 1 of the MIR contains data bits BSDT00 through BSDT07. This data is also loaded into the address area of the CCB in the R/W RAM.

The function code translator shifts a fourth time, and PRDTB3, again a logic One, shifts the MIR again, and address bits BSAD00 through BSAD07 are loaded into the address area of the CCB in the R/W RAM*.

This completes the description of the MIR operation for the Output Address portion of the IOLD instruction. This instruction is completed as described in subsection 3.3.3.1.

Had this been a firmware-implemented instruction, the firmware would shift the MIR via strobe decoding of control store bits 27-31. (Refer to subsection 3.4.1.) Firmware must shift the MIR as described above in order to examine its content and determine the instruction to be executed. The content of the MIR is transferred via the M-mux to the microprocessor.

3.4.3 MIR Operation When MLCP Is Master

This subsection describes the operation of the MIR when the MLCP is acting as a master and is sending data to the Megabus. The example followed is that of a hardware-implemented instruction named Input ID. Implementation of this instruction requires that the MLCP identify that it is being addressed, load the information contained in the Megabus address and data lines into the MIR, analyze the function code to determine the instruction to execute, and then load the desired information into the MIR and transfer it to the Megabus. Address comparison, function code translation, and MIR loading from the Megabus are accomplished as described in subsection 3.4.2. Only those aspects of this instruction which relate to the MIR and B-mux operation are described here.

Assuming that the MIR has been loaded from the Megabus and that the function code translator has been addressed by bits BSAD18-23, this instruction executes in sequence the steps shown in Table 3-17 (address decimal 48). Figure 3-27 shows the content of the MIR during each step of the Input ID instruction execution.

The function code translator output at decimal address 48 has bit three (PRDTB3) set to a logic One. This bit raises the shift signal SFTBDR to shift the MIR. The serial input to the MIR is from the B-mux, whose input is selected by function code translator output bits PRDTB1 and PRDTB2 (see Figure 3-23).

The function code counter (an input to the function code translator address) is incremented by one, to decimal address 49. PRDTB1 and PRDTB2 again select the B-mux input to be loaded into the MIR,

*BSAD00 through BSAD07 are typically Zeros, as they are not required by the present CPU R/W RAM. These address bits can be used in the future, should a larger R/W RAM be available to the CPU.

HONEYWELL PROPRIETARY AND CONFIDENTIAL

and PRDTB3, again a One, shifts the MIR a second time. The content of the MIR following this shift and load operation is as shown in Figure 3-27. Since this is an input instruction, the MLCP must send the address of the requesting device when it places the requested data on the Megabus data lines. This address is in MIR rows 1 and 2, and is sent on Megabus address lines BSAD08 through BSAD23 (BSAD16 through BSAD23 must be Zeros). The requested data is sent on Megabus data lines BSAD00 through BSAD15.

The function code counter increments to decimal address 50, at which time the Megabus busy logic in the MLCP is reset, and the MLCP is ready for another instruction.

3.4.4 Parity Checking and Generation (Figure 3-28)

3.4.4.1 Parity Checking - Data Received by MLCP from Megabus

For all firmware-implemented commands, and for data from the memory, data received from the Megabus is checked for odd parity by the Megabus parity checker (no parity checking is performed for hardware-implemented commands). As the MIR is being loaded, the following Megabus parity lines are simultaneously loaded into the Megabus parity buffer:

1. BSAP00+ - Odd parity for BSAD00 through BSAD07
2. BSDP00+ - Odd parity for BSDP00 through BSDP07
3. BSDP08+ - Odd parity for BSDP08 through BSDP15.

The parity buffer output (PCKBIT+) is one input to the Megabus parity checker. The remaining eight inputs are from the output of the MIR. PCKBIT+ is XORed with the MIR output and should result in PCKEVE+ remaining low as long as the parity on the MIR bits equals the parity bit on the associated Megabus parity line.

After checking the content of row 1 of the MIR (which contains BSAD16-23), both the MIR and the parity buffer are shifted by SFTBDR (note that this signal does not occur in order to check parity; rather, it occurs in the normal process of instruction execution as described in subsection 3.4.1). This places address bits BSAD08-15 in row 1 of the MIR and at the input to the Megabus parity checker, and the parity on these bits is checked.

Each time the MIR is shifted, new address or data is placed in row 1, and its parity is checked against the associated Megabus parity line. Detection of a parity error raises the parity error function PCKEVE+. An input to the I-mux, PCKEVE+ is continuously being monitored by firmware, which will take the appropriate action should a parity error be detected.

Firmware is not concerned with address information BSAD08-23, and there is no associated Megabus parity line. It is possible, therefore, for PCKEVE+ to be raised as this information is shifted into row 1 of the MIR and checked by the parity checker. This is a no-effect condition, because firmware does not check PCKEVE+ during the cycles in which address bits 8-23 are checked for parity.

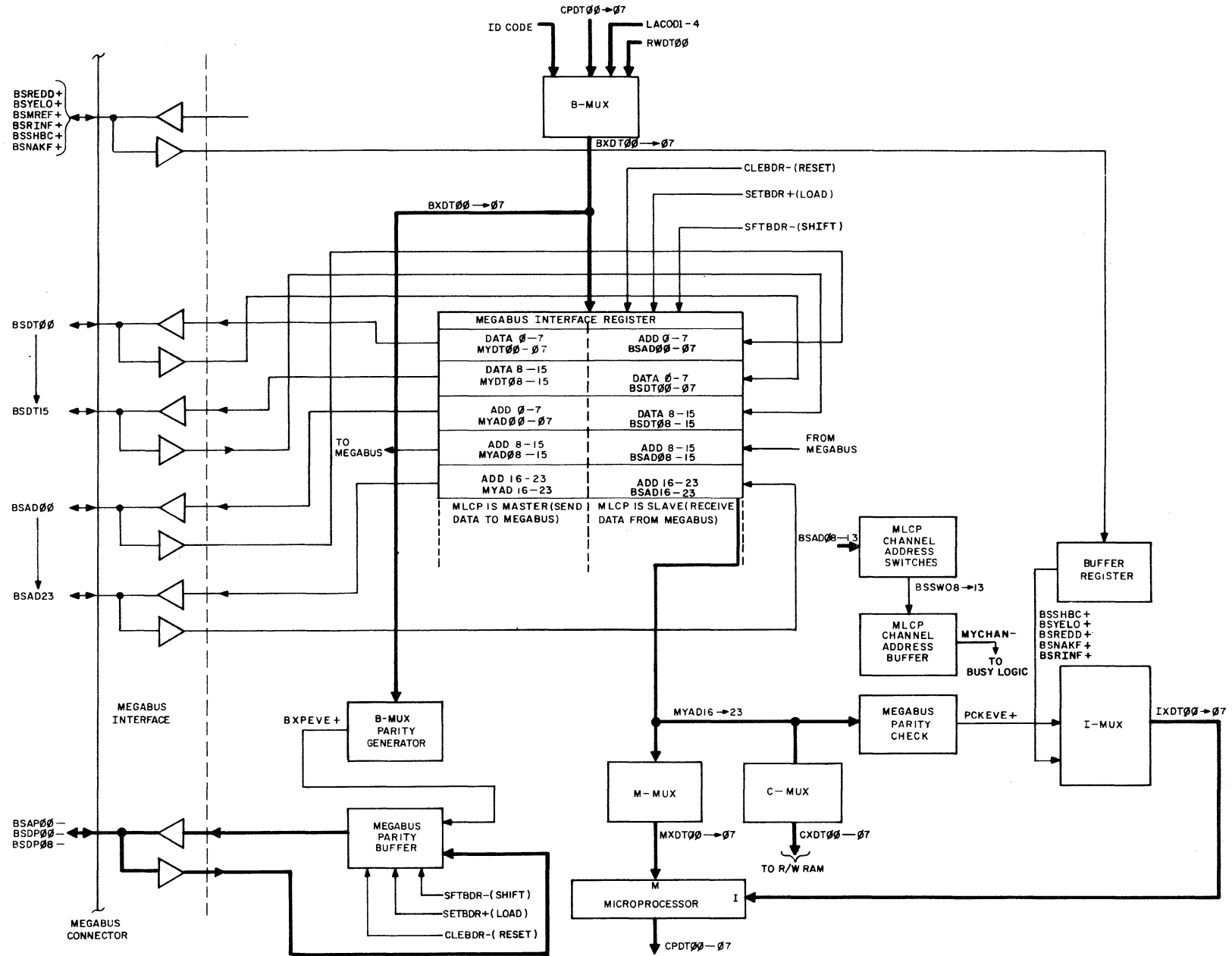


Figure 3-22 Megabus Interface Register and Control Logic

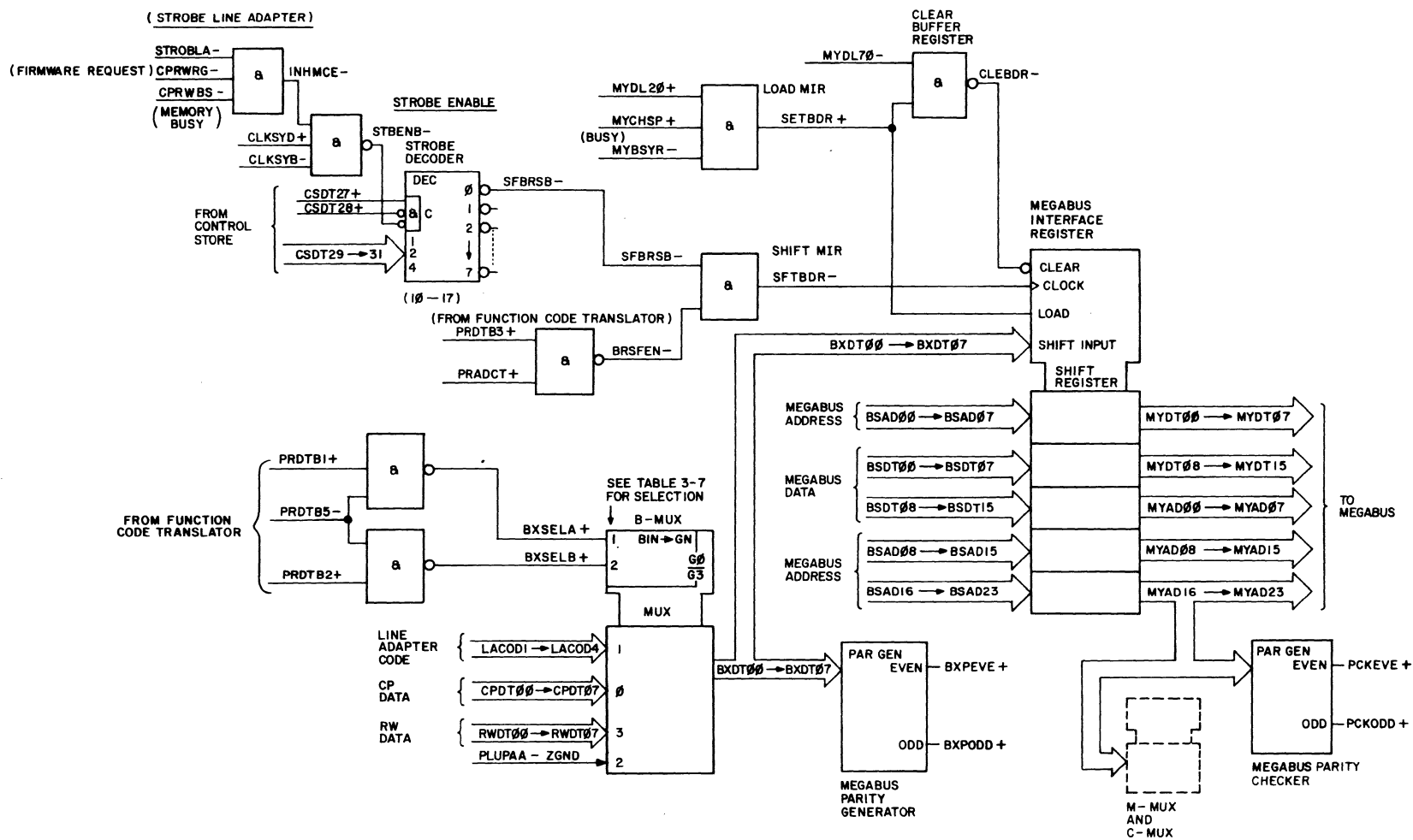


Figure 3-23 Megabus Interface Register (MIR) and B-Mux Functionality

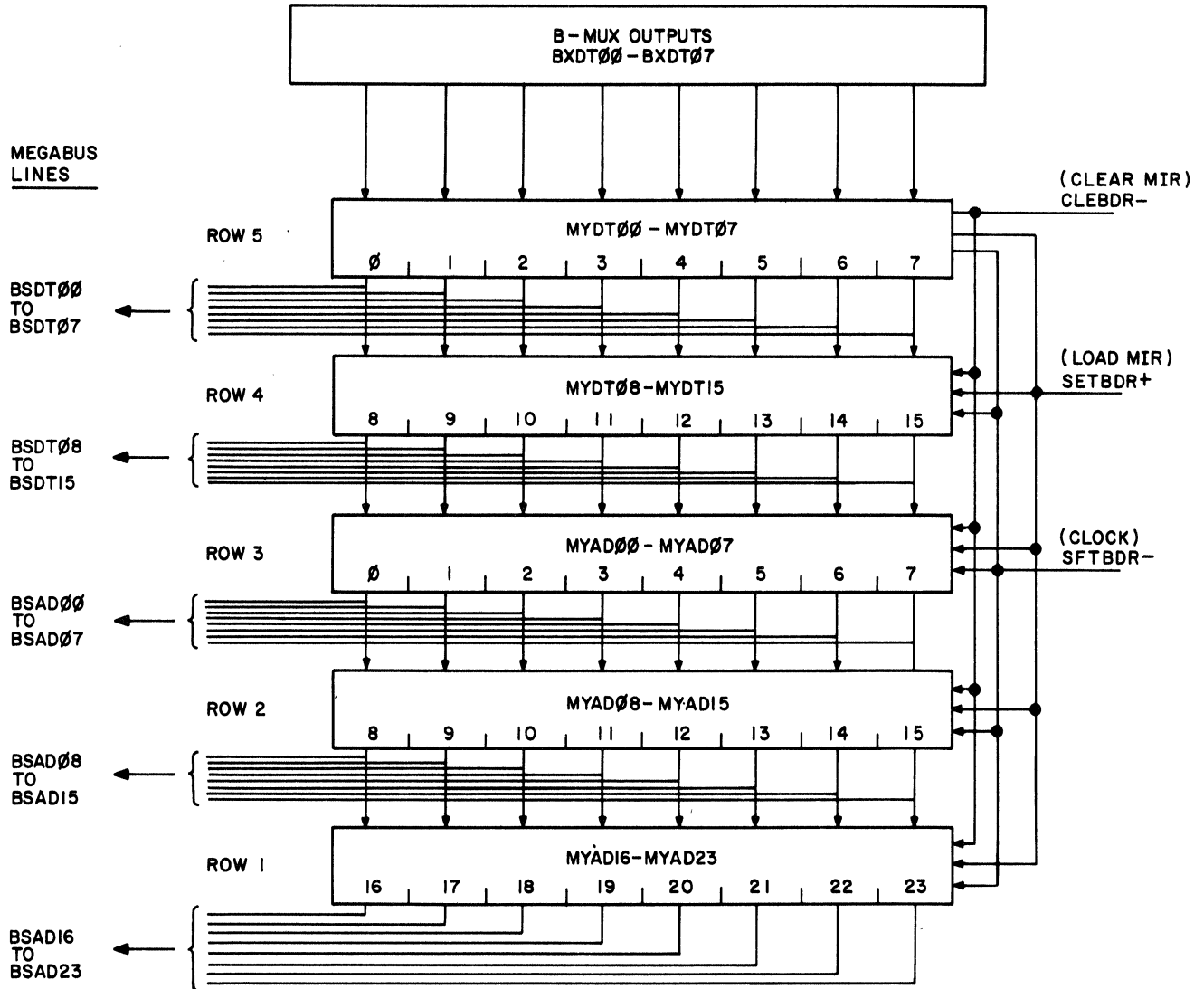


Figure 3-24 MIR (MLCP Is Master Sending Data to Megabus)

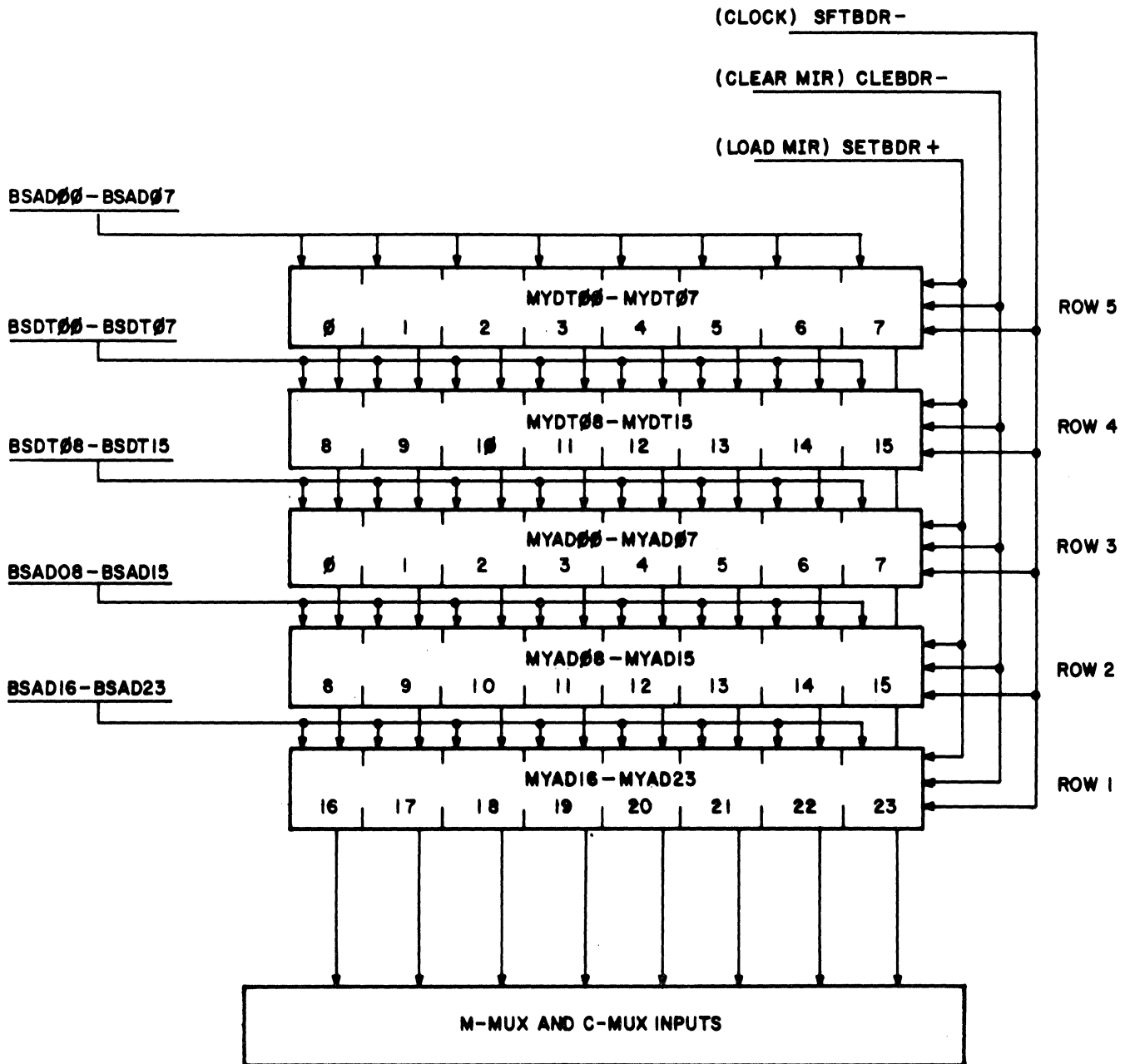


Figure 3-25 MIR (MLCP Is Slave Receiving Data from Megabus)

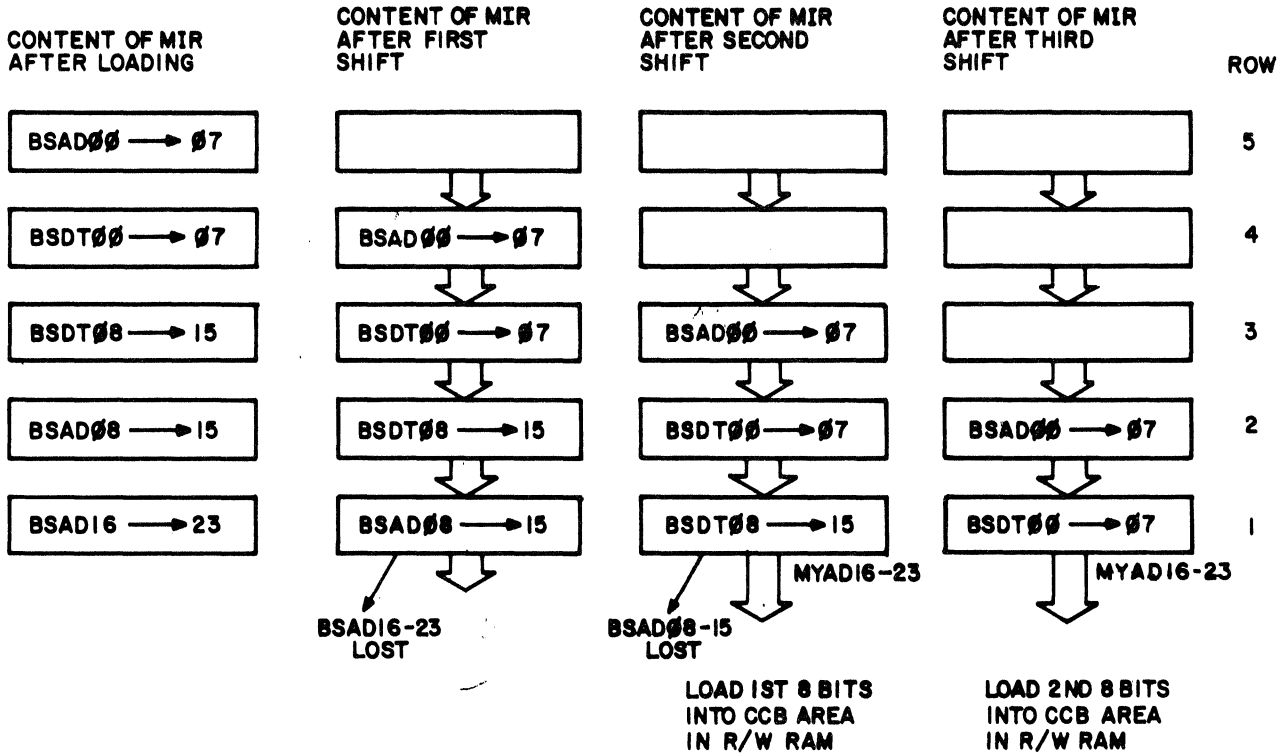


Figure 3-26 Content of MIR During an Output (IOLD Output Address) Instruction

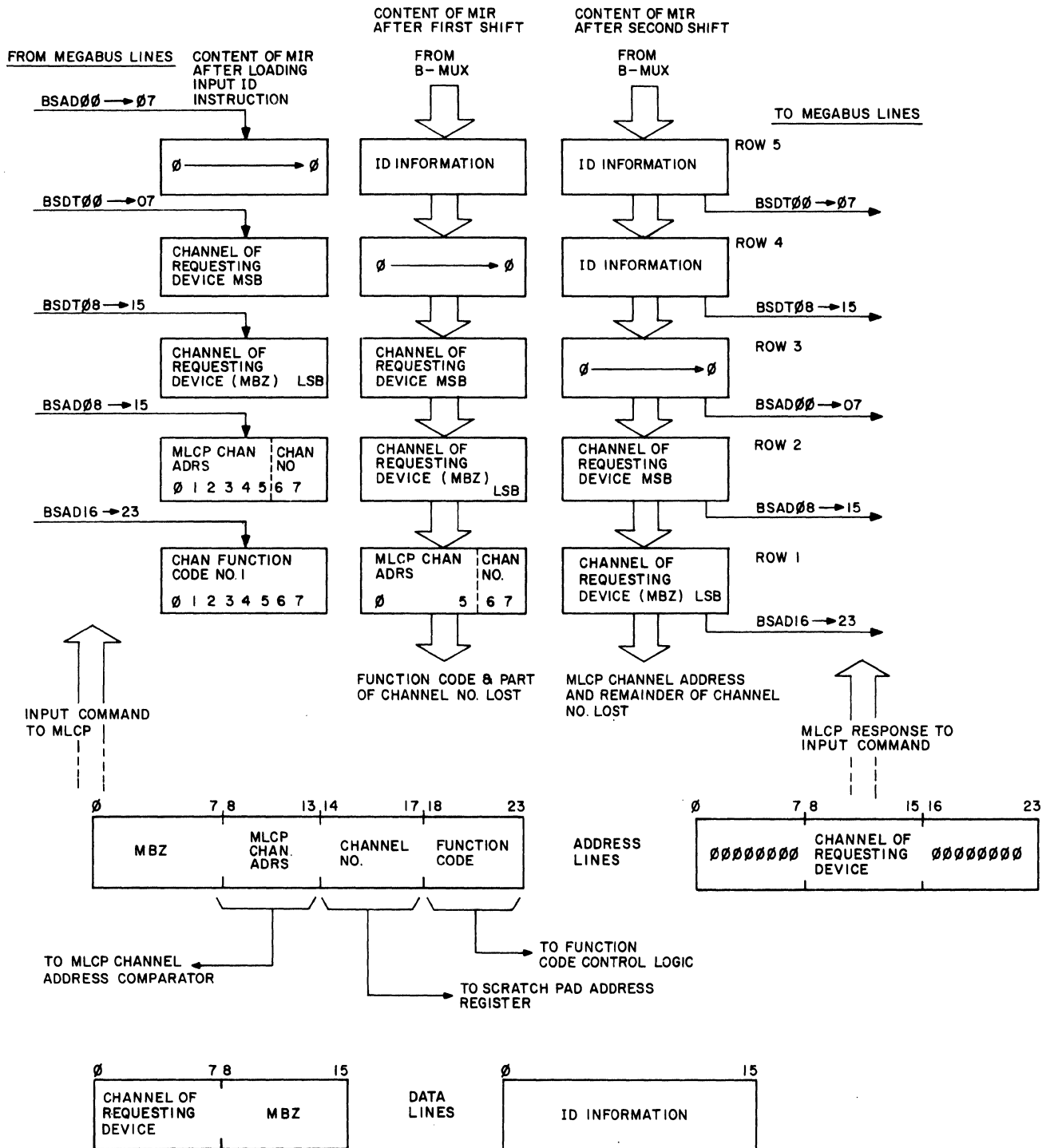


Figure 3-27 Content of MIR During an Input (Input ID) Instruction

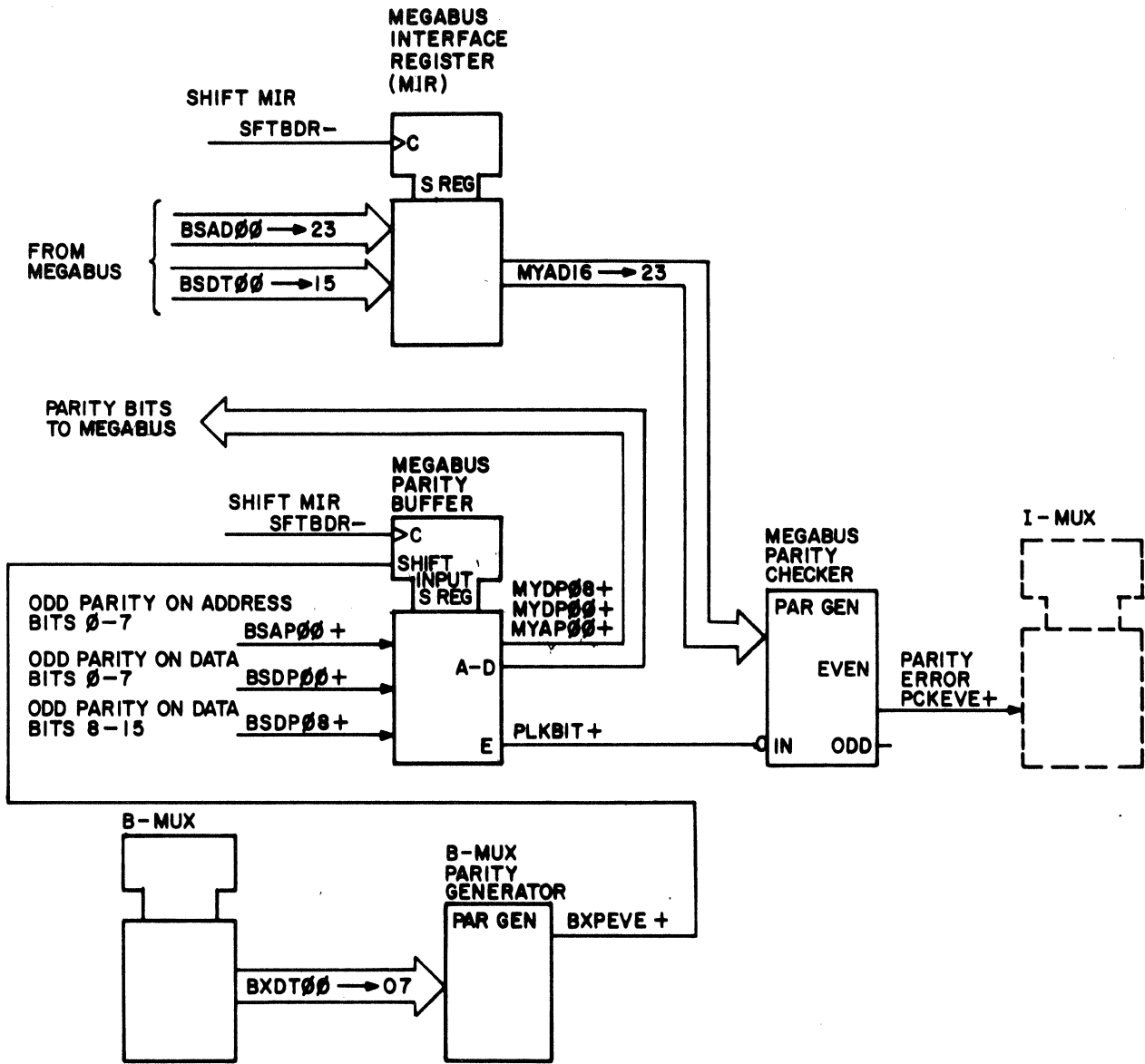


Figure 3-28 Parity Checking and Generation Logic

3.4.4.2 Parity Generation - Data Sent from MLCP to Megabus

A parity bit is generated for each eight bits loaded into the MIR from the B-mux by the B-mux parity generator. The output of this generator (BXPEVE+) becomes the serial input to the parity buffer. Each time the MIR is shifted and a new byte of information is loaded into row 5 from the B-mux, the parity buffer is also shifted. Each output from the B-mux parity generator corresponds to one of the five rows in the MIR, and contains the parity bit for that row (except for rows 1 and 2). As the content of MIR is sent to the Megabus, the associated parity bits are also sent to the Megabus as illustrated in Figure 3-28.

3.5 READ/WRITE RAM LOGIC

The read/write memory is a 4K by eight-bit MOS type random access memory. Figure 2-6 (Section II) shows a map of the R/W RAM. Information loaded into R/W RAM by the CP program is used by the MLCP to service the attached line adapters. As shown in Figure 3-29, this logic consists of the following:

1. 4K by eight-bit R/W RAM
2. Request memory logic
3. Refresh address counter
4. RAM busy logic
5. R/W address multiplexers
6. R/W data buffer
7. Start write cycle logic
8. Start memory cycle logic
9. Refresh timing/request logic
10. Hardware request logic
11. Firmware request logic.

Figure 3-30 is an intermediate block diagram of the R/W RAM address control logic described in the following subsections.

3.5.1 Memory Cycle Requests

The R/W RAM can be addressed (i.e., a memory request can be made) by the microprocessor (for commands being implemented by firmware), by the function code control logic (for commands being implemented by hardware), or by the refresh logic. Memory requests are prioritized as follows:

1. Refresh request - lowest priority
2. Hardware request - second highest priority
3. Firmware request - highest priority.

Figure 3-31 shows the logic required to initiate a memory cycle. The R/W RAM cycles when its clock signal CLKWRA+ is raised. CLKWRA+ is raised in response to a refresh request (RFBUSY-), to a hardware request (IORWBS-), or to a firmware request (CPRWBS-) in accordance with the priority levels described above.

3.5.2 I/O Hardware Requests for Memory

This subsection describes the operation of the R/W RAM and associated logic in response to an I/O command to be implemented by MLCP hardware. As shown in Figure 3-30 the I/O Hardware Request function (IORWST) is set when bit five of the function code translator (PRDTB5) is a logic One if the memory is not busy (RWBUSY-) and no firmware request is present (CPRWRQ-).

Address bit 23 of the I/O command determines whether the memory cycle is to be a read or a write cycle. This bit is latched into the function code latch as function PRAD23+. If PRAD23+ is a logic One, the function WRITPS- is low, the write enable flip-flop sets, and Write Enable (RWWRITE-) goes low to select the write function at the R/W RAM. If PRAD23+ is a logic Zero, the read function is selected. All I/O hardware requests address the CCB area in the R/W RAM.

HONEYWELL PROPRIETARY AND CONFIDENTIAL

At the R/W address multiplexer, the address bits are generated as follows:

<u>Bits</u>	<u>Source</u>
0,1,2	These bits are forced to logic Ones (to select upper 512 bytes of the R/W RAM)
3,4,5,6	From the scratch pad address counter (SCPAD00-03)
7,8	From the CCB address multiplexer (CBAD07, 08)
9,10,11	From the function code translator (PRDT00, B1, B2)

A detailed description of the development of these address bits is provided in subsection 3.3.3.

Hardware requests (IORWBS) have second highest priority and will therefore block refresh requests (RFBUSY-) and will be blocked by firmware requests (CPRWBS+) as shown in Figure 3-31.

Setting of the hardware I/O Request (IORWST) flip-flop sets the I/O memory busy (IORWBS) flip-flop, which in turn sets the memory busy (RWBUSY) flip-flop. The System Clock (CLKSYA) provides the correct memory cycle timing to complete the memory access and, following the access, to reset the request and busy flip-flops.

3.5.3 I/O Firmware Requests for Memory

This subsection describes the operation of the R/W RAM and associated logic in response to an I/O command to be implemented by MLCP firmware. As shown in Figure 3-31 the firmware request flip-flop (CPRWRQ) is set in response to a firmware request (CRPWSB). If the memory is not busy, the memory busy flip-flop (CPRWBS) is set, which in turn sets the memory busy function (RWBUSY), thereby making the R/W RAM busy to other requests.

Control store bit seven (CSDT07) determines whether the memory cycle is to be a read or a write cycle as shown in Figure 3-30. If CSDT07 is a logic One, the write function is selected at the R/W RAM; if it is a logic Zero, the read function is selected.

At the R/W address multiplexer, the address bits for R/W RAM are generated as follows:

<u>Bits</u>	<u>Source</u>
0 - 3	From the data lines of the AC register in the microprocessor (CSDT04 - 07)
4 - 11	From the memory address register in the microprocessor (RWAD04 - 11)

These address lines from the microprocessor bypass the R/W address mux and are wire-ORed to the address mux outputs.

Firmware requests have the highest priority and will therefore block refresh requests (RFBUSY-) and hardware requests (IORWBS) as shown in Figure 3-31.

The system clock (CLKSYA) provides the correct memory cycle timing to complete the memory access and, following the access, to reset the request and busy flip-flops.

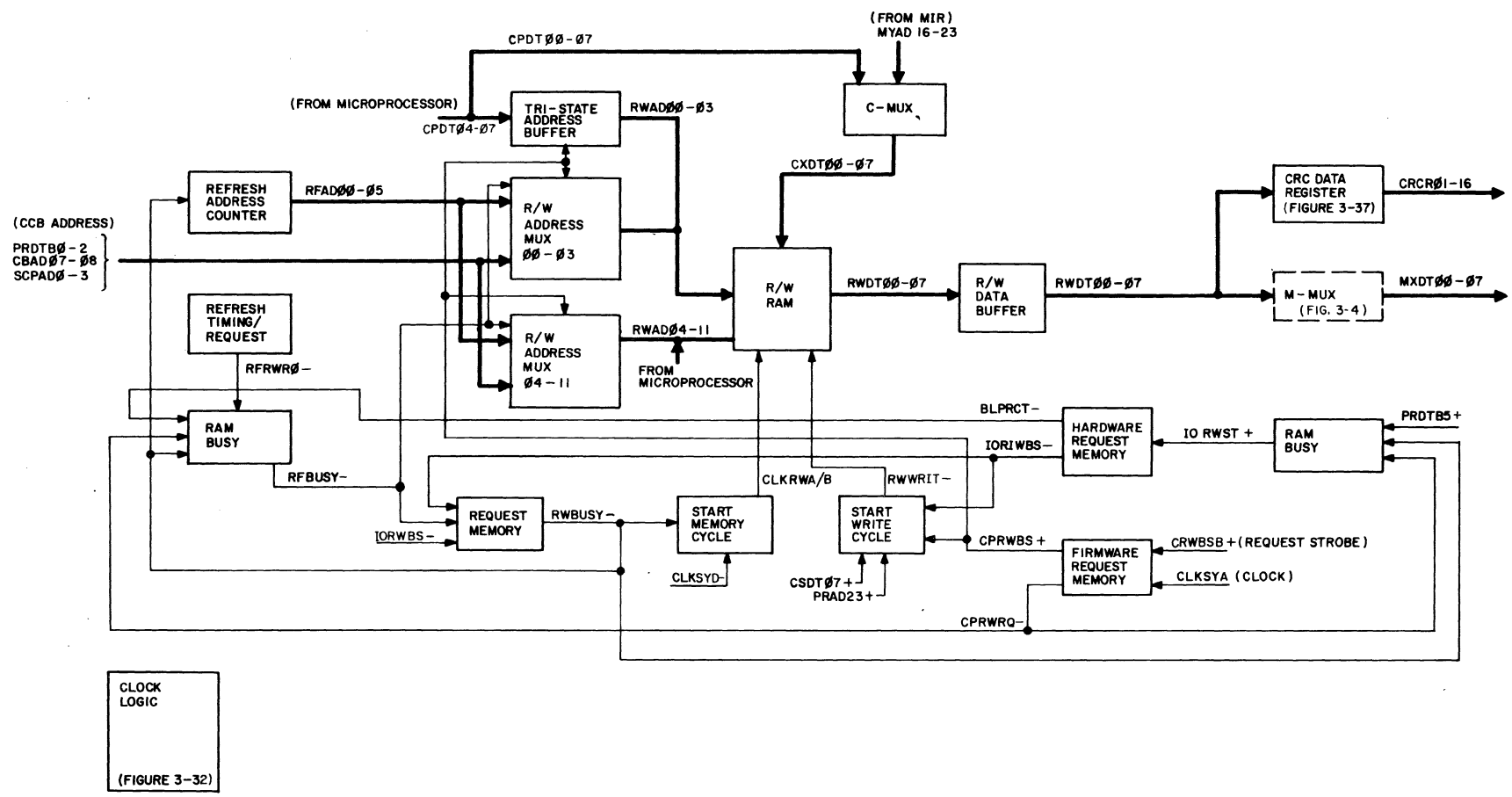


Figure 3-29 R/W RAM Logic Functional Block Diagram

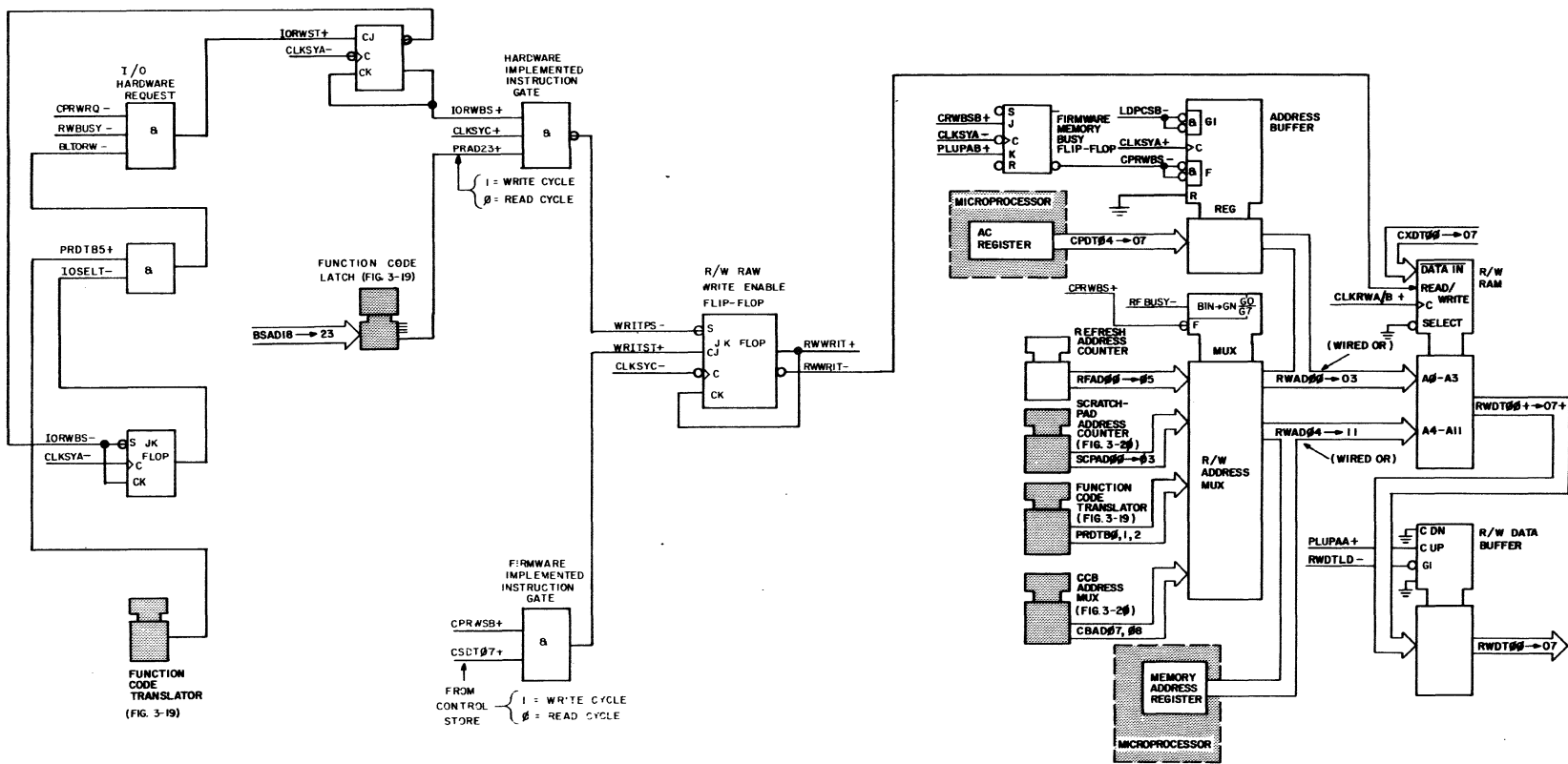


Figure 3-30 R/W RAM Address Control Logic

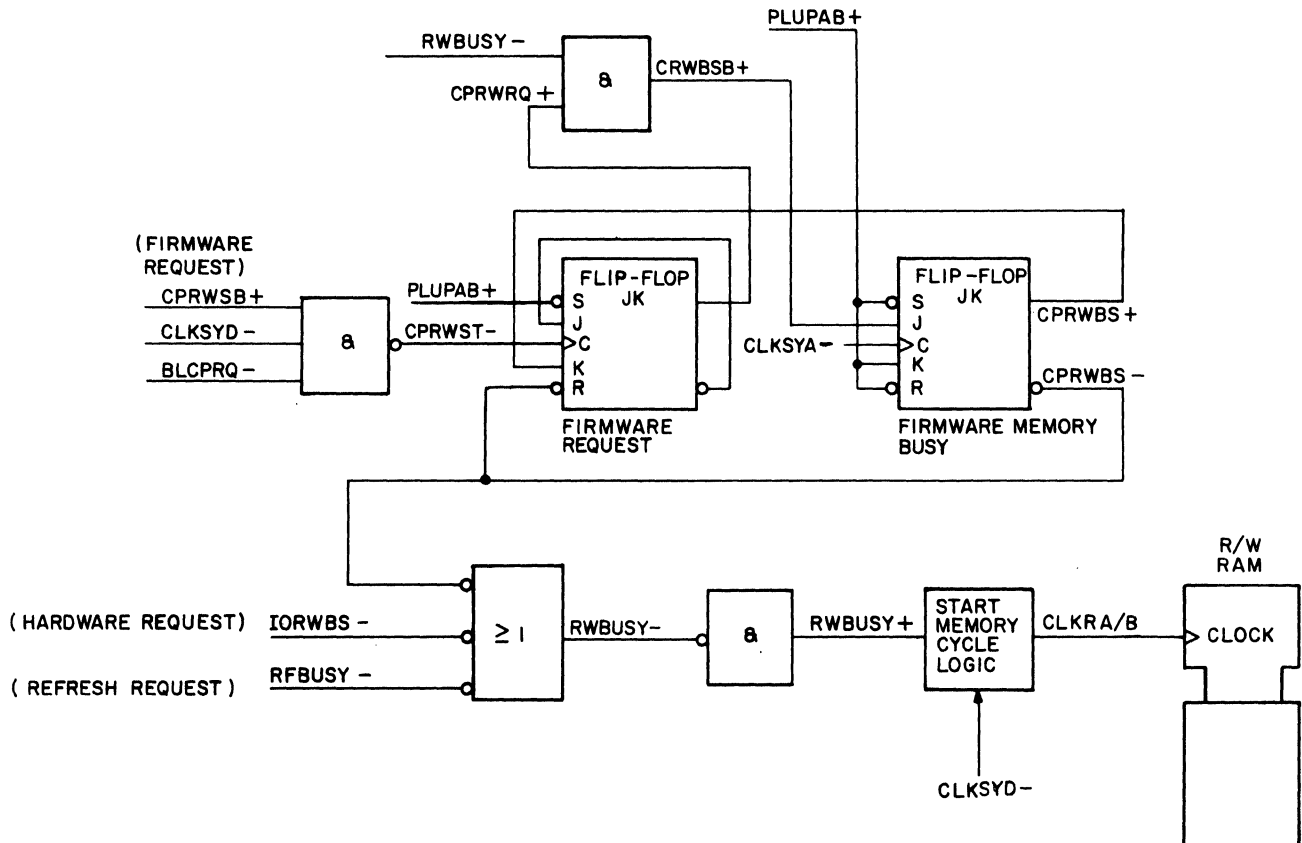


Figure 3-31 Request Memory/Start Memory Cycle Logic

3.5.4 System Clock and MLCP Timing

The System Clock shown in Figure 3-32 supplies the basic system clock signals (CLKSYA through CLKSYD) and acts as an input to the clocking logic located throughout the MLCP. Figure 3-32 sheet 1 portrays the system clock logic with a baud rate generator configured by jumpers and Figure 3-32 sheet 2 portrays the system clock logic with a baud rate generator configured by a hex rotary switch. Clocking applications are as follows:

<u>Area</u>	<u>Clock Functions</u>
R/W memory timing	CLKSYA - CLKSYD
Microprocessor timing	CLK302
Microprogram control unit timing	CLK301
Line adapter timing	CLKASC, CLKSYN
Line adapter strobe	STOBLA

Figure 3-33 illustrates the major clock timings and operational timings within the MLCP. Figure 3-33A portrays normal operation. Figure 3-33B portrays long cycle operation, in which the MLCP runs at a slower rate because the trailing edge of CLKSYD is extended 34 nanoseconds via the Long Cycle (LONGCY) function. This function, normally tied to a pullup resistor, may be grounded to force the MLCP to run slower than normal for test purposes.

Synchronous communications line adapters require a clock signal from the MLCP to operate in test mode or direct connect mode. This signal, Synchronous Clock (CLKSYN), is derived from a baud rate generator which produces one of 16 frequencies.

Figure 3-32 sheet 1 shows the system clock and baud rate generation logic for an MLCP with a board assembly number of 60127901. The baud rate of the synchronous clock output is determined by the insertion or removal of jumpers at the input to the baud rate generator as indicated in Table 3-19.

Figure 3-32 sheet 2 shows the system clock and baud rate generation logic for an MLCP with a board assembly number of 60130470 or 60130810. The baud rate of the synchronous clock output is determined by the setting of a hex rotary switch located in location E04 of the MLCP. Table 3-19 indicates the baud rate generated for each hex rotary switch setting.

The Asynchronous Clock (CLKASC) signal, shown in Figures 3-32 sheet 1 and sheet 2, is the basic clock for asynchronous communications line adapters. This clock operates at a frequency of 921.6 kHz and is delivered to the baud rate generator in the asynchronous CLA. See the appropriate asynchronous line adapter manual for baud rates generated in the line adapter.

HONEYWELL PROPRIETARY AND CONFIDENTIAL

A. JUMPER CONFIGURATION, BOARD ASSEMBLY NUMBER 60127901-001

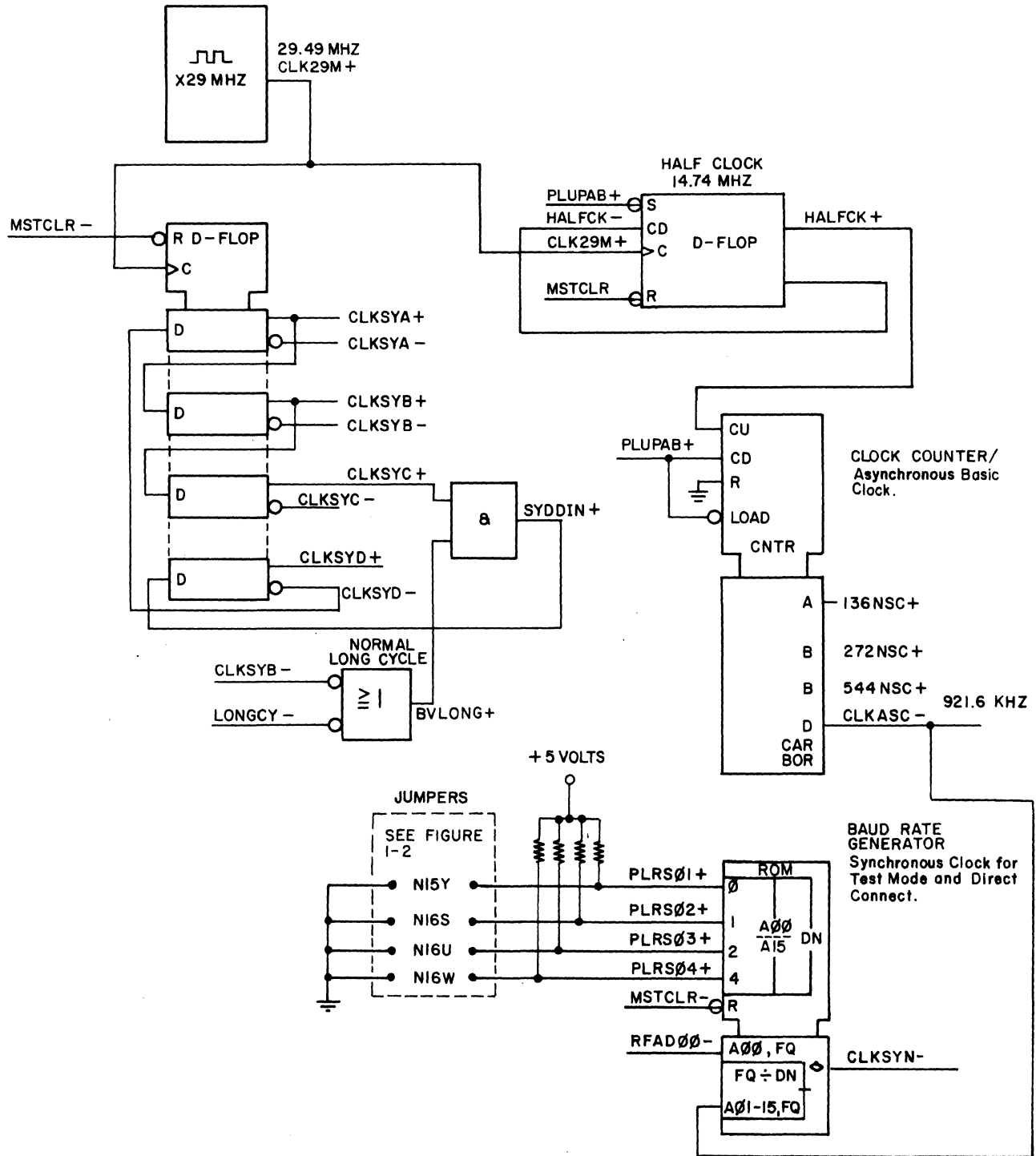


Figure 3-32 System Clock (Sheet 1 of 2)

B. HEX ROTARY SWITCH CONFIGURATION, BOARD ASSEMBLY
 NUMBER - 60130470 OR 60130810

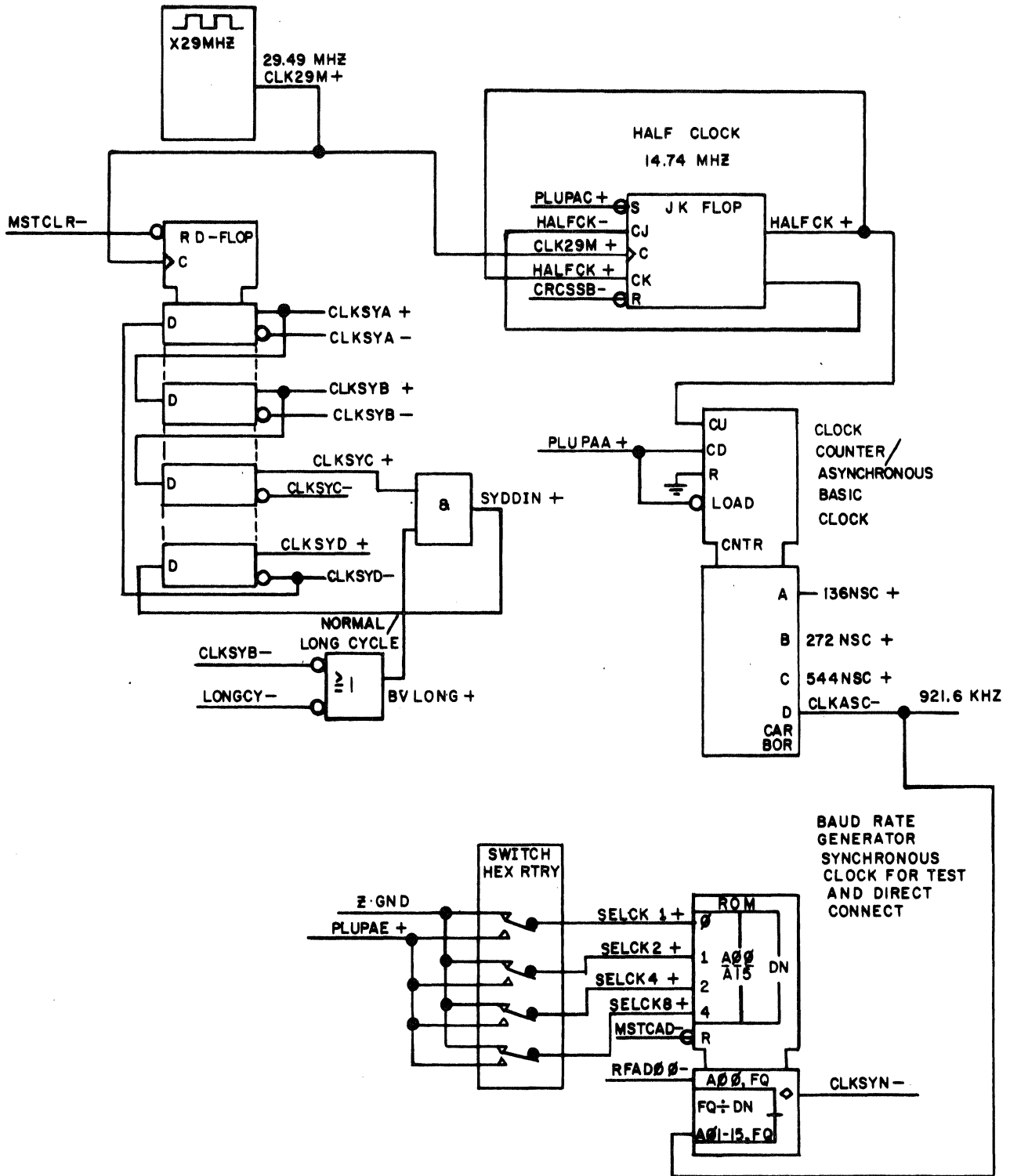
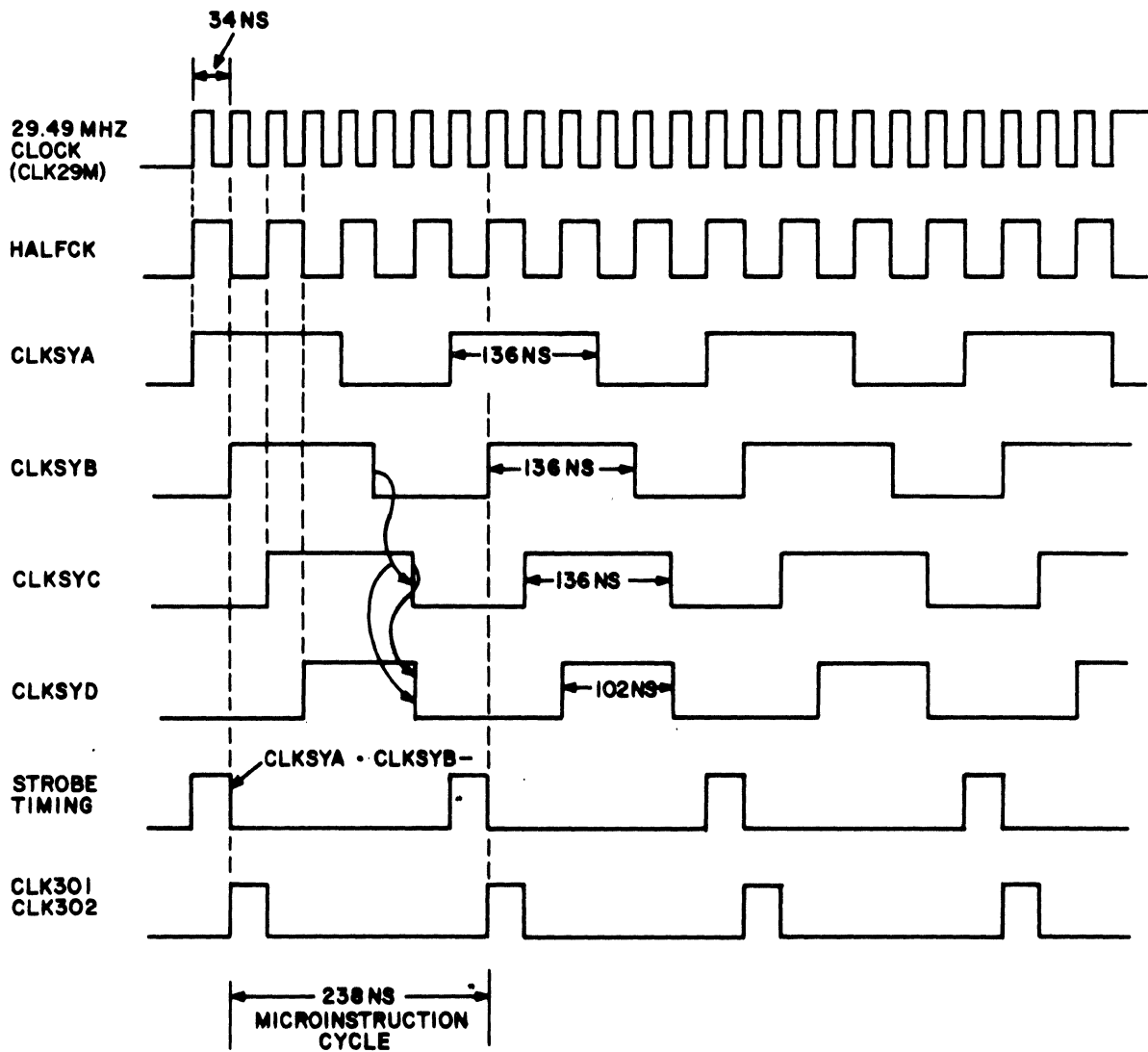
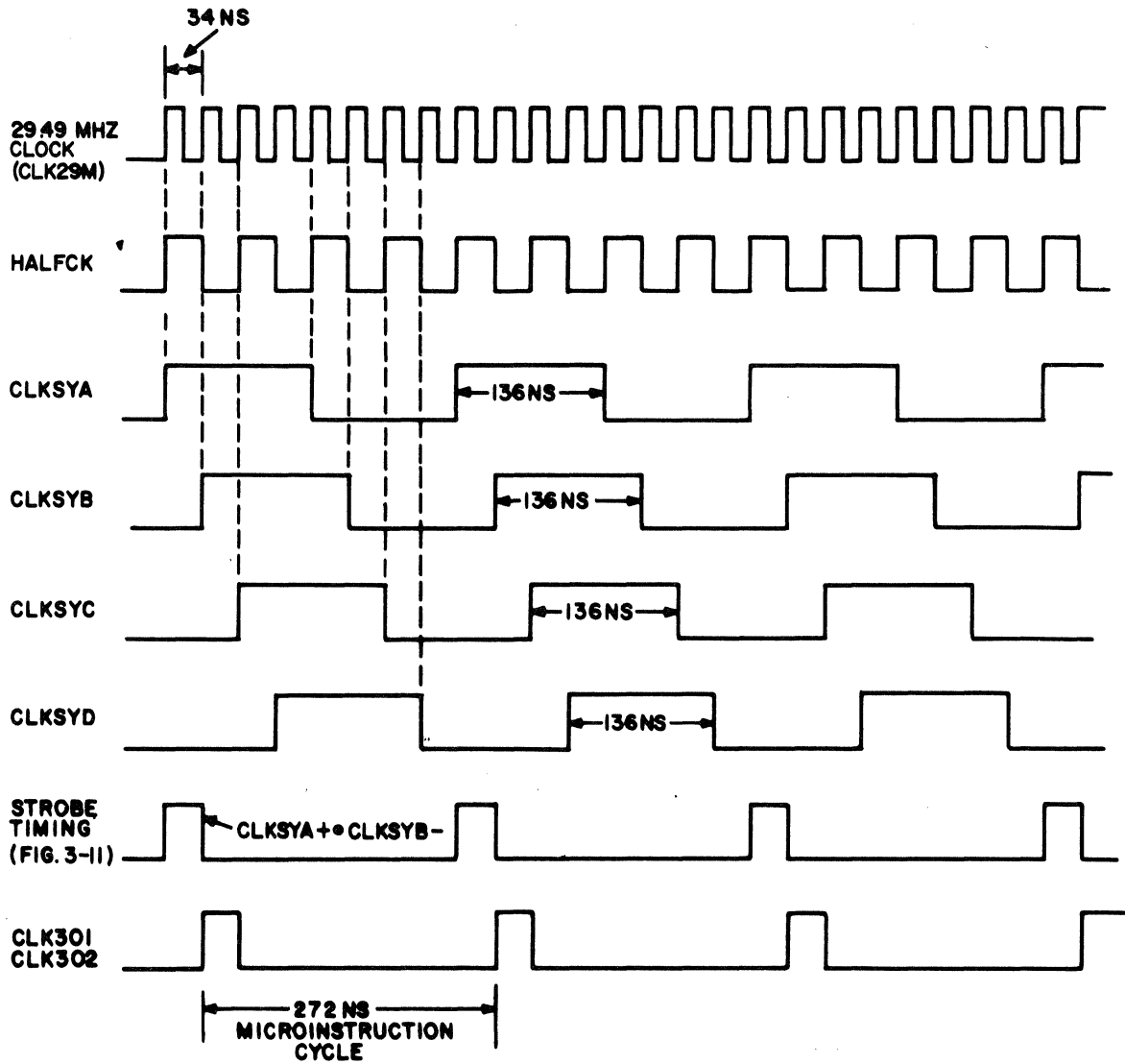


Figure 3-32 System Clock (Sheet 2 of 2)



A. NORMAL OPERATION

Figure 3-33 MLCP Timing Diagram (Sheet 1 of 2)



B. LONG CYCLE OPERATION (FOR TEST PURPOSES)

Figure 3-33 MLCP Timing Diagram (Sheet 2 of 2)

Table 3-19 Synchronous Clock Line Frequencies

HEX ROTARY SWITCH SETTING FIGURE 3-32B	JUMPERS 1 = OUT 0 = IN FIGURE 3-32A				DIVISOR	OUTPUT FREQ kHz	OUTPUT BAUD RATE
	N16W	N16U	N16S	N15Y			
1	0	0	0	1	1152	0.800	800
2	0	0	1	0	768	1.200	1,200
3	0	0	1	1	524	1.760	1,760
4	0	1	0	0	428.5	2.152	2,152
5	0	1	0	1	384	2.400	2,400
6	0	1	1	0	192	4.800	4,800
7	0	1	1	1	96	9.600	9,600
8	1	0	0	0	64	14.400	14,400*
9	1	0	0	1	48	19.200	19,200*
10	1	0	1	0	32	28.800	28,800*
11	1	0	1	1	24	38.400	38,400*
12	1	1	0	0	16	57.600	57,600*
13	1	1	0	1	12	76.800	Not Used
14	1	1	1	0	8	115.200	Not Used
15	1	1	1	1	6	153.600	Not Used
0	0	0	0	0	-	EXT	EXT

*Current Mode Synchronous Adapter only.

3.6 LINE ADAPTER INTERFACE CONTROL LOGIC (Figure 3-34)

The line adapter interface control logic interfaces the MLCP to any of the attached line adapters. The major functions of this logic are:

1. Resolve conflicting requests from line adapters on a positional priority basis.
2. Multiplex data from the line adapter into the microprocessor.
3. Generate and check parity (where required) for each character sent to the line adapter or received from the line adapter.
4. Transmit data from the microprocessor to the line adapter.
5. Generate clock signals for use by asynchronous line adapters, for synchronous line adapters operating in the direct mode without a modem, and for test purposes.
6. Multiplex the ID for each attached channel (0-15) into the B-mux for transmission to the Megabus.
7. Multiplex the line adapter present lines (LAHERE) to identify attached communication lines (i.e., two channels/line).
8. Provide control/strobe signals which define the function to be performed by the line adapter.

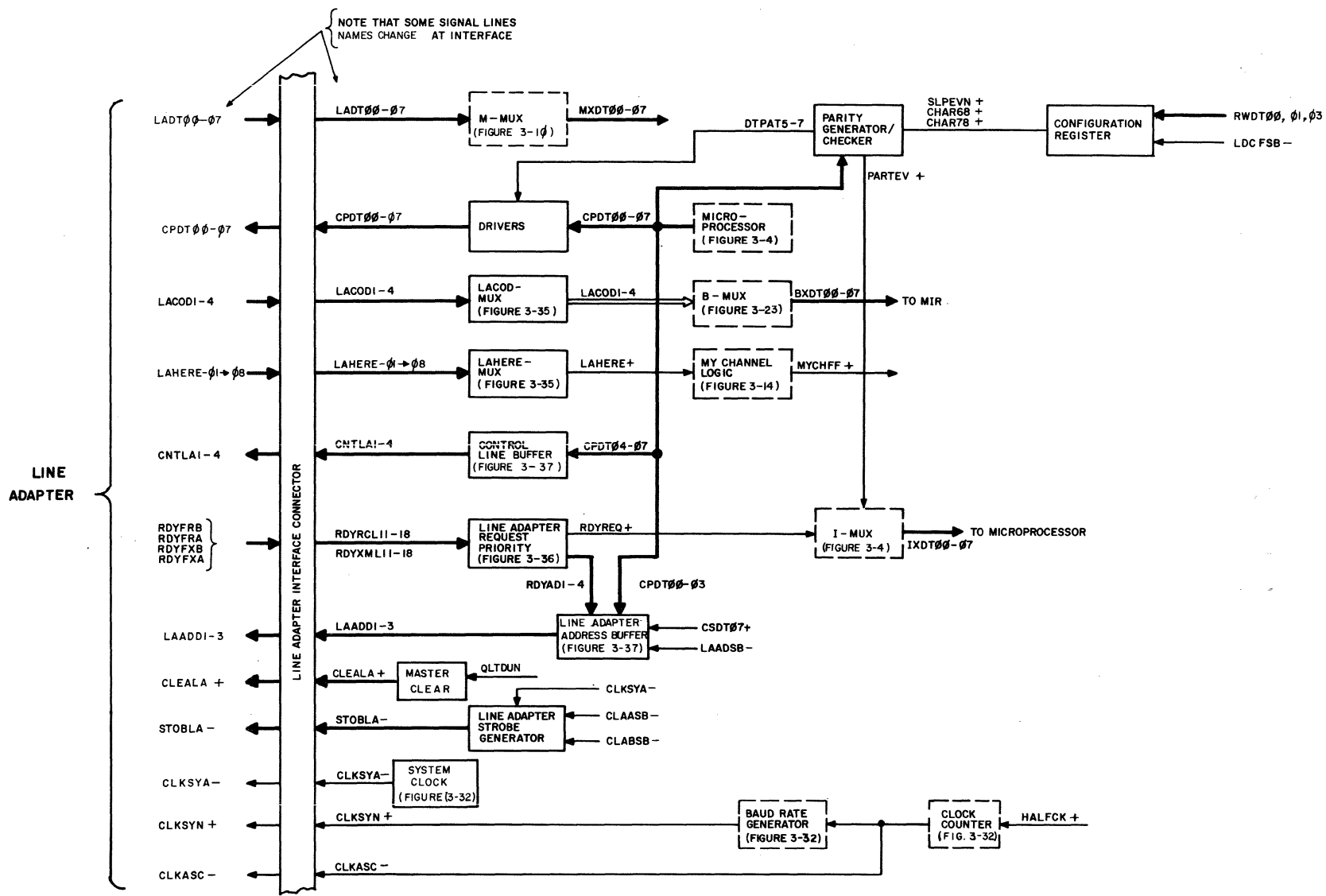


Figure 3-34 Line Adapter Interface Logic

3.6.1 Line Adapter Presence and Identification
(Figure 3-35)

The MLCP ascertains the presence or absence of a line adapter and attached communication channels (up to 16 channels), the type of each line, and the ID code for each line, by means of the Input ID command. One Input ID is required for each channel, transmit or receive, attached to each line adapter.

Implementation of the Input ID command requires the MLCP to identify that it is being addressed, and load the information on the Megabus address and data lines into the MIR, analyze the function code to determine the instruction to execute, and then load the desired information into the MIR and transfer it to the Megabus. Address comparison, function code translation, and MIR loading from the Megabus are described in subsection 3.4.2.

Since this is a hardware-implemented I/O command, the MLCP operation is controlled by the output of the function code translator (refer to subsection 3.3.3 for a detailed description of how the function code translator works). Assuming that the MIR has been loaded from the Megabus, and that the function code translator has been addressed by Megabus address bits BSAD18 through BSAD23, the content of the MIR is as shown by A in Figure 3-35. Starting with a function code translator address of (decimal) 48, the command executes in sequence the steps shown in Table 3-20 and described as follows:

1. Function Code Translator at Decimal Address 48
 - a. Scratch pad address counter bits two and three select the line adapter code lines (LACOD1 through LACOD4) from the line adapters through the LACOD mux as shown in Figure 3-35. There are four code lines from each line adapter. These lines are multiplexed to the input of the B-mux to provide the ID code 2lxx (where xx is CLA-specific).
 - b. PRDTB3 shifts the MIR row to row.
 - c. PRDTB1 and PRDTB2 select the A₁ input of the B-mux, and LACOD1 through LACOD4 are sent to the shift (serial) input of the MIR and loaded into MIR row 5.
 - d. The content of the MIR is now as shown by B in Figure 3-35.
 - e. The function code translator is incremented to decimal address 49 by the function code counter.
2. Function Code Translator at Decimal Address 49
 - a. Megabus address bits 14, 15, and 16 select the line adapter present lines (LAHERE01-08) through the LAHERE mux, and provide the fixed MLCP ID code.
 - b. The LAHERE lines go through a pull-up resistor network and become PLUPAA and PLUPAC, which appear at the input to the B-mux, and provide the fixed MLCP ID code of 2lxx.

- c. PRDTB3 shifts the MIR row to row.
- d. PRDTB1 and PRDTB2 select the A₂ input of the B-mux, and PLUPAA and PLUPAC are sent to the shift (serial) input of the MIR and loaded into MIR row 5.
- e. The content of the MIR is now as shown by C in Figure 3-35. This is the data that is sent to the requesting device.
- f. The function code translator is incremented to decimal address 50 by the function code counter.

3. Function Code Translator at Decimal Address 50

PRDTB7 resets the Megabus busy logic, and the content of MIR is sent to the Megabus.

If no line adapter is present for an addressed line, the MLCP does not respond for that instruction and a bus time-out occurs. The program is taken to a trip location from which it can determine that there was no response from the MLCP for that channel number.

Once the program knows the number of lines, it may start a CCP to service one of the lines. The CCP is resident in the R/W RAM memory.

Table 3-20 Determination of Line Adapter Presence and ID Codes

INPUT ID COMMAND (FC-26)											
FUNCTION CODE TRANSLATOR DECIMAL ADDRESS	FUNCTION CODE TRANSLATOR (PRDTB0-7)							B-MUX SELECT LINES AND SELECTED INPUT		SHIFT MIR	
	7	6	5	4	3	2	1	0	SELA		SELB
48	0	0	0	0	1	1	0	0	Low (A1)	High LACOD 1-4	Yes
49	0	0	0	0	1	0	1	0	High (A2)	Low PLUPAA	Yes
50	1	0	0	0	0	0	0	0	Don't care		No

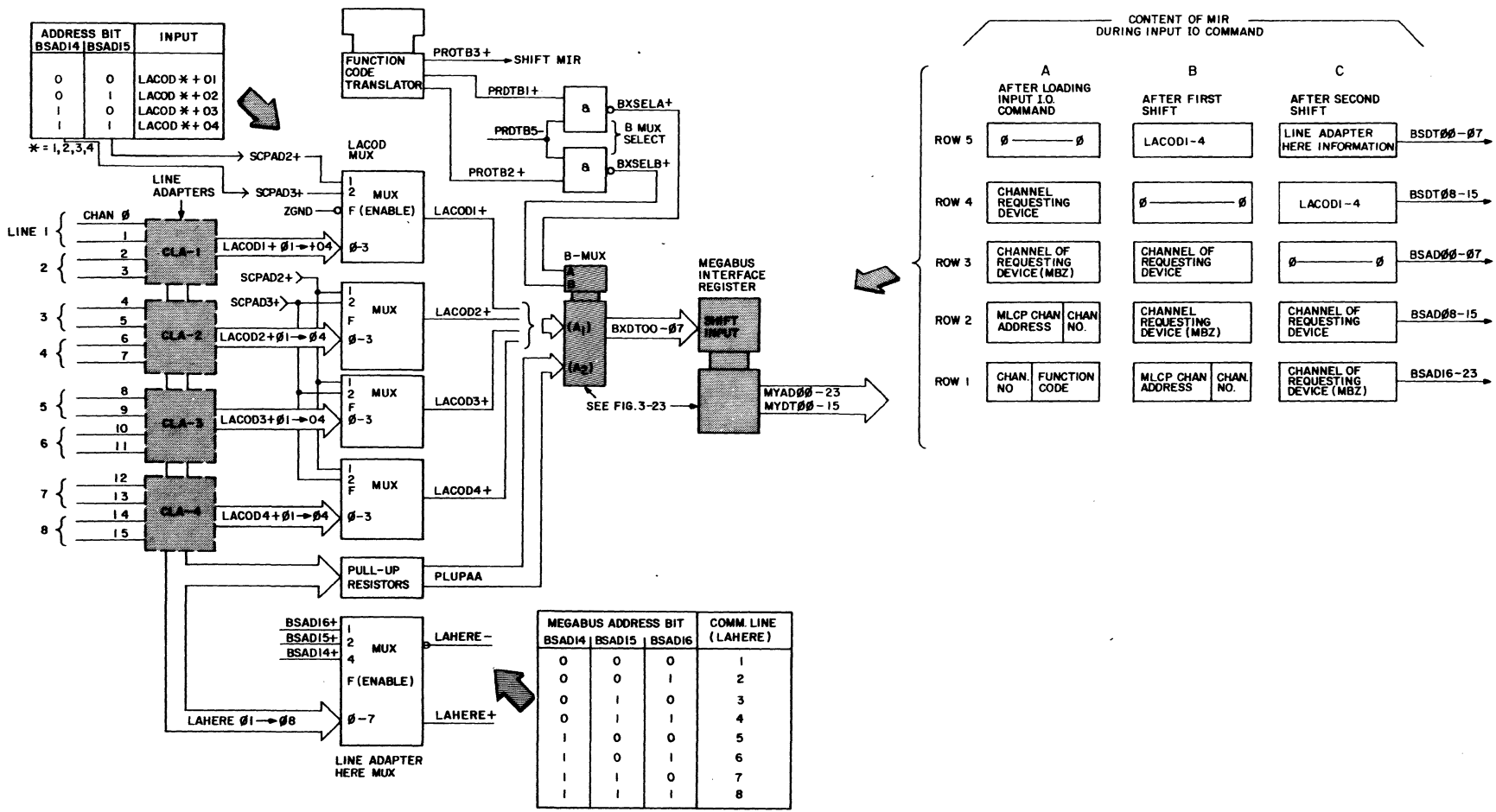


Figure 3-35 LACOD/LAHERE Multiplexer Selection

3.6.2 Service of Lines by Channel Control Program

A CCP may be activated in any of the following ways:

1. By a Start I/O command from the CPU.
2. By a status change in the CLA when channel scan mechanism is enabled.
3. By a data transfer request from a CLA channel.

In the first two cases firmware selects the channel by loading the channel address register LAADD1, 2, 4, 8 from CPDT04-07. In the third case the channel address is formed by the priority interrupt address generator.

3.6.2.1 Transmit Channels

3.6.2.1.1 Priority Resolution

This discussion assumes that the MLCP has been instructed via an I/O command to service a channel configured to transmit. Once activated, the firmware background routine checks the appropriate line adapter Ready lines to determine if the line adapter ready request flip-flop is set. The MLCP provides a clock to synchronize the line adapter ready request flip-flops. When a channel makes a request for service by raising a Ready (RDYFRA, RDYFRB, etc.) line, the flip-flop is set by this clock.

When the line adapter raises one of its Ready lines, the line is examined by the line adapter request priority logic (Figure 3-36) to determine whether or not that line adapter is to be granted service. If there are no conflicting requests of higher priority, the address of the requesting line adapter (RDYAD1 through RDYAD4) is loaded into the line adapter address multiplexer/buffer (LAADD1-3), and the Ready Request line (RDYREQ) is gated through the I-mux to the microprocessor.

If firmware detects that a line adapter has a request for service, it loads the start address of the appropriate CCP into the program counter (i.e., the R/W RAM address register), and the CCP assumes responsibility for servicing the channel.

Odd-numbered channels are always transmit channels, while even-numbered channels are always receive channels. The firmware program scans the line adapter ready request lines via the I-mux, determines when a channel has a request, and gets its address from the line adapter address multiplexer/buffer.

3.6.2.1.2 Addressing Line Adapter Registers

The channel control program must address and load the line adapter channel registers, reset the line adapter ready request flip-flop, as well as perform all data manipulation specified by the CCP.

Addressing of the line adapter registers is provided by the CCP via the M-mux and the microprocessor. (Refer to Figure 3-37.) Bits five through seven of the CCP instruction appear at the input

HONEYWELL PROPRIETARY AND CONFIDENTIAL

to the control line buffer on lines CPDT05 through CPDT07. This buffer is the source of the control lines which select the register (CNTLA1 through CNTLA4) for the line adapter interface. Bit four of the CCP instruction is also stored in this buffer, and is used to reset the ready request flip-flop. Table 3-21 shows the functions of the control lines within the line adapter.

3.6.2.1.3 Channel Selection

Channel selection is accomplished by bits zero through three of the CCP instruction. These bits also go through the M-mux and the microprocessor, and appear at the input to the CLA address multiplexer. These bits are selected onto the address lines by control store bit seven with the line adapter strobe (LAADSB-). Channel selection is made in a two-stage decoding of the four address lines (LAADD1 through LAADD4). The two most significant bits are decoded in the MLCP to select one of the four line adapters, while the remaining two bits are decoded in the line adapter to select one of four channels. Table 3-21 shows the functions of these four bits.

3.6.2.1.4 Special Parameters

Firmware next branches to the appropriate Line Control Table (LCT) in the R/W RAM to obtain special parameters, such as character size, whether odd or even parity is to be used, etc. This information is then loaded into the configuration buffer (Figure 3-34) via lines RWDT00 through RWDT03 under firmware control.

3.6.2.1.5 Parity Generation

The content of the configuration buffer controls operation of the parity generator/checker. Data to be sent to the line adapter from the microprocessor appears simultaneously at both the line drivers and the Parity Generator/Checker (PGC). The PGC, under control of the configuration buffer, determines the bit position in which to place the parity bit (according to the size of the character) and generates the necessary odd or even parity bit for that character. The PGC output (DTPAT5 through DTPAT7) overrides the bit configuration appearing at the drivers from the microprocessor (i.e., driver positions zero through two have ORed inputs), thereby forcing the generated parity bit into the correct position prior to transmission of the character to the line adapter.

3.6.2.1.6 Data Transmission

When it determines that all necessary processing has been accomplished, the firmware program causes the line adapter strobe generator to raise strobe line STOBLA, indicating that the data byte (DTLA00 through DTLA07) is valid on the line adapter interface.

3.6.2.2 Receive Channels

The line adapter interface logic functions in much the same manner for requests by lines configured to receive as it does for lines configured to transmit. Once the requesting line is given priority, the line address is loaded into the line adapter address

multiplexer/buffer (LAADD1-3). Firmware then determines the type of line by examining the line adapter address code lines (LACOD1 through LADOD4) and transfers the configuration information from the appropriate line control table in the R/W RAM to the configuration buffer.

Data from the line adapter (interface lines LADT00 through LADT07) is gated through the M-mux to the microprocessor, and from there to the parity generator/checker, where the parity bit is checked. If parity is not correct, the Parity Error (PARTEV) signal is generated and gated through the I-mux to the microprocessor. Firmware checks this I-mux bit for possible parity errors.

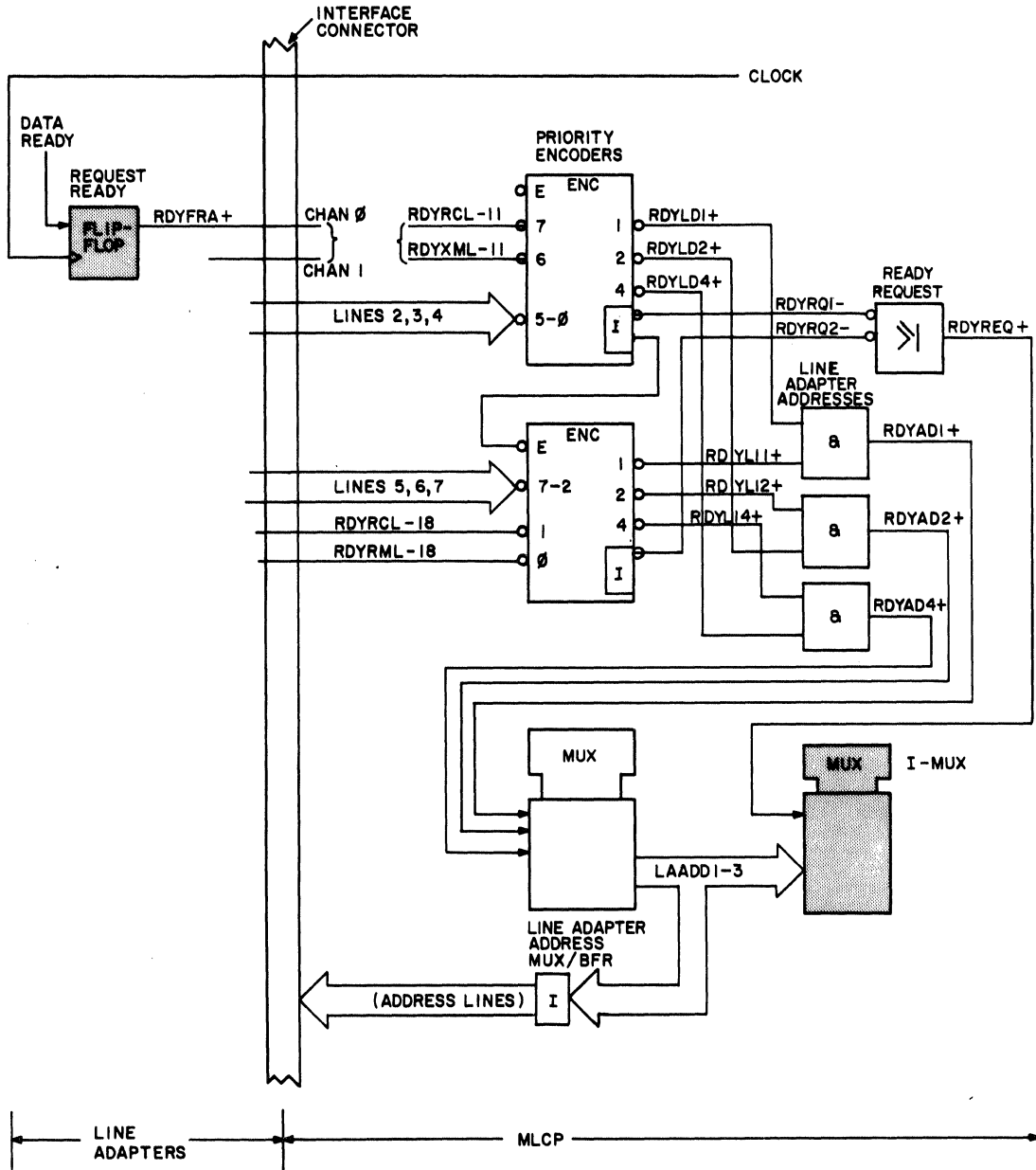


Figure 3-36 Line Adapter Request Priority Logic

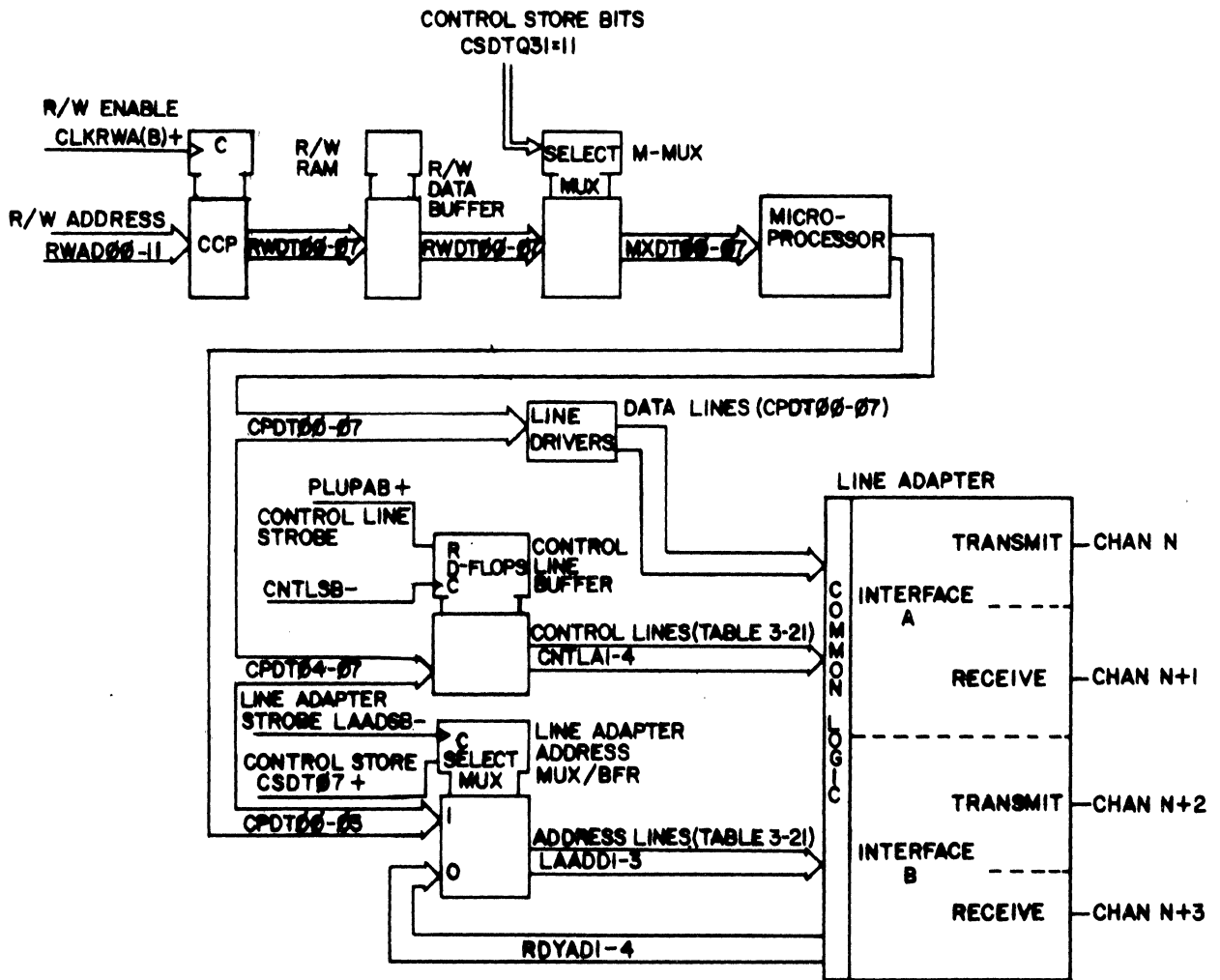


Figure 3-37 Addressing the Line Adapter

Table 3-21 Address and Control Line Functionality
at the MLCP/Line Adapter Interface
(for Dual Synchronous and Dual Asynchronous CLA's)

CONTROL LINE (CNTLA1-4)				FUNCTION
1	2	3	4	
1	0	0	X	Load sync (SCLA)/clock speed (ACLA) register
0	1	0	X	Load data set register
1	1	0	X	Load receiver/transmitter with configuration information
0	0	1	X	Output/input data to/from transmitter or receiver
1	0	1	X	Input status
1	0	1	1	Reset data request ready flip-flop
ADDRESS LINE (LAADD1-3)			FUNCTION	
1	2	3		
1	X	1	Select transmitter	
0	X	1	Select receiver	
X	1	1	Select interface B	
X	0	1	Select interface A	
X	X	1	Indicates to the line adapter at this position that it has been selected by the MLCP. The MLCP can control up to four line adapters, but enables only one at a time with LAADD3.	
X	X	0	Indicates to the line adapter at this position that it has not been selected by the MLCP. When LAADD3 is Zero, no address line decoding is performed by the line adapter.	

Note 1: X = decoding logic does not care about the status of this bit when determining indicated control function.

Note 2: Except for Registers 1 (data) and 5 (status) register function is Line Adapter specific.

3.6.3 Clock Counter/Baud Rate Generator

For asynchronous lines the line adapter interface logic contains a clock counter (CLKASC) which is driven by an output of the system clock (HALFCK) to generate a 921.6 kHz clock signal for use by attached asynchronous line adapters. This clock is further modified (stepped down) by the baud rate generator to generate the Clock Sync (CLKSYN) line, which is used during testing (where transmit data is looped back from the line adapter when it is being tested), or with synchronous line adapters for direct mode operation when no modem is employed. The frequency of the Clock Sync line is selectable by a hex rotary switch at the input to the baud rate generator. These clock signals are described in subsection 3.5.4 and illustrated in Figure 3-32.

3.7 CRC LOGIC (Figure 3-38)

Cyclical Redundancy Checking (CRC) is a form of error checking particularly suited to the burst error conditions typically encountered on communications channels. The basic CRC mechanism treats the transmitted message as a polynomial which is divided by the check polynomial (e.g., CRC-16), with the resulting remainder of this division process being appended to the end of the message. The receiver divides the incoming message by the same check polynomial. Since the check character makes the message evenly divisible by the polynomial, the remainder at the receiver should be zero. If it is not zero, an error is indicated.

The CCITT variation of the basic mechanism overcomes certain deficiencies in the detection of erroneous leading and trailing zeros in the received message.

Because the CRC hardware is shared by all MLCP channels, the process described above is performed on a byte-by-byte basis for each channel. It is, therefore, necessary to store partial CRC remainders (residue) in each channel's Line Control Table (LCT) between bytes.

The CRC hardware (Figure 3-38) implements any one of four error detection codes. One code, LRC-8, utilizes Longitudinal Redundancy Checking (LRC), and three codes, CRC-12, CRC-16 and CRC-CCITT, utilize CRC. The generator polynomials for these codes are:

$$\begin{aligned} \text{LRC-8} &= 1 + X^8 \\ \text{CRC-12} &= 1 + X + X^2 + X^3 + X^{11} + X^{12} \\ \text{CRC-16} &= 1 + X^2 + X^{15} + X^{16} \\ \text{CRC-CCITT} &= 1 + X^5 + X^{12} + X^{16} \end{aligned}$$

These polynomials are a definition of the configuration of the 16 bit CRC shift register shown in Figure 3-38. This configuration is program selectable by each Channel Control Program (CCP). The CRC hardware, shared by all of the CCPs, includes the following functional areas:

1. 16-bit CRC shift register (CRCR01 through CRCR16)
2. Exclusive-OR multiplexer

3. Eight-bit data register (DTCHAR)
4. CRC multiplexer (CRDT00 through CRDT07)
5. Configuration register/clock

Each CCP configures and then loads and saves the CRC residue for each line. The CCP loads the configuration register from the LCT for the channel (location 0 for receive channel or location 32 for transmit channel). Selection of character size and CRC polynomial is made by certain configuration register bits as follows:

CHAR68	CHAR78	CHARACTER SIZE	SLCRC0	SLCRC1	POLYNOMIAL
0	0	5 bits	0	0	CRC-16
1	0	6 bits	0	1	CRC-CCITT
0	1	7 bits	1	0	CRC-12
1	1	8 bits	1	1	LRC-8

The CCP next loads the CRC shift register with the residue CRC information from the LCT (locations 3 and 4 for receive channels or locations 35 and 36 for transmit channels). If this is the first character to be accumulated, the CCP will zero out the CRC location in the LCT (or set that location to all Ones if a CCITT) before loading the CRC shift register. The configuration register and the CRC shift register are loaded directly from the R/W RAM (RWDT00 through RWDT07).

The data character which is to be accumulated is stored in the microprocessor. When the CCP issues the command to accumulate CRC the microprocessor output (CPDT00 through CPDT07) is loaded into the data register. The output of this register, DTCHAR, is a serial input to the CRC shift register. The data character is shifted into the CRC shift register one bit at a time by the shift clock, CLKCRS. The number of shifts is determined by the character size configuration. As shown in Figure 3-39, the character size is loaded from the R/W RAM (bits 0 and 1) into the character size shift counter (CRCNTX). Each shift clock (CRCCLK) decrements this counter until it reaches zero, at which time CRCNTB disables the shift clock and terminates accumulation for that character.

The inputs to the several CRC shift register positions are wired from the exclusive-OR multiplexer through which the data character must pass. This exclusive-OR multiplexer determines the CRC.

The CCP loads the developed check character into the microprocessor via the CRC-mux (CRDT00 through CRDT07) and the M-mux. The CCP may check the CRC character after each accumulation. The CRC character must be checked at the end of a message for receive channels to determine if the message had any errors.

Figures 3-40 through 3-43 contain examples of check character accumulation for each of the four polynomials.

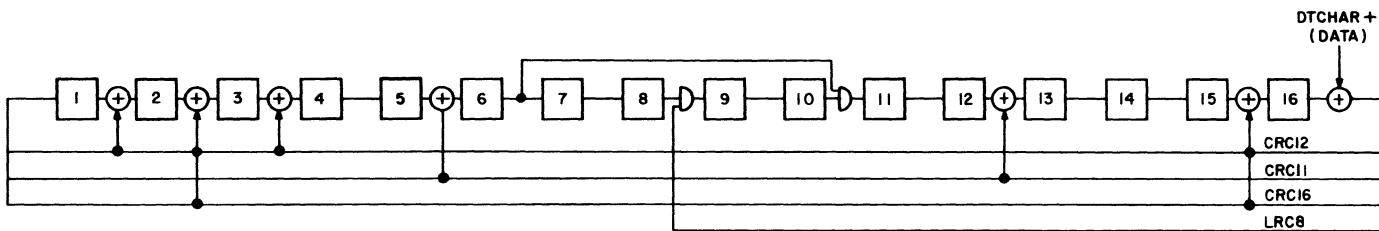
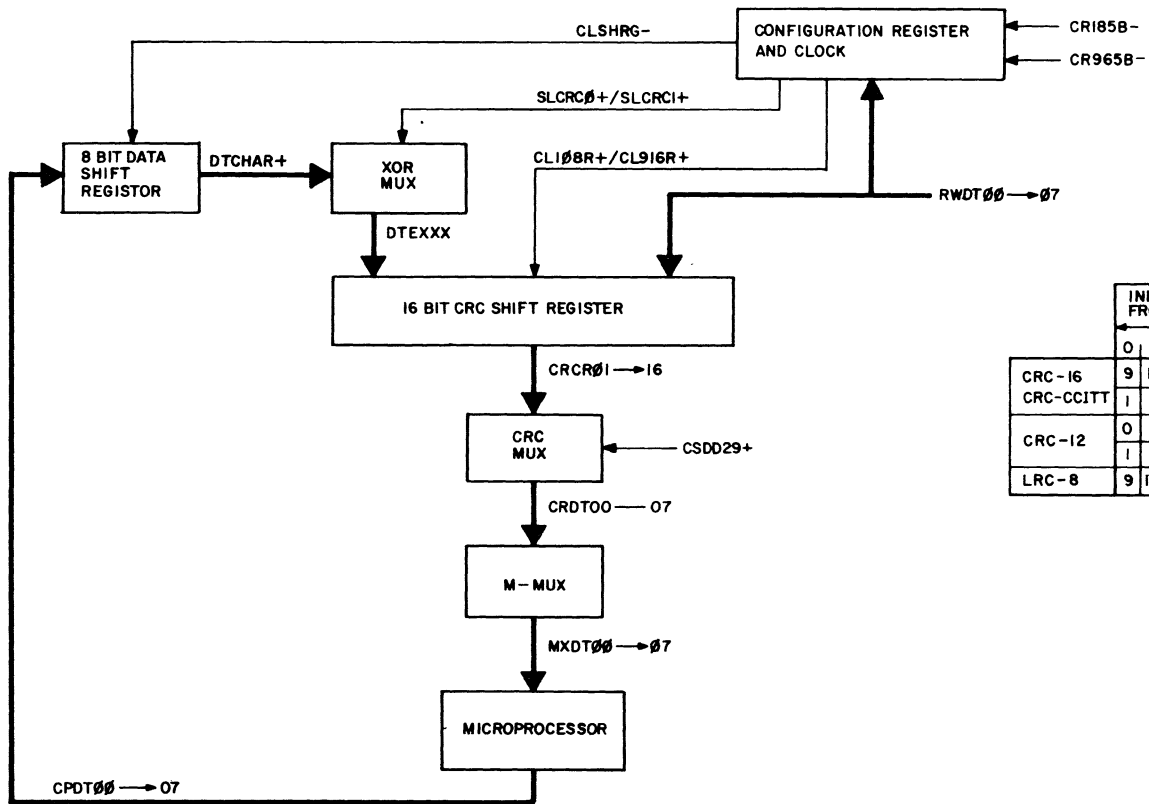


Figure 3-38 CRC Logic

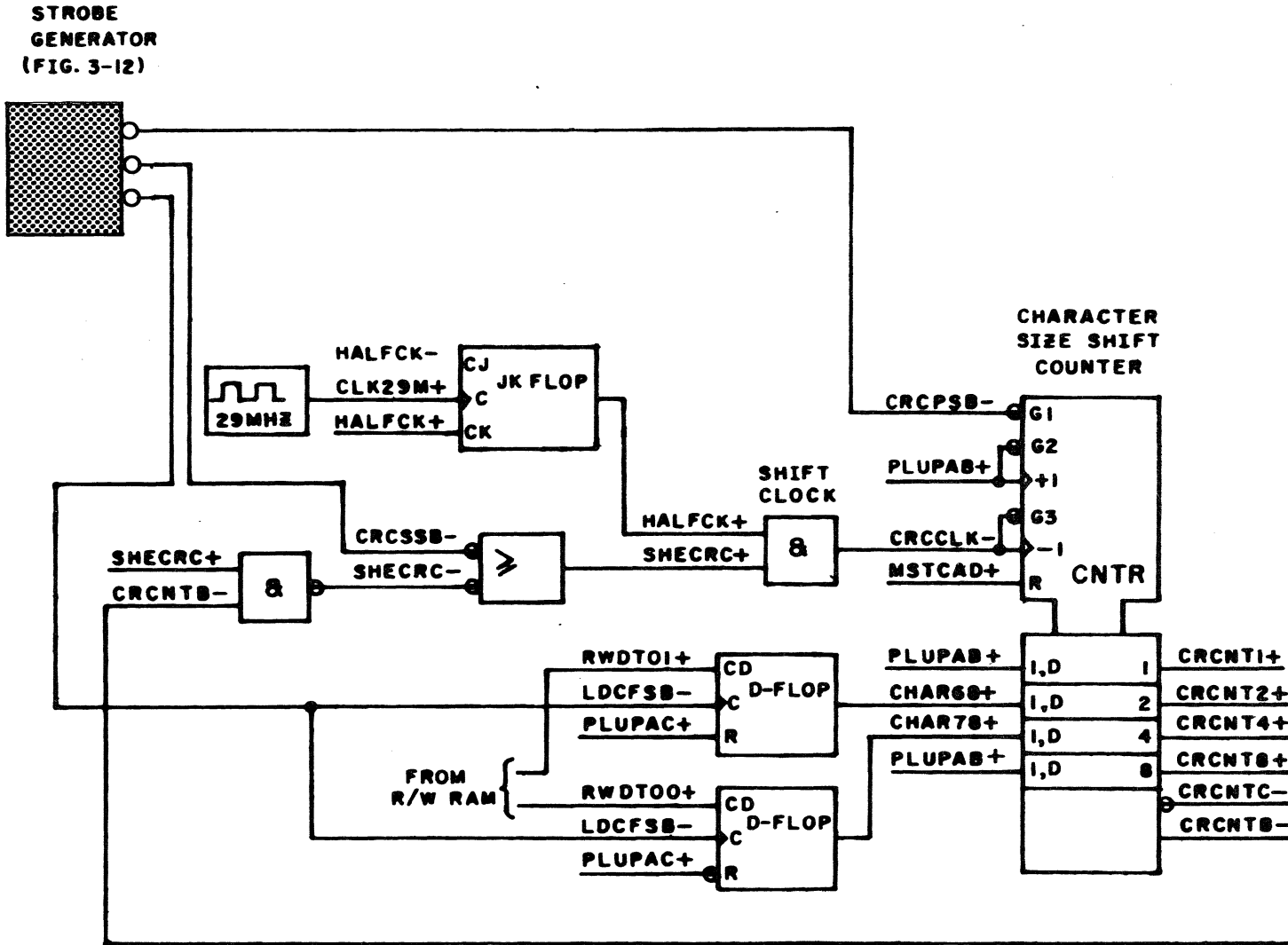


Figure 3-39 Character Size Logic

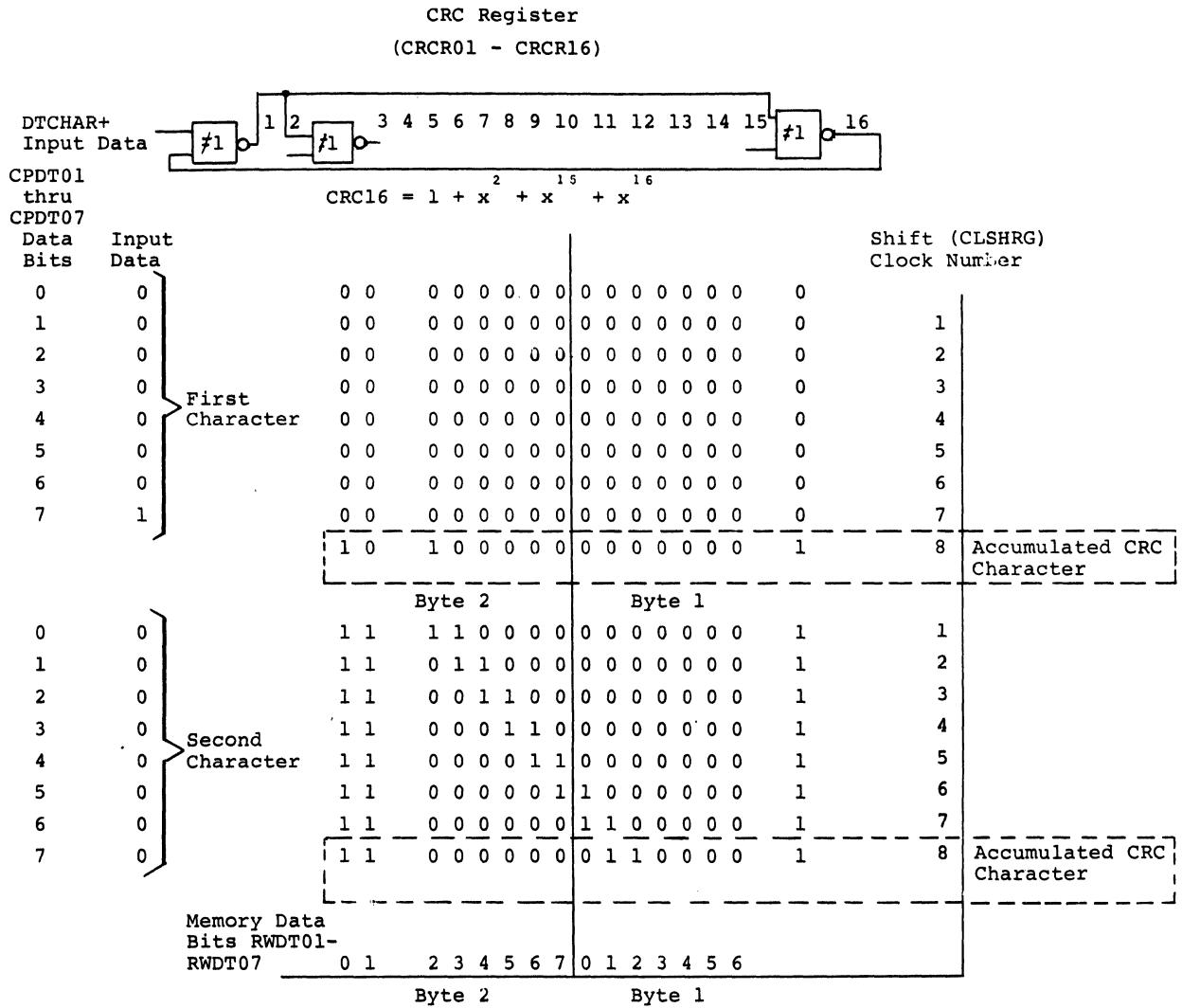


Figure 3-40 CRC-16 Accumulation

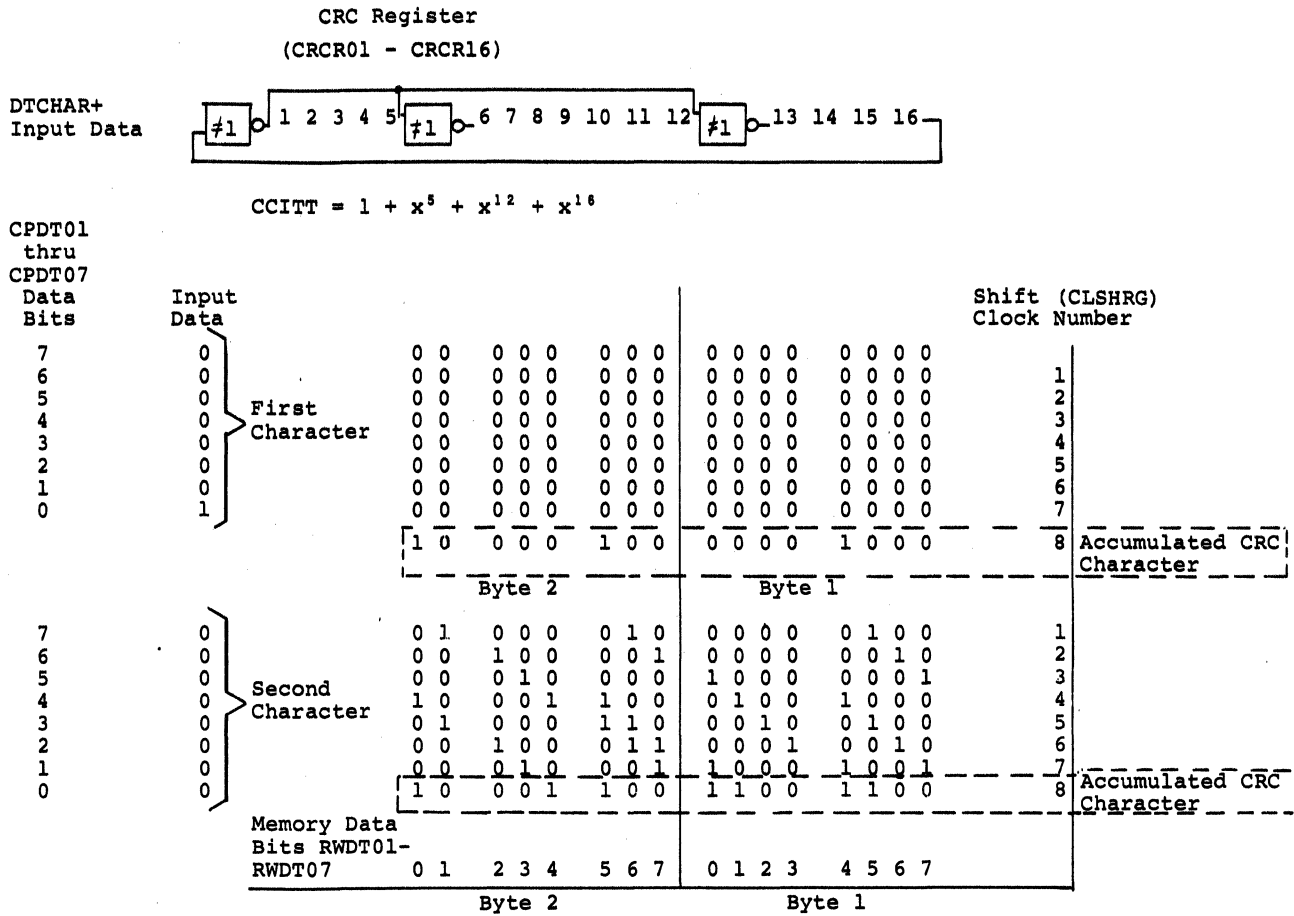


Figure 3-41 CCITT Accumulation

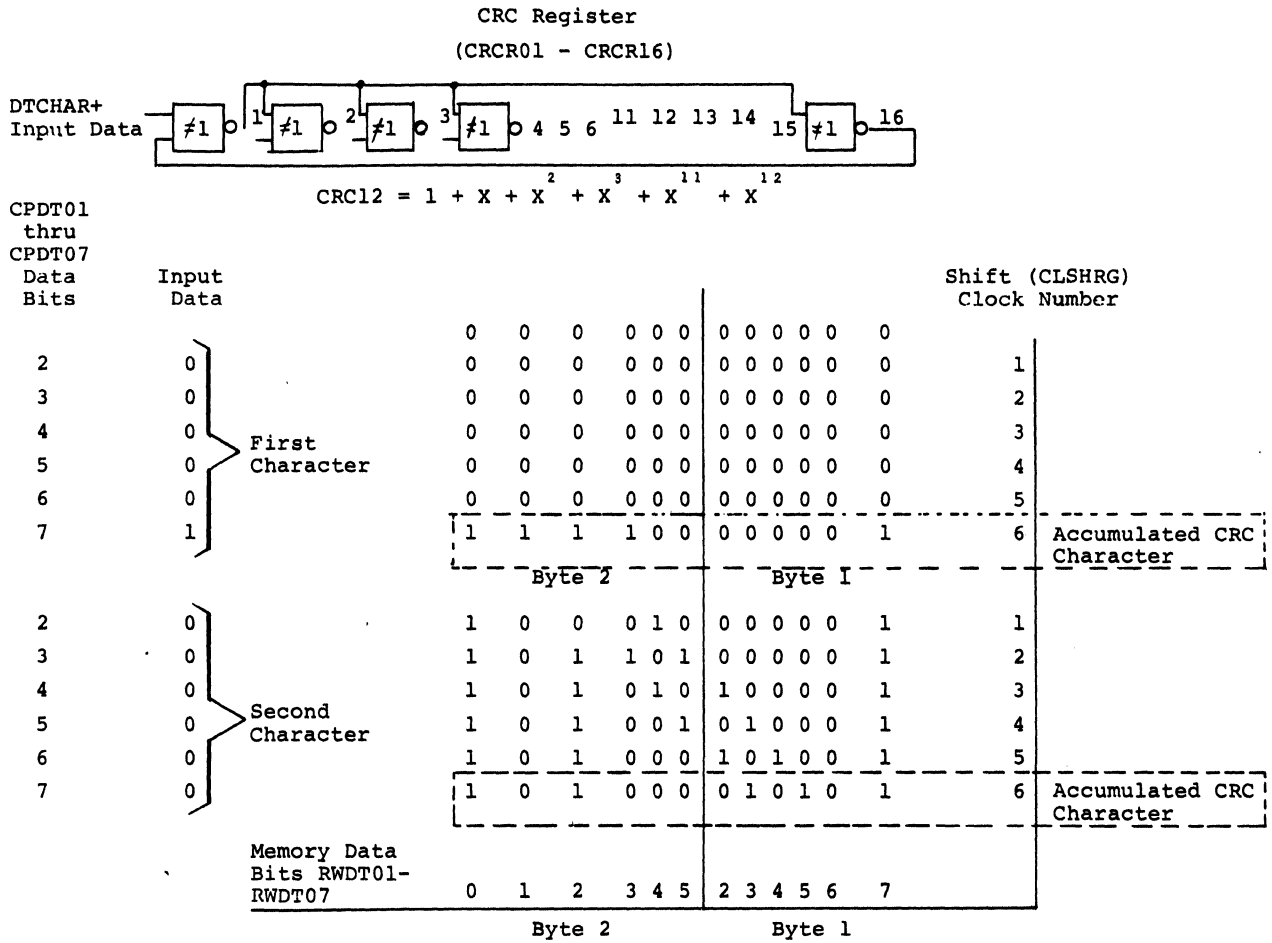


Figure 3-42 CRC-12 Accumulation

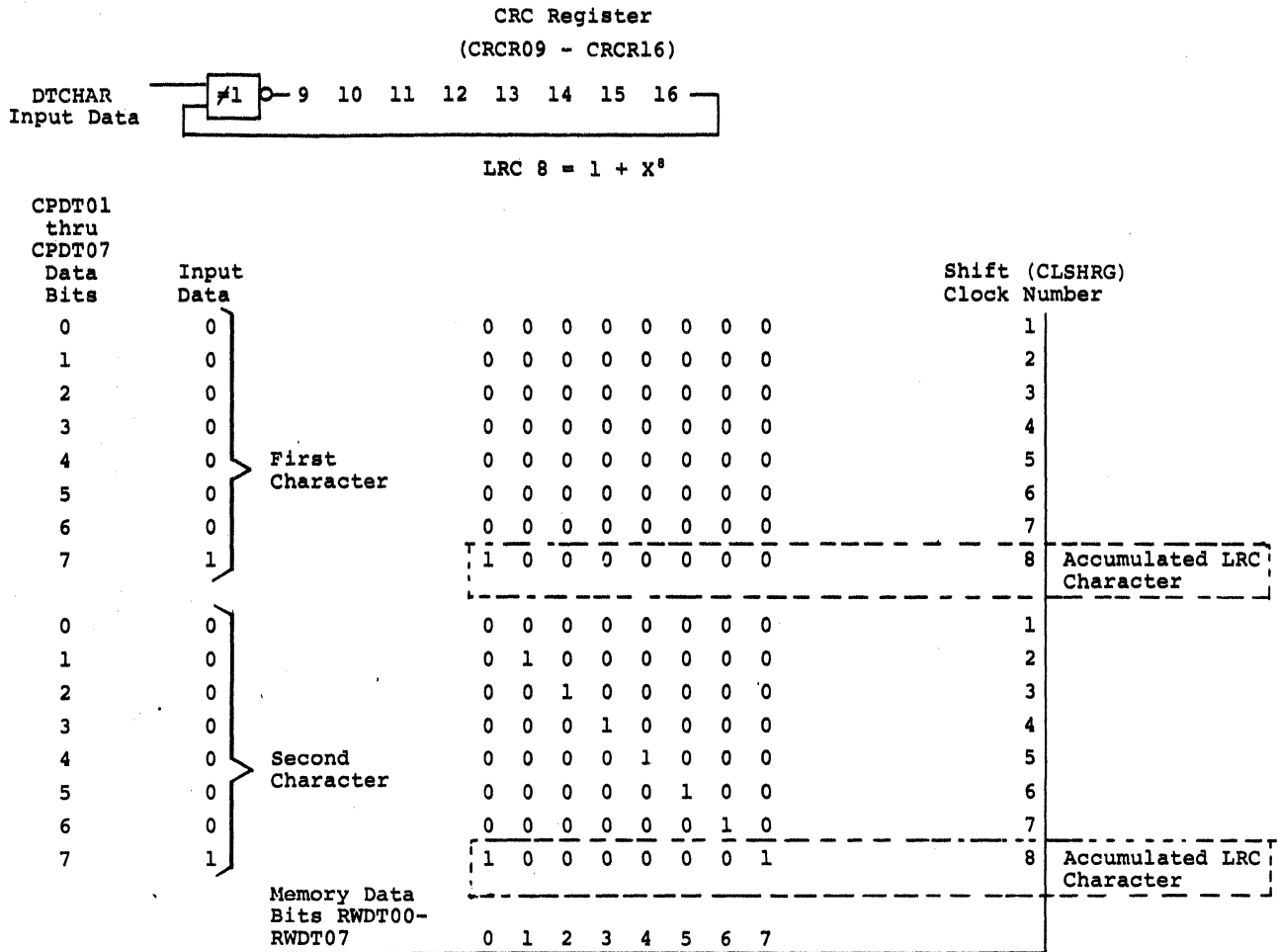


Figure 3-43 LRC-8 Accumulation

IV THEORY OF OPERATION - CYCLE FLOW

4.1 MLCP FIRMWARE

The MLCP firmware* is comprised of routines which are formed from various types of microinstructions (refer to subsection 4.2) resident within the MLCP read-only storage memory. The storage has the capability of housing up to 1.5K of locations with each location containing a 32-bit encoding of microinstructions referred to as micro-ops/firmware commands. When read from the read-only storage and decoded, a firmware command results in the specific action of various hardware elements.

A set sequence of hardware operations can be obtained by performing designated serial execution of firmware commands. These groupings of firmware commands are referred to as firmware routines. The firmware routines provide an operational link between the Level 6 software and the MLCP subsystem controller. Software commands are executed by firmware decoding of the software command. This causes the MLCP to exit from the scanning process and select the proper firmware routine. The sequencing of firmware commands continues until all the routines required to complete the software command have been executed.

In addition to I/O-initiated software commands, the MLCP firmware supports an instruction set located in its random access memory. These instructions, loaded initially by the central system software, are assembled in lists called Channel Control Programs

*This manual reflects Firmware Release Level 11.0.

HONEYWELL PROPRIETARY AND CONFIDENTIAL

(CCP) (refer to subsection 4.4.2). The instructions within the CCP, decoded and executed under direct control of the firmware, are required for the efficient handling of a data stream.

The random access memory is used not only for storage of CCPs but also to store Line Control Tables (LCT) and Communication Control Blocks (CCB) (refer to subsections 4.4.1 and 4.4.3). Each of these areas is capable of being interrogated or manipulated by the firmware for the purpose of interpreting or updating status, configuration, control, etc., information.

Several scratch pad locations/registers (refer to subsection 4.3) are also available to the firmware for application as work registers, address register backups, counters, and alterable storage of flag control information.

4.2 FIRMWARE INSTRUCTIONS

Firmware instructions are made up of bit structures known as microinstructions and are located in a 1.5K by 32-bit deep storage area referred to as the Read-Only Storage (ROS) memory. The 32 bits of each command are broken into six fields designated as:

1. CPE Function/Register Code field (bits 0 through 6)
2. Multiplexer Select/Subcommand field (bits 7 and 27 through 31)
3. K-Bus field (bits 8 through 15)
4. MCU address control field (bits 16 through 22)
5. Flag "O" field (bits 23 and 24)
6. Flag "I" field (bits 25 and 26).

The description of each of these fields, as well as their hardware implementation and subcommand generation, is located in Section III of this manual.

4.3 SCRATCH PAD LOCATIONS/REGISTERS

The firmware is supplied with eleven eight-bit registers which are utilized as scratch pad areas. In some instances the application of these registers varies with the operation being performed. Table 4-1 contains a listing of the 11 registers with a description of the contents of each during normal operation. Figures 4-1 through 4-5 show the bit significance of the registers whose contents have a specific configuration.

4.4 RANDOM ACCESS MEMORY

The MLCP contains a Random Access Memory (RAM) which is 4096 locations by eight bits deep. The content of this RAM is implemented to direct the actions of each channel attached to the MLCP.

The RAM is segmented into three basic areas: 512 locations are for line control table information, 3072 locations are provided to store channel control programs, and the remaining 512 locations are utilized for communication control block storage. Figure 4-6 is a representation of the relative positioning of the three major areas and their respective address locations.

HONEYWELL PROPRIETARY AND CONFIDENTIAL

Table 4-1 Register Application

REGISTER	DESCRIPTION
Register 0	1. Channel scan number counter. 2. Extention of register 1 for use as an indicator register (see Figure 4-1).
Register 1	Register 1 is utilized as an indicator register in block mode and during instruction execution (see Figures 4-2 and 4-3).
Register 2	Work Register
Register 3	Work Register
Register 4	Work Register
Register 5	Work Register for I/O instructions. Program visible register during execution of CCP. This register is referred to as the "R" register.
Register 6 and Register 7	Contain the channel number of the currently active channel (see Figure 4-4).
Register 8 and Register 9	Contain the RAM address of the next instruction to be executed within the current channel control program (see Figure 4-5). This register is referred to as the "P" counter.
T-Register	Work Register.

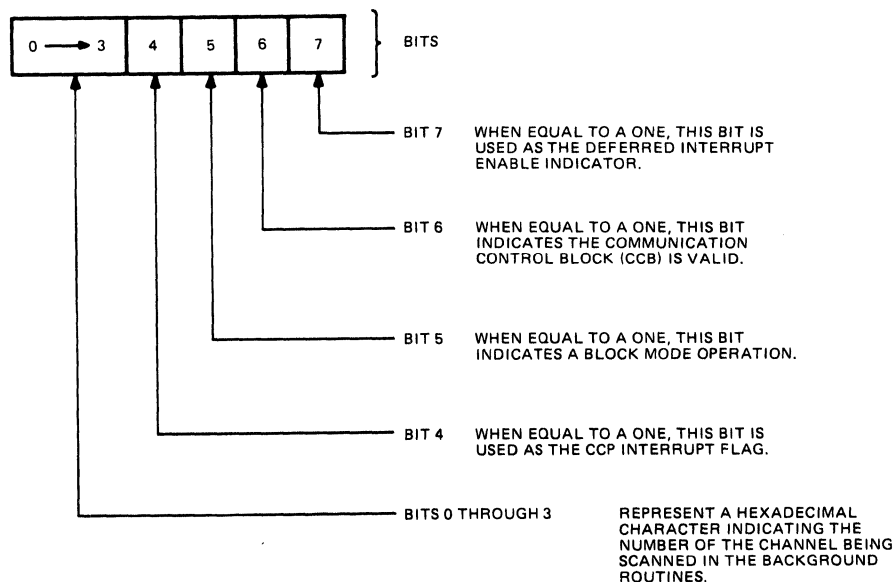


Figure 4-1 Register 0 Bit Significance (Copy of LCT 21/43)

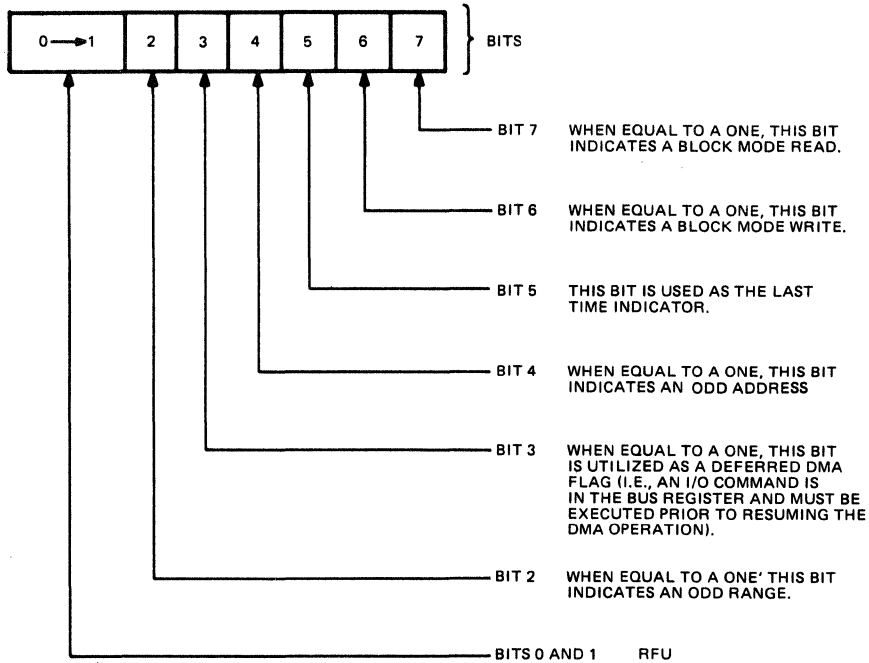


Figure 4-2 Register 1 Bit Significance in Block Mode
(Copy of LCT 5/37)

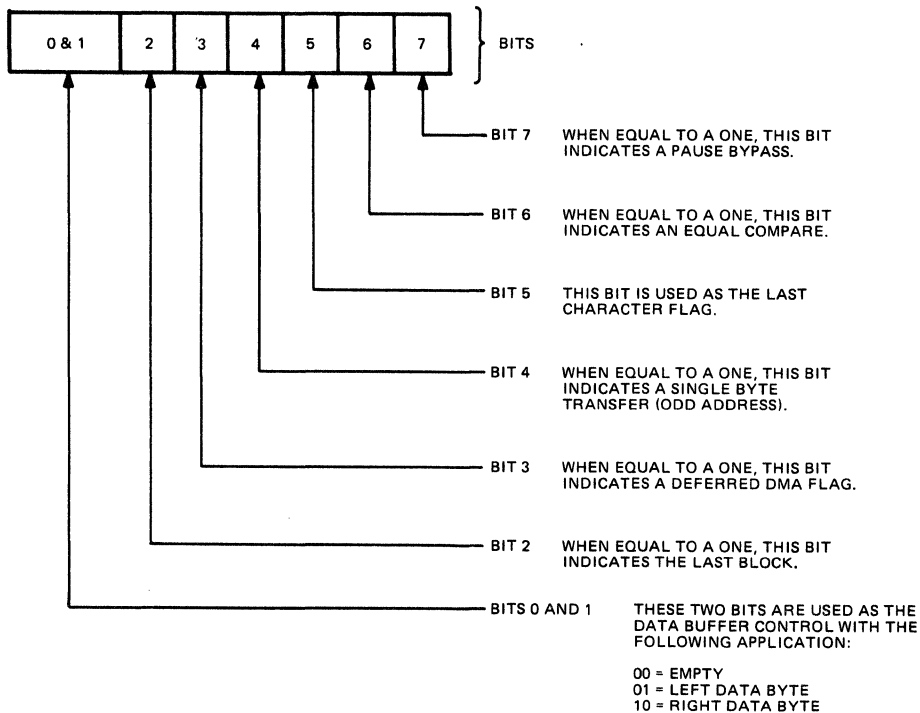


Figure 4-3 Register 1 Bit Significance During Instruction Execution

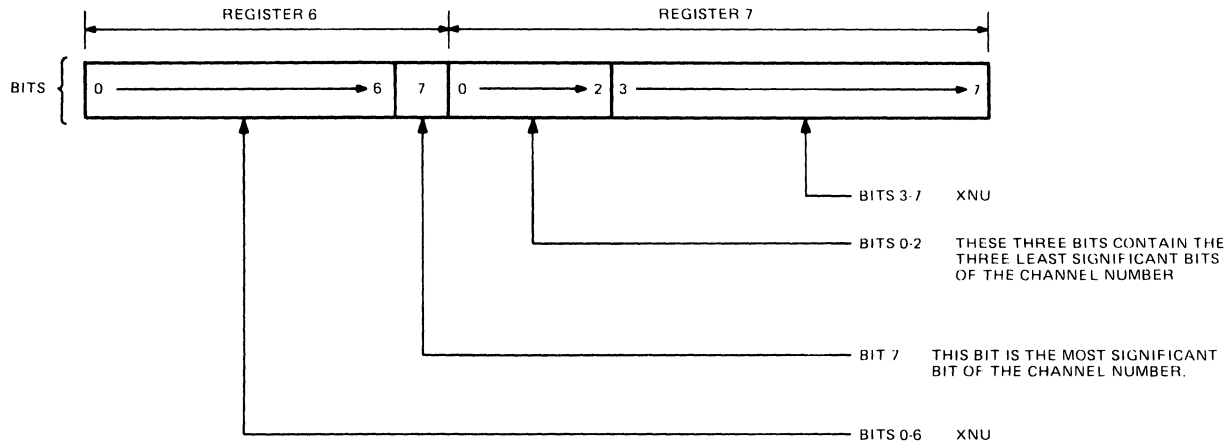


Figure 4-4 Channel Number Assignments Within Register 6 and Register 7

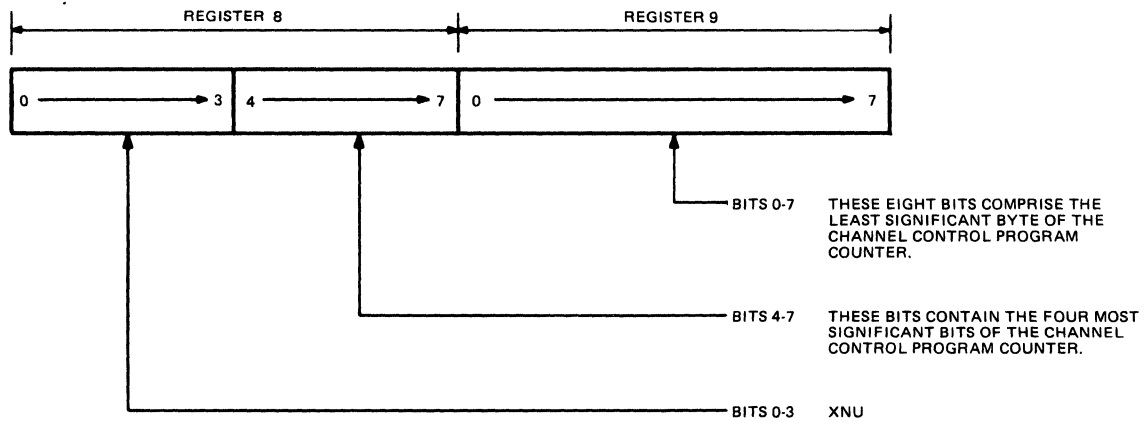


Figure 4-5 Channel Control Program Counter Arrangement Within Register 8 and Register 9

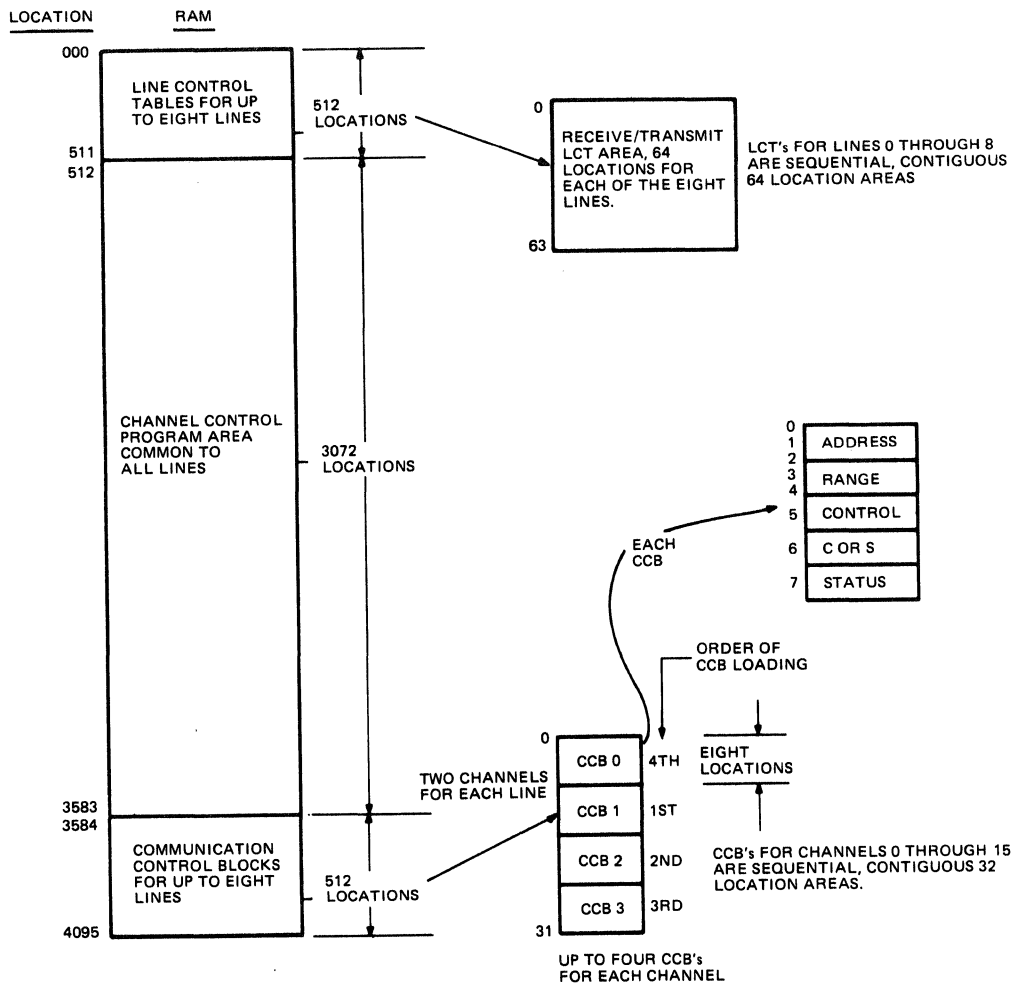


Figure 4-6 Random Access Memory Segmentation

4.4.1 Line Control Tables (LCT)

Each of the possible eight lines capable of being attached to the MLCP has its own unique and independent LCT located within the first 512 locations of the RAM. All the configuration, setup, status, and control information appears in the LCT during the operation of the MLCP.

The information located within the LCT is the prime element in coordinating the MLCP operation in conjunction with a specific line. This is evidenced by the fact that the LCT is visible to the CPU via I/O operation, to the MLCP through channel control programs, and to the MLCP firmware.

An LCT consists of a block of 64 contiguous bytes with 32 (0 through 31) bytes dedicated to the even (or receive) channel, and 32 (32 through 63) bytes designated for the odd (or transmit) channel. These paired channels (even and odd) comprise one line and are therefore allocated only one LCT.

HONEYWELL PROPRIETARY AND CONFIDENTIAL

An LCT is line dependent and as such the CCP associated with a particular line will not be allowed access to any other line's LCT.

The contents of each LCT must be preconditioned by a load, channel initialize, or a start/stop I/O sequence before the startup of any data transfer by a channel. The hard initialize function of the MLCP and the channel initialize provide known starting values for each LCT location (nominally all Zeros).

The LCT can be written by the CPU by way of the Output LCT Byte instruction or by the execution of a Block Mode Write. The contents of the LCT can be read by utilizing the Block Mode Read or an Input LCT Byte instruction. Table 4-2 lists the LCT address and byte content topology; Table 4-3 calls out each byte and gives a brief description of its usage. Where an LCT byte has a bit-significance purpose, the bit structure and use are shown in Figures 4-7 through 4-12.

Table 4-2 Line Control Table Topology

LCT BYTE ADDRESS		CONTENTS
RECEIVE	TRANSMIT	
00	32	Firmware work locations
01	33	Receive revision number of firmware/ firmware work location
02	34	Receive/transmit configuration
03*	35*	Receive/transmit CRC residue-byte 1
04*	36*	Receive/transmit CRC residue-byte 2
05	37	Receive/transmit program indicators
06*	38*	Receive/transmit CCP pointer-4 MSB
07*	39*	Receive/transmpt CCP pointer-8 LSB
08	40	Receive/transmit channel command
09	41	Receive/transmit channel control
10*	42*	Receive/transmit data-left byte
11*	43*	Receive/transmit data-right byte
12	44	Receive/transmit return channel number
13	45	Receive/transmit level
14	46	Receive/transmit CLA status
15*	47*	Receive/transmit CLA status mask
16*	48*	Receive/transmit status-byte 1
17	49	Receive/transmit status-byte 2
18*	50*	Receive/transmit CCP subroutine pointer-4 MSB
19*	51*	Receive/transmit CCP subroutine pointer-8 LSB
20*	52*	Receive/transmit CLA control
21	53	Firmware work locations
22	54	Firmware work locations
23*	55*	CCP work location/transmit address for Input LCT Byte Command
24*	56*	CCP work locations/CCP work locations ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓
31*	63*	CCP work locations/CCP work locations

*Indicates location which can be modified by the CCP.

HONEYWELL PROPRIETARY AND CONFIDENTIAL

Table 4-3 Line Control Table Byte Description
(Sheet 1 of 2)

LCT BYTES	DESCRIPTION
Receive Firmware Revision Number (LCT01)	This byte is loaded with the revision number of the resident firmware.
Receive/Transmit Configuration (LCT 2/34)	This byte is used to define the CLA configuration. The bit definitions are CLA specific and correspond to the definition of LR6 on the channels. Refer to the MLCP Programmer's Reference Manual for bit definitions.
Receive/Transmit Residue Bytes (LCT 3,35/4,36)	These bytes represent an intermediate block check result and are referenced by both MLCP hardware and CCP procedures.
Receive/Transmit Program Indicators (LCT 5/37)	This byte is used to store the CCP indicators; equal and last character in block.
Receive/Transmit CCP Pointer (LCT 6,38/7,39)	The channel CCP counter is used in the MLCP during instruction set processing and points to the initial location in the RAM for current CCP execution. When the channel is serviced, these bytes are loaded into registers 8 and 9 (P-register).
Receive/Transmit Channel Command (LCT 8/40)	This byte is used by the firmware as a reference for required action during channel operation on this line. See Figure 4-7 for specific bit designations.
Receive/Transmit Channel Control (LCT 9/41)	This byte is used by the MLCP firmware to control the operation of the channel on this line. See Figure 4-8 for specific bit application.
Receive/Transmit Data Bytes (LCT10,42/11,43)	These two bytes are the left and right data respectively to or from the Megabus data lines.
Receive/Transmit Return Channel Number (LCT 12/44)	This byte contains the eight least significant bits of the return channel number for use with interrupts.
Receive/Transmit Level Number (LCT 13/45)	This byte contains the level number for the channel of this line used with interrupts. The level number is contained in six bits (2 to 7), and the two most significant bits of the return channel number are located in bits 0 and 1.

HONEYWELL PROPRIETARY AND CONFIDENTIAL

Table 4-3 Line Control Table Byte Description
(Sheet 2 of 2)

LCT BYTES	DESCRIPTION
Receive/Transmit CLA Status (LCT 14/46)	This byte is used for CLA-LR5: 1. CLA data set change (bits 0 to 4). Refer to the appropriate CLA manual for bit definitions. 2. CLA status storage in bits 5 to 7. See Figure 4-9 for specific bit designations.
Receive/Transmit CLA Status Mask (LCT 15/47)	This byte is used for data set status change detection. Only bits 0 to 4 are used, with the bits which are set to a One in the mask selecting those bits which can be monitored for status change.
Receive/Transmit Status Bytes 1 & 2 (LCT 16,48/ 17,49)	Status bits 0 to 11 are accumulated and updated in these two LCT locations during CCB execution. Bits 12 to 15 are I/O related indicators. Figures 4-10 and 4-11 identify and define the specific bit structures.
Receive/Transmit CCP Subroutine Pointer (LCT 18,50/19,51)	These bytes are used to store the subroutine return pointer utilized by the instruction set. Only one level of subroutine addressing is provided on a per channel basis.
Receive/Transmit CLA Control (LCT 20/52)	This byte is used to monitor the data set control (refer to a specific CLA manual for bit structure (LR2) of bits 0 to 4). See Figure 4-12 for the definitions and configurations of bits 5 to 7.
Transmit LCT Byte Address (LCT55)	This byte contains the address from which the LCT byte is to be delivered in response to the Input LCT Byte command. The address is a six-bit quantity located in bits 2 through 7; bits 0 and 1 disregarded by firmware. If the input LCT Byte command is not used, LCT 55 is considered a CCP working location.

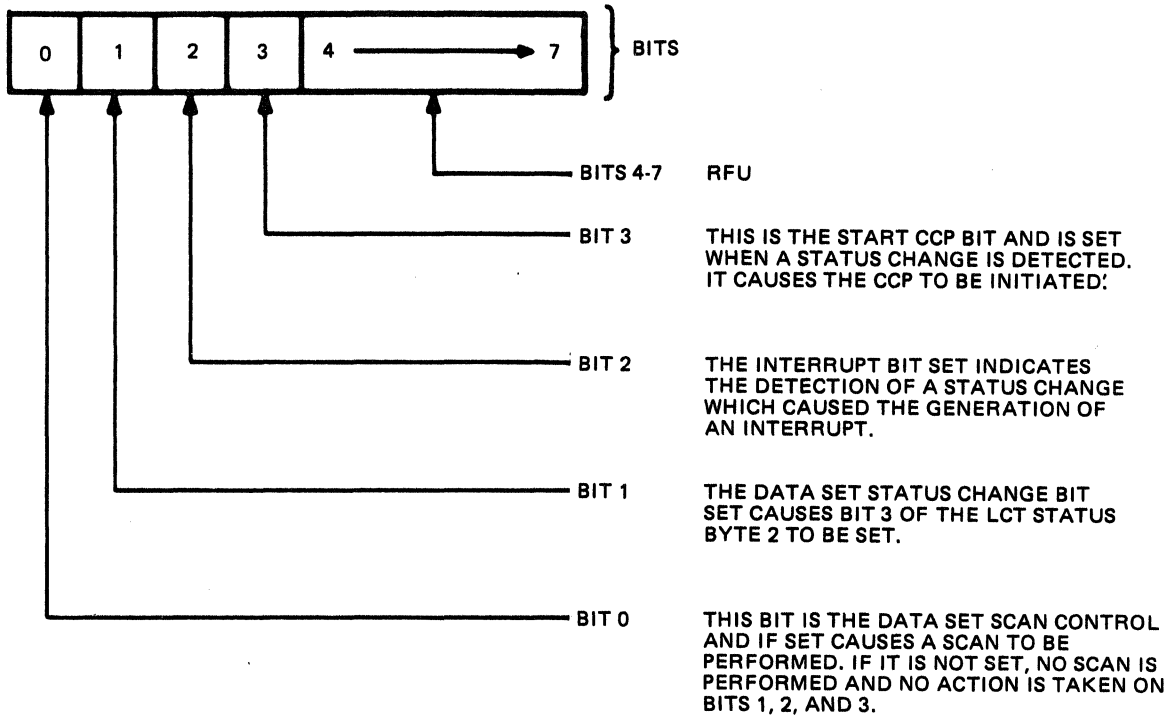


Figure 4-7 LCT Channel Command Byte LCT 8/40 Bit Significance

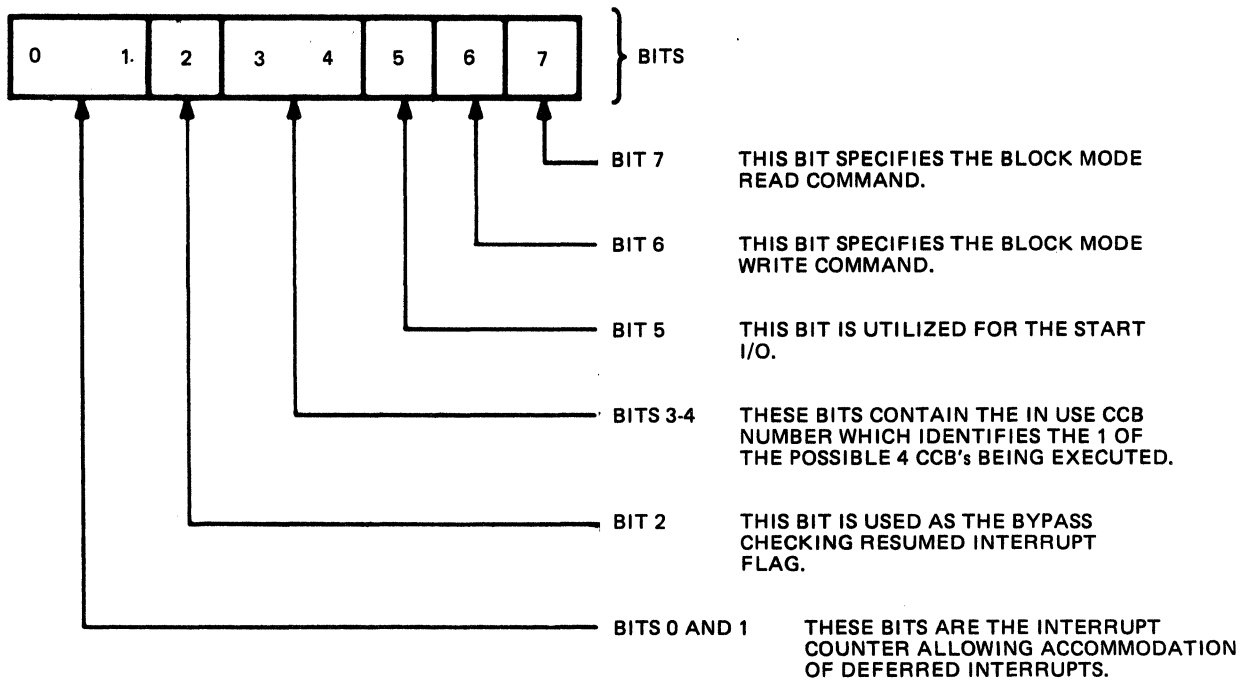


Figure 4-8 LCT Channel Control Byte Bit Significance

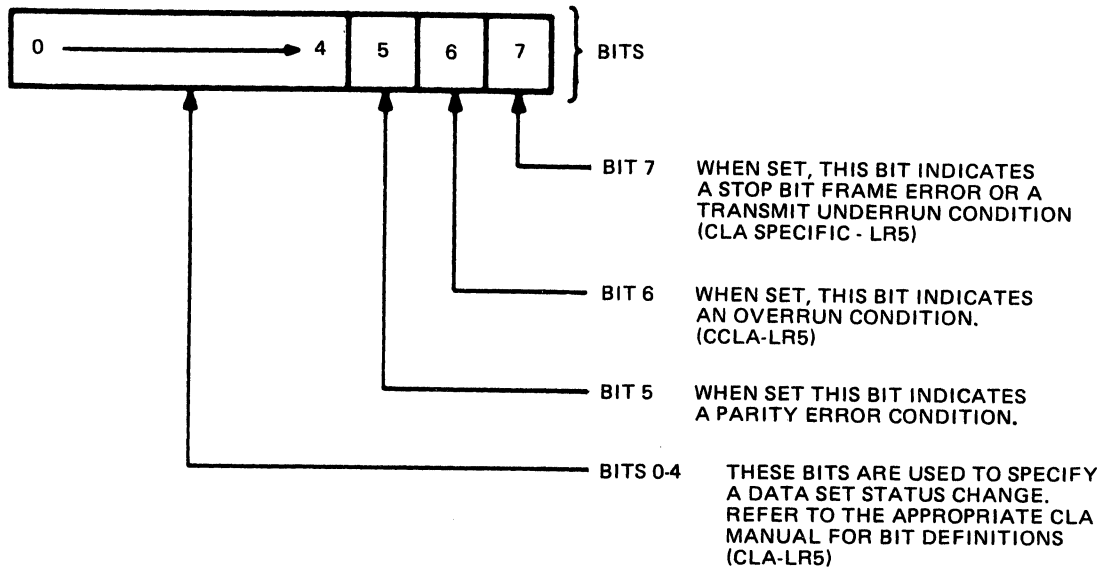


Figure 4-9 LCT CLA Status Byte Bit Significance

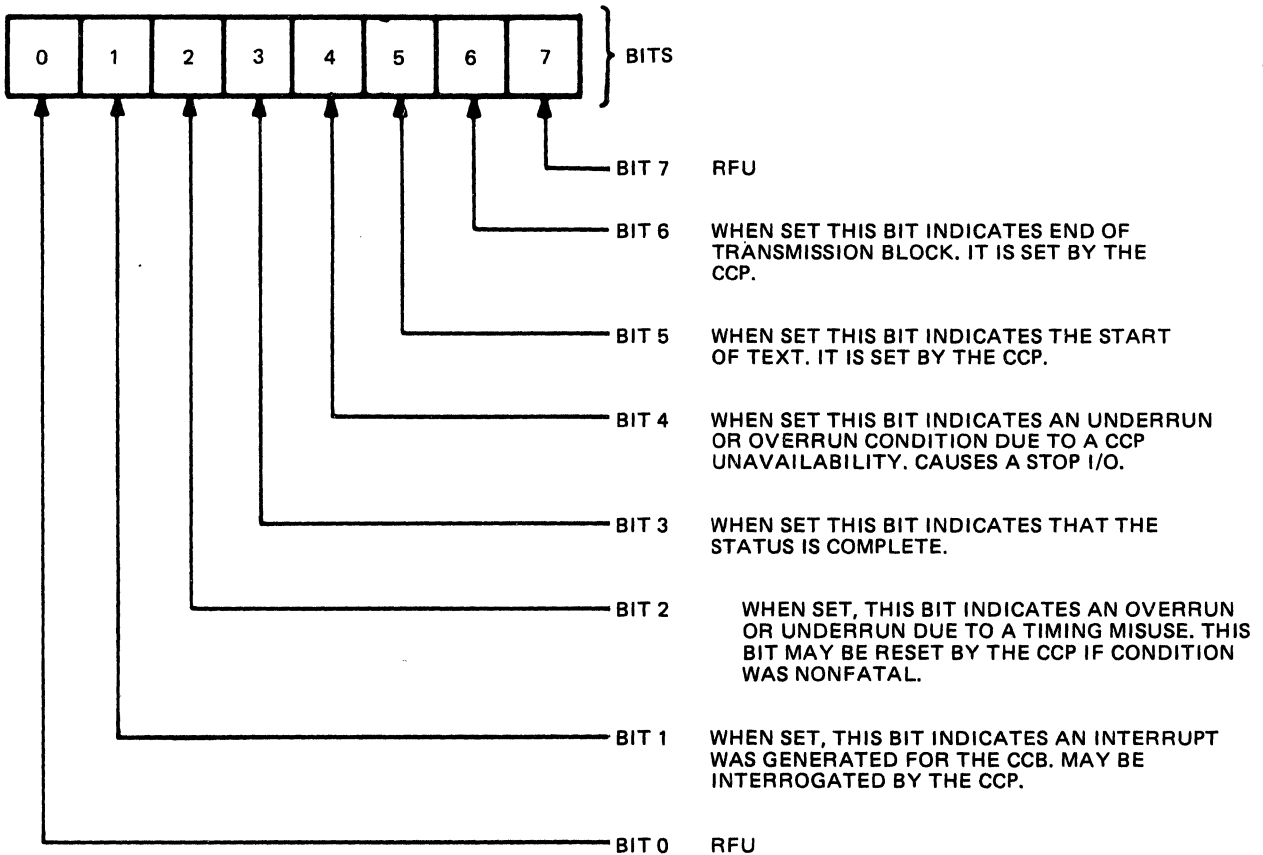


Figure 4-10 LCT Status Byte 1 Bit Significance

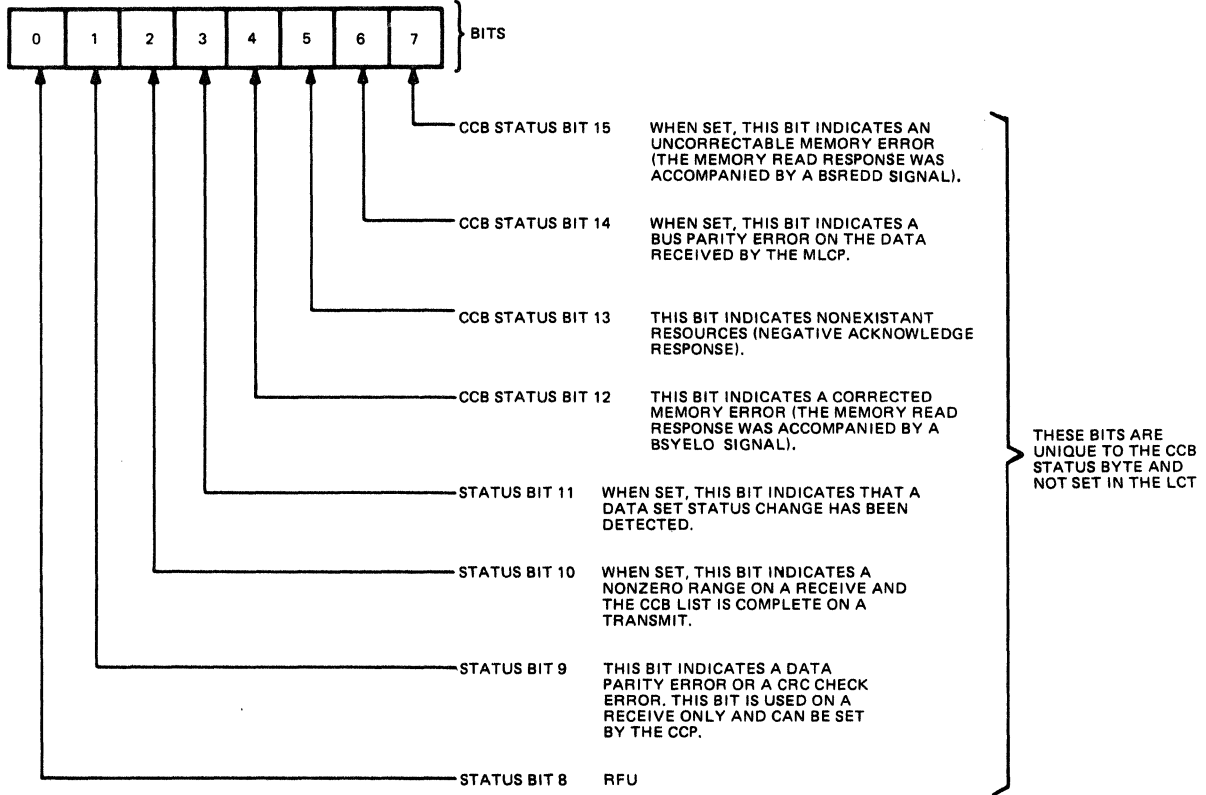


Figure 4-11 LCT Status Byte 2 Bit Significance

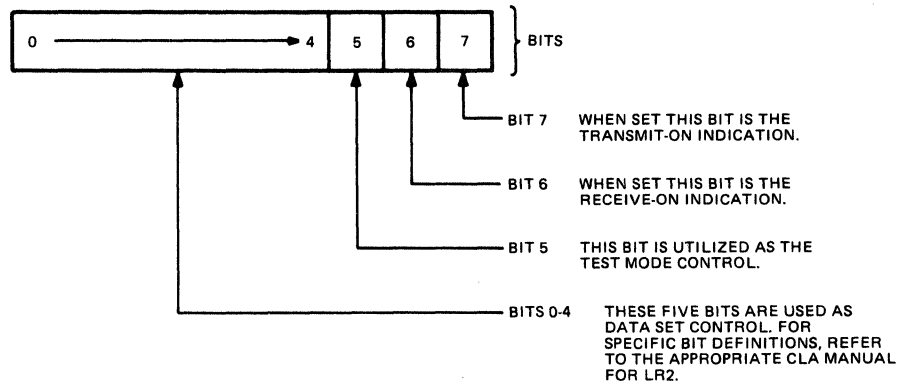


Figure 4-12 LCT CLA Control Byte Bit Significance

4.4.2 Channel Control Program (CCP)

A CCP is stored in the MLCP random access memory (RAM) and is used for application on one or more MLCP channels. CCPs are lists of sequential instructions which are for processing a channel's data and control information character stream. Table 4-4 is a categorization of the instruction set by op code, major format type, and minor instruction. Within the ten major format types, there is an overlapping of the minor instruction set such that they may be grouped into five basic areas: generic, double operand (Next Character, LCT and I-Addressing, and Immediate Operand), channel input/output, branch, and data input/output instructions. An instruction definition and a description for each of the five areas are supplied in Tables 4-5 through 4-9.

A CCP pointer is maintained in the LCT (locations 6/38 and 7/39) which designates the first CCP location to be referenced upon detection of a CLA channel request interrupt which must be serviced. During the processing of a CCP, the firmware stores the current location CCP pointer in registers 8 and 9 to be used for addressing the RAM. When a Wait or a Pause instruction is encountered in a CCP, the contents of registers 8 and 9 are stored in the LCT by firmware. This procedure allows CCPs to be re-entered so that sharing of the CCP between channels is possible.

A CCP is utilized as a procedure for manipulation of a data stream or part of a data stream. It also is capable of handling control status, CCB transitions, data communications equipment devices, CLA configuration, CLA control, CLA status, and all related channel configuration. Each CCP is comprised of a program or arbitrary length, the execution of which is initiated by:

1. A channel request interrupt from the CLA
2. A Start I/O in the channel command byte
3. A data set control change condition for which the start CCP bit is set.

The CCP pointer, the program indicators, and other selected information from the LCT are monitored by the firmware whenever a CCP is executed. The CCP continues until a Wait instruction is processed, at which time execution is suspended until another start CCP condition arises.

The program executed after a Wait instruction is typically that associated with the manipulation of a single character of the data stream. Initialization, configuration, and data set control functions can also be performed after a Wait instruction although this is unnecessary because they are related to the data transfer and can occur as part of the CCP.

Channel control programs are formed in contiguous locations of the RAM so there is no overlap with any other CCP. The last entry in a CCP must be a branch instruction, and each branch of a specific CCP must have an address within its own boundaries.

Table 4-4 Instruction Formats and Decode

BITS 0 to 3 HEX DECODE	MAJOR FORMAT TYPE	MINOR INSTRUCTION HEX DECODE (BITS 4 to 7)															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	Generic	NOP	WAIT	GNB	SFS	CCH	DEC	RET	SHIFT RIGHT	INTR	INZ						
1	Next Character	LD	ST														
2	Inputs LRs	LR0	LR1	LR2	LR3	LR4	LR5	LR6	LR7								
3	Output to LRs	LR0	LR1	LR2	LR3	LR4	LR5	LR6	LR7								
4	RFU																
5	LCT and D Addressing	LD	ST	C	AND	OR	XOR	TLU									
6	Send	No Parity or CRC	CRC	Parity	CRC and Parity												
7, 8	RFU																
9	Immediate Operand	LD		C	AND	OR	XOR										
A	Receive	No Parity or CRC	CRC	Parity	CRC and Parity												
B,C,D	RFU																
E	Branch True	B	BET	BZT	BLCT	BLBT	BLRYT	Jump	BVBT								
F	Branch	BS	BEF	BZF	BLCF	BLBF	BLRYF		BVBF								

*For minor instruction decodes of 8 through F, use the CRC logic.

Table 4-5 General Instruction Set

MINOR INSTRUCTION	INSTRUCTION DESCRIPTION	INSTRUCTION DEFINITION
NOP	No Operation	Effectively delays for one cycle
Wait	Program Wait Go to Next Program	Causes the current CCP P-register contents to be stored in the LCT CCP pointer locations. Causes storage of the indicators and suspends this CCP processing until its next channel request interrupt.
GNB	Get Next Block	Terminates the current CCB and goes to the next CCB.
SFS	Search for Synchronization	Causes the current channel to enter the character synchronization mode.
CCH	Calculate Block Check	Causes the contents of register 5 to be added to the current CRC residue.
DEC	Decrement	Causes the contents of register 5 to be decremented by 1.
RET	Return from Subroutine	Causes the contents of the LCT subroutine pointer to be loaded into the current location P-counter registers 8 and 9.
SR	Shift Right	Shift CCP-visible register one bit right and Zero fill from left.
INTR	Interrupt CPU	Channel program generates an interrupt without terminating the CCB.
INZ	Halt CCP	Initialize all CLA's and clear all channel commands and channel control bytes.

HONEYWELL PROPRIETARY AND CONFIDENTIAL

Table 4-6 Double Operand Instruction Set

MINOR INSTRUCTION	INSTRUCTION DESCRIPTION	INSTRUCTION DEFINITION
LD	Load	Causes EA to be loaded into register 5.
ST	Store	Causes register 5 to be loaded into EA.
C*	Compare	Causes register 5 to be compared with EA and the result of the compare to be reflected in the equal indicator.
AND*	AND	Causes the contents of register 5 to be ANDed with EA and the result stored in register 5.
OR*	OR	Causes the contents of register 5 to be ORed with EA and the result stored in register 5.
XOR*	Exclusive OR	Causes the contents of register 5 to be XORed with EA and the result stored in register 5.
TLU**	Table Look-Up	The TLU instruction may be either a branch or a translate function, depending upon the state of bit 0 of the accessed table location. If a branch, the CCP-visible register is preserved; if a translate, it is changed. See MLCP Programmer's Reference Manual (Order No. AT97) for details.

*These instructions are of the LCT and D Addressing and Immediate Operand sets only.

**This instruction is of the LCT and D Addressing set only.

NOTE

1. EA for next character instruction set is the next character in the data buffer.
2. EA for the LCT and D addressing instruction set is the first location of the current LCT plus the displacement.
3. EA for the immediate operand instruction set is the next character in the CCP.

HONEYWELL PROPRIETARY AND CONFIDENTIAL

Table 4-7 Line Register Input/Output Instructions

INSTRUCTION	DESCRIPTION	INSTRUCTION DEFINITION
In	Input	Causes the contents of the specified line register to be read and stored in register 5.
Out	Output	Causes the contents of register 5 to be written into the specified line register.

Table 4-8 Branch Instruction (Sheet 1 of 2)

MINOR INSTRUCTION	INSTRUCTION DESCRIPTION	INSTRUCTION DEFINITION
B	Unconditional Branch	Causes EA to be loaded into the current location P-counter (registers 8 and 9).
BET	Branch Equal True	If the Equal bit is set, this instruction causes EA to be loaded into the current location P-counter (registers 8 and 9).
BEF	Branch On Equal False	If the Equal bit is reset, this instruction causes EA to be loaded into the current location P-counter (registers 8 and 9).
BZT	Branch on Zero True	If register 5 is equal to zero, this instruction causes EA to be loaded into the current location P-counter (registers 8 and 9).
BZF	Branch on Zero False	If register 5 is not equal to zero, this instruction causes "EA" to be loaded into the current location P-counter (registers 8 and 9).
BLCT	Branch on Last Character True	If the last character indicator is set, this instruction causes EA to be loaded into the current location P-counter (registers 8 and 9).
BLCF	Branch on Last Character False	If the last character indicator is reset, this instruction causes EA to be loaded into the current location P-counter (registers 8 and 9).
BLBT	Branch on Last Block True	If the last block indicator is set, this instruction causes EA to be loaded into the current location P-counter (registers 8 and 9).

Table 4-8 Branch Instruction (Sheet 2 of 2)

MINOR INSTRUCTION	INSTRUCTION DESCRIPTION	INSTRUCTION DEFINITION
BLBF	Branch on Last Block False	If the last block indicator is reset, this instruction causes EA to be loaded into the current location P-counter (registers 8 and 9).
BS	Branch to Subroutine	This instruction causes the current location P-counter plus 1 to be loaded into the LCT CCP subroutine pointer locations. It also causes the current location P-counter (registers 8 and 9) to be loaded with the result of adding the displacement to the present address of the current location P-counter.
BART	Branch on Adapter Ready True	If adapter buffer is not full (transmit) or not empty (receive) this instruction causes EA to be loaded into the current location P-counter (Registers 8 and 9).
BARF	Branch on Adapter Ready False	If adapter buffer is full (transmit) or empty (receive) this instruction causes EA to be loaded into the current location P-counter (Registers 8 and 9).
JUMP	Unconditional Branch	The two byte displacement is algebraically ANDED to the current P-counter contents.
BVBT	Branch On CCB Valid Bit True	If the CCB valid bit is set in the current CCB, this instruction causes EA to be loaded into the current location P-counter (registers 8 and 9).
BVBF	Branch on CCB Valid Bit False	If the CCB valid bit is set in the current CCB, this instruction causes EA to be loaded into the current location P-counter (registers 8 and 9).

NOTE

1. EA is defined as the current location P-counter (registers 8 and 9) plus the displacement.
2. The current location P-counter (registers 8 and 9) is assumed to be pointing at the CCP location of the displacement when the address is calculated.

Table 4-9
Data Input/Output Instructions

INSTRUCTION	DESCRIPTION	INSTRUCTION DEFINITION
Send	Output Data to Channel	Causes the data to be sent to the channel with the proper check characters appended.
Receive	Input Data from Channel	Causes the data to be read from the channel with verification or generation of the check characters.

4.4.3 Communication Control Blocks

Communication Control Blocks (CCB) are implemented by the MLCP in order to describe the address of the data's destination, range, control, and status for each block transfer between the MLCP and the Level 6 System. A CCB is set up by way of I/O instructions in the byte form shown in Figure 4-13.

The eight bytes are divided so that there are three for address, two for range, one for control, and two for status. These fields are defined as follows:

- Address - The 24-bit address is loaded into the CCP initially by way of an IOLD command. The address marks the starting main memory location of a Communications Data Block (CDB) used in an MLCP/system data transfer.
- Range - The 16-bit range is loaded into the CCB initially by way of an IOLD command. The range defines in bytes the size of the CDB to be read or written during CCB execution. This field is decremented once for each byte transferred.
- The control field contains CCB-specific control information. See Figure 4-14 for specific bit designations of this control byte.
- Status - The status bytes are used for storage of status which has been accumulated in the LCT during CCB execution. These status bytes also have bit designations for the CCB which are not found in the LCT status word. Figures 4-12 and 4-13 show the bit significance of these bytes. The CCB status word is reset to Zero when an IOLD command for the CCB is executed.

CCBs are accessed by the MLCP firmware during their execution; however, they are not available to the MLCP CCP procedures. During a transfer the address field is incremented by one for each byte passed via the firmware DMA routines. For CCBs executed on a transmit, the address points to the location of the last CDB byte transferred to the MLCP. In the case of a receive, the address points to the location of the last CDB byte sent from the MLCP to the main memory.

HONEYWELL PROPRIETARY AND CONFIDENTIAL

The CCB range count is always decremented by one for each byte transfer until it has reached zero. A termination condition may also request the end of the block, causing a range result (residue) that is not zero. The range count (zero or nonzero) for the affected CCB will remain at this value until an IOLD command to the same CCB overwrites it. The range field, if nonzero, will contain a count equal to the number of bytes for which DMA transfers did not occur.

When a CCB is terminated due to range depletion or a forced termination, the status bytes for this CCB are updated with the information located in the LCT status word. The CCB execution is complete only after the status update is complete as indicated by the status complete bit in status byte 1.

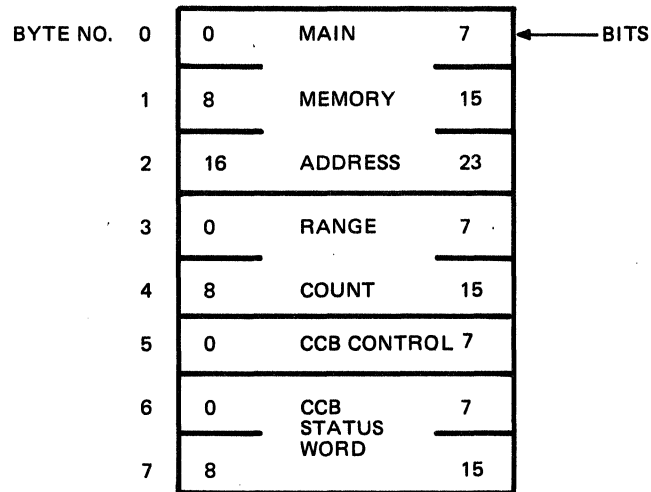


Figure 4-13 Eight-Byte CCB Format

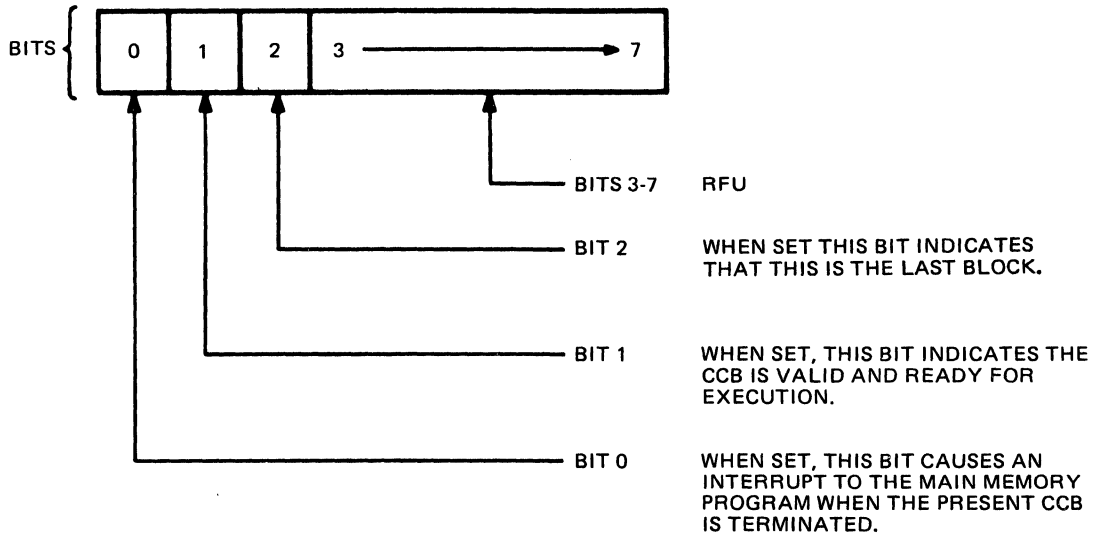


Figure 4-14 CCB Control Byte Bit Significance

4.5 INTERMEDIATE FLOW CHARTS

The firmware flow charts in this manual are generated at an intermediate level, designed more as an aid in understanding firmware routine functions rather than as a maintenance aid. A description and illustration of the symbology used in the intermediate flow chart is contained in subsection 4.5.1 with subsection 4.5.2 describing the flow chart organization. Information concerning the use of these automated flow charts in conjunction with an actual flow chart example is detailed in subsection 4.5.3.

4.5.1 Intermediate Flow Chart Symbology

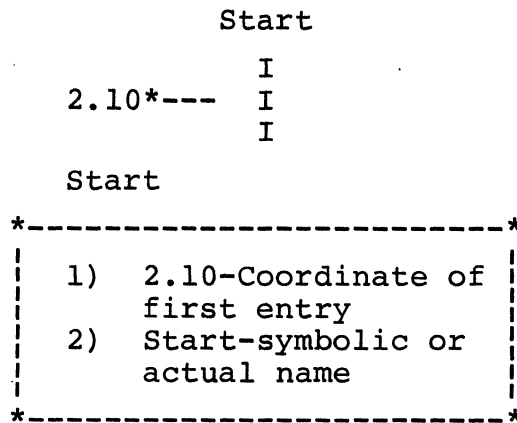
The intermediate flow charts are a composite of symbols each of which has a specific application. In some instances the same symbol may have more than a single meaning depending upon its usage. A description and illustration of each symbol that may appear in the intermediate flow charts is found in Table 4-10.

4.5.2 Flow Chart Organization

The symbols found on each individual flow chart sheet are numbered sequentially starting with the symbol numbered 01. The numbering process is initiated from the leftmost column with the symbol number 01 appearing outside and at the upper right-hand corner of the first symbol in the left column.

The first symbol number on each new flow chart sheet starts with an 01. Therefore, any symbol can be accurately located by citing the sheet number on which it appears and the symbol number on the sheet. This information is referred to as the coordinate of the symbol and is normally expressed in the format PG.BX, where PG is the sheet number and BX is the symbol number. For example, a symbol that has the coordinate of 03.07 is the seventh symbol found on sheet three of the flow chart.

In some instances due to the flow chart layout it is not feasible to draw the connecting line from one symbol to another. When this situation arises a coordinate will be implemented at the symbol which is the destination indicating that entry was from another portion of the flow chart. In the following example the coordinate 2.10 indicates that there is a logical connection between this symbol and the tenth symbol on sheet two. This application of a coordinate is called an in-connector. The asterisk (*) following the coordinate indicates that an entry to this symbol from some other point in the flow chart exists, and that this (2.10) is just the first.



The preceding example of an in-connector also illustrates that any flow chart symbol can be assigned a name. This name could represent up to ten digits of a firmware address or tag associated with a particular line of the firmware file, or it can represent a symbolic address for use as the destination of a branch operation. Regardless of the name's use, it will be located outside the upper left-hand corner of the symbol.

4.5.3 Flow Chart Usage

Figure 4-16 is an illustration of an intermediate flow chart utilizing one of the MLCP firmware routines as an example. This flow chart contains a description of the overall operations being performed by one or more firmware commands being executed. With one exception only, the name (or address) associated with the addresses or tags actually located in the firmware listing will be shown outside the upper left-hand corner of the symbol. The exception to this is that if a symbolic destination tag (name) is required for a test or branch operation, the tag will not appear in the firmware listing.

Since the operation performed in one or more cycles may be "don't care" operations (i.e., operations which are not reflected in the overall procedure being performed), only those commands necessary to justify an end result will be incorporated into the intermediate flow chart.

Table 4-10 Cycle Flow Chart Symbols (Sheet 1 of 6)

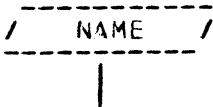
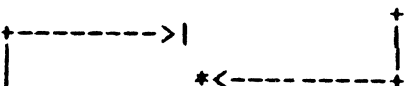




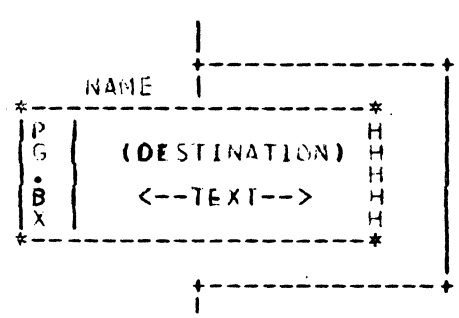
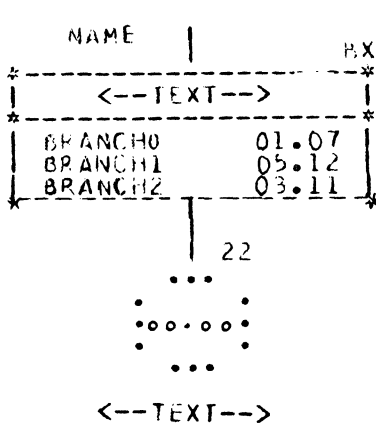
OPERATION	SYMBOL	DESCRIPTION
Initial Connector		<p>This symbol always begins a new flow chart "cluster." The name is taken from the next symbol. A flow chart cluster is the flow path tree created during the chart drawing process.</p>
Direct Connectors	<p>HORIZONTAL CONNECTORS</p>  <p>VERTICAL CONNECTORS</p>  <p>'T' CONNECTORS</p>  <p>CROSS CONNECTORS</p> 	<p>These symbols are examples of mechanical representation of hand-drawn flow lines. Such lines can cross with or without connection. A tied cross point is indicated by a + symbol.</p>

Table 4-10 Cycle Flow Chart Symbols (Sheet 2 of 6)

OPERATION	SYMBOL	DESCRIPTION
Back-referencing Connectors	<pre> * -----*---> PG.BX*----> </pre>	<p>The * entry in a flow path line indicates that more than one reference exists at that point of entry. Coordinate information is provided from remote branch points. Only the first such reference is given. The absence of an * indicates that the branch path is unique; that is, no 'FAN-IN' of logical flow is indicated.</p>
Intermediate Connector	<pre> -----/ /PG.BX </pre>	<p>This symbol is generated whenever the flow path is continued in the near vicinity. It arises due to physical limitations of page size.</p>
Process	<pre> NAME BX -----*-----* <--TEXT--> -----*-----* </pre>	<p>The symbol shows in-line computational processing. Box depth is determined by extent of the enclosed comment.</p>
Subroutine Call	<pre> NAME BX -----*-----* P (DESTINATION) H G <--TEXT--> H E H X H -----*-----* </pre>	<p>A subroutine is given control. Return is assumed to be in-line. Box encloses a destination name and coordinate reference which appear vertically on the left as PG.BX.</p>

Table 4-10 Cycle Flow Chart Symbols (Sheet 3 of 6)

OPERATION	SYMBOL	DESCRIPTION
Branch	<pre> NAME BX ... :PG.BX: ...DESTINATION <--TEXT--> <--TEXT--> </pre> 	<p>This symbol represents an unconditional branch. The flow path is terminated and any accompanying text appears below the fixed-sized branch symbol.</p> <p>The destination path is drawn as a direct flow line whenever possible. As before, text appears below the break.</p> <p>NOTE</p> <p>This symbol may or may not represent an actual Branch instruction in the firmware.</p>
Asynchronous Subprogram Initiation	<pre> NAME ----- ----- * (DESTINATION) * P <--TEXT--> H G H . H B H X H * * </pre> 	<p>The initiation of a parallel process is indicated. Return is not assumed and the flow path is made to bypass the box. Coordinate reference appears on the left.</p>
Branch Table	<pre> NAME BX ----- ----- * <--TEXT--> * * BRANCH0 01.07 * BRANCH1 05.12 BRANCH2 03.11 * * </pre> 	<p>The Branch Table is used when multiple branch paths exist. The branch condition is described in the TEXT box that appears above the branch table. The destination coordinates for each branch path appear on the right. The flow is terminated by a Branch symbol with a 00.00 coordinate.</p>

HONEYWELL PROPRIETARY AND CONFIDENTIAL

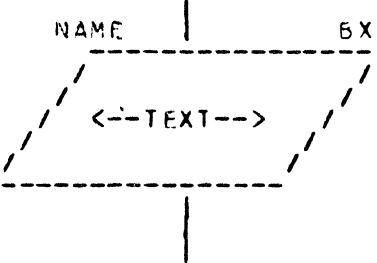
Table 4-10 Cycle Flow Chart Symbols (Sheet 4 of 6)

OPERATION	SYMBOL	DESCRIPTION
EXIT	<pre> NAME BX ----- * EXIT * ----- <--TEXT--> </pre>	<p>This symbol shows an exit flow path terminator. Accompanying text appears immediately below the symbol.</p>
EXIT or Continue	<pre> NAME BX ----- * EXIT * ----- <--TEXT--> -----> </pre>	<p>This construction is used for EXIT instruction sequences which are executed conditionally, depending on processing elsewhere in the flow chart.</p>
HALT	<pre> NAME BX ----- * HALT * ----- <--TEXT--> </pre>	<p>This symbol shows a HALT flow path terminator. Accompanying text appears immediately below the symbol.</p>
HALT, Continue	<pre> NAME BX ----- * PAUSE * ----- <--TEXT--> </pre>	<p>This symbol represents a HALT without flow path termination. Accompanying text appears immediately below the symbol and the flow path continues downward.</p>
NOTE	<pre> NAME NOTE BX * * * * * * * * * * * <--TEXT--> * * * * * * * * * * * </pre>	<p>A NOTE is a descriptive or explanatory notation embedded directly in the flow path.</p>

Table 4-10 Cycle Flow Chart Symbols (Sheet 5 of 6)

OPERATION	SYMBOL	DESCRIPTION
Decision	<p>The top diagram shows a diamond-shaped decision symbol. At the top, the word 'NAME' is on the left and 'BX' is on the right, separated by a vertical line. Below this, a horizontal line contains the text '<--TEXT-->'. On the left and right sides, there are vertical lines leading to the word 'LABEL'. At the bottom, there is a vertical line leading to a box containing 'PG' and 'BX', with the word 'DESTINATION' below it.</p> <p>The bottom diagram shows a similar diamond-shaped decision symbol. At the top, the word 'NOTE' is on the left and 'BX' is on the right, separated by a vertical line. Below this, a horizontal line contains the text '<--TEXT-->'. On the left and right sides, there are vertical lines leading to the word 'LABEL'. In the center of the diamond, the text 'SEE NOTE ABOVE' is written. At the bottom, there is a vertical line leading to a box containing 'PG' and 'BX', with the word 'DESTINATION' below it.</p>	<p>The diamond shape is the basic decision symbol; it is fixed in size. Whenever possible, any text is fitted within the box. If the text does not fit, it appears in a note which immediately precedes the diamond. SEE NOTE ABOVE appears as the text within the diamond. Notice that the left-hand path continues to another symbol not illustrated here. The decision symbol may involve two or three possible branch paths, each having a distinct label consisting of up to five characters. The in-line label continues directly from the bottom of the symbol to the very next symbol. All other labels are called out-of-line. A decision connector is generated whenever directly connecting flow paths cannot be drawn.</p>
Decision Connector	<p>The diagram shows a vertical line leading to a box containing 'PG' and 'BX', with the word 'DESTINATION' below it.</p>	<p>This fixed-length symbol is generated for cases in which an out-of-line path cannot be directly connected by a line on the same chart page.</p>

Table 4-10 Cycle Flow Chart Symbols (Sheet 6 of 6)

OPERATION	SYMBOL	DESCRIPTION
Input/Output		<p>This is the distinctive fixed size I/O symbol. If accompanying text cannot be completely contained within this symbol, it will appear as a note immediately preceding the I/O parallelogram, as previously described for the decision symbol with excessive text.</p> <p style="text-align: center;">NOTE</p> <p>This symbol is only used for descriptive purposes in the flow charts.</p>

4.6 FIRMWARE CYCLE FLOW

4.6.1 MLCP Overview Flow Chart

The MLCP firmware is divided into essentially eight major areas with many of the areas comprised of a combination of multiple sub-routines. Figure 4-15 shows the subroutine names for each block, and a figure reference for the individual firmware routines.

Upon initialization of the MLCP the firmware enters a routine for verification of the hardware referred to as the Quality Logic Test (QLT). The QLT has only one possible path for firmware continuation: entry into the Service Scan routine. At this point it is necessary for the firmware to decide which of its three available routes it must follow.

HONEYWELL PROPRIETARY AND CONFIDENTIAL

Provided that there are no requests of any type present, the Service Scan routine will exit to the Background routine. The process of looping from the Service Scan routine to the Background routine and back to the Service Scan routine continues until one of four conditions occurs:

1. A deferred interrupt is detected.
2. A data transfer is initiated.
3. A data set scan is required.
4. A start I/O is detected.

When any one of these situations is present, the Background routine will enter the applicable firmware area: Deferred Interrupt Handling routines, DMA Service/Execution routines, Data Set Scan routine, or the Start I/O entry of the channel program initiation. When completion of the required routines is accomplished, re-entry into the service scan/background loop is performed, except for the Start I/O which enters the Line Adapter Ready routine.

If an I/O command was detected while in the Service Scan routine, an area called the I/O Command Decode/Execution routine is entered. After processing the proper command operation, the firmware may return to the service scan/background loop; if a data transfer is necessary, it will go to the DMA Service/Execution routine; or if an I/O Interrupt is detected, it will go to the Line Adapter Ready routine.

The third path provided for in the Service Scan routine is to the Line Adapter Ready routine (if a line adapter request is present). The Instruction Fetch/Execution routines are subsequently utilized to provide the appropriate data manipulation.

The processing of a CCP is continued until a Wait, Pause, or I/O Command Interrupt instruction is recognized, causing an exit from the Instruction Fetch/Execution routines to the Service Scan routine.

4.6.2 Firmware Flow Charts (See Figures 4-16 through 4-42)

4.6.2.1 Quality Logic Test (QLT)

The QLT (see Figure 4-16) is initiated by way of the Output MLCP Control command. It is utilized to clear the entire 4K random access memory to zero. During this clearing procedure the addressing, next address generation, random access memory storage elements, refresh logic, and other hardware elements are verified. The data is written, then read and compared for a zero result. In the event of a data error, the firmware enters a location in which a branch on itself is performed, effectively halting microinstruction processing. After the successful completion of the QLT, a firmware command is issued which extinguishes the LED located at the rear edge of the MLCP board, and the firmware enters the Service Scan routine.

4.6.2.2 Service Scan Routine (See Figure 4-16)

The Service Scan routine can be entered from any one of the major areas of the firmware, which are shown in Figure 4-15. This routine is used to determine what, if any, type of request is pending.

Initially the Service Scan routine ascertains if the I/O Command line is set. If set, the firmware proceeds to the I/O Command Decode routine, which establishes the necessary action to be taken.

Next, the line adapter ready flip-flop is scanned to establish whether a line adapter request is present. If this flip-flop is set, the firmware begins the Line Adapter Ready routine and starts the processing and execution of the Channel Control program.

The Service Scan routine will go to the Channel Scanner routine when there are no line adapter or I/O command requests.

4.6.2.3 Channel Scanner Routine

The Channel Scanner routine (see Figure 4-16) is referred to as the Background routine. This routine executes several low priority process checks to determine if any action on the part of the firmware is necessary.

The Channel Scanner routine first checks to see if a block mode operation is pending. If any of these conditions are met, the firmware branches to the appropriate Block Mode routine within the DMA Service/Execution routine area and performs the required action.

The Channel Scanner routine then checks the start I/O flip-flop. If start I/O is set, firmware enters the Channel Program Initiation routine at the start I/O entry.

The Channel Scanner routine will then, if no prior action was required, determine if there is a deferred interrupt pending on this channel. If the deferred interrupt flag is set, the routine goes to the Deferred Interrupt routine and processes the interrupt.

The last check made by the Channel Scanner routine is to establish if a data set scan is required. When the scan bit is set, the Data Set Scan routine is entered, and the scan procedure is implemented.

After all these functions have been checked, and if no low priority actions are necessary, the Channel Scanner routine re-enters the Service Scan routine to scan high priority requests (refer to subsection 4.6.2.2).

4.6.2.4 Instruction Fetch/Execution Routines (Figures 4-17 and 4-22 through 4-30)

The Instruction Fetch/Execution routines are those portions of the firmware utilized for implementation of the Channel Control Program (CCP). The CCPs are delegated to a section of the random access memory and in order to be implemented their instruction must be read from storage, decoded, and finally executed. The firmware routines performing these operations consist of three phases:

HONEYWELL PROPRIETARY AND CONFIDENTIAL

1. The instruction fetch
2. The instruction decode
3. The instruction execution.

The Instruction Fetch routine accesses the P-counter located in registers 8 and 9 and loads it into the random access memory address register. When the address register is loaded, the P-counter is incremented and restored to registers 8 and 9. The random access memory location is then read to obtain the instruction for the decode process.

The decode process consists of two levels: a major decode and a minor decode. The major decode is a one-of-eight identifier which determines the format type (refer to Table 4-4). Once the format type has been established, the minor decode specifies the actual instruction of this format to be performed.

Finally, the instruction execution routines perform the indicated action and if applicable may set or reset the CCP program indicators.

The processing of a CCP is continued by repeating these three operations until a Wait, Pause, or I/O Command Interrupt instruction is recognized, causing an exit from the Instruction Fetch/Execution routines to the Service Scan routine.

4.6.2.5 I/O Command Decode/Execution Routines (Figures 4-18 through 4-21)

Firmware response is not required for all types of I/O commands. Where firmware action is required, the function code read from the I/O data buffer register is decoded to determine if an input or output operation is to be performed. Once this functionality is decoded, a further decode is performed to allow the correct execution routine to be entered.

Entry into the I/O Command Decode/Execution routines is gained from the Service Scan routine if an I/O command is detected. The DMA Overhead routine (Figure 4-35) utilizes the Service Scan routine to access the I/O Command Decode/Execution routine if use of the I/O data buffer register is required. After the I/O command execution, when entry is from DMA overhead, the DMA routine is returned to for continuation or exit to the Service Scan routine. When the Command Decode/Execution routines have been entered from the Service Scan routine, the exit will be back to the Service Scan routine.

4.6.2.6 DMA Service/Execution Routines (Figures 4-31 through 4-39)

The DMA Service/Execution routines are utilized to read data from or write data to the system main memory. Many of the routines within this area are used to update address, range, data, and pertinent status information.

4.6.2.6.1 DMA Write Operation

This operation is performed to transmit data from the MLCP into the main memory and requires the use of several of the routines which comprise the DMA Service/Execution area. These routines request the I/O data register and gain access in one of two ways if it is busy:

1. If busy because of the hardware, a stall is performed until the hardware is available.
2. If busy because of an I/O command pending, the command is processed, and a return to these routines is performed.

The routines then load the three bytes of address and the two of data into the I/O data register. Control signals are then set, and the DMA cycle initiated. The firmware then waits for cycle completion and exits to the Service Scan routine directly or by way of the Termination routine.

4.6.2.6.2 DMA Read Operation

This operation is performed to transfer (receive) data from the main memory into the MLCP, and requires the application of several routines which comprise the DMA Service/Execution routines. The routines request access to the I/O data register in one of two ways if it is busy:

1. If busy due to the hardware, a stall is initiated until the hardware is available.
2. If busy because of an I/O command pending, the command is processed, and a return to these routines is performed.

The routines then load the main memory address and the channel number of the source into the I/O data register. Control signals are then set, and the request cycle is initiated. The firmware then stalls, waiting for a response from the main memory. In the event of a NAK response, the nonexistent resources error is set, and the Service Scan routine is entered by way of the Termination routine. If an acknowledge was received by the MLCP, the firmware then waits for a Second Half Read cycle, at which time the data integrity is verified. The data bytes are loaded into the random access memory, and an exit to the proper procedure is performed.

4.6.2.7 Deferred Interrupt Handling Routines
(Figures 4-40 and 4-41)

The Deferred Interrupt Handling routines utilize a counter which indicates the number of different interrupts which have been NAK'd previously. If the counter is at some value other than zero for the scan of this channel, the firmware will test the Resume Interrupt indicator. If this indicator is reset, the bypass resume interrupt flag is tested. If this bit is also reset, the Service Scan routine is entered.

In the case where the bypass resume interrupt flag is set, the bypass flag is cleared, and firmware enters the same point that would be entered if the resume interrupt indicator was set.

HONEYWELL PROPRIETARY AND CONFIDENTIAL

In the case where the resume interrupt indicator is set, the firmware request I/O access to determine if an I/O command is pending. If it is, firmware exits to the I/O Command Decode routine. If not, and the hardware logic is not busy, the deferred interrupt counter is decremented, and the interrupt is generated. To execute the interrupt generation, the I/O data register is loaded with:

1. The channel number of the destination
2. The interrupt level number
3. The channel number of the source.

The firmware then sets up the proper Megabus control signals and initiates a request cycle. A stall is used to wait for the response, and if the response is a NAK, the deferred interrupt counter is incremented, and the routine requesting the interrupt is returned to. When an acknowledge response is received, the requesting routine is returned to directly without incrementing the deferred interrupt counter.

4.6.2.8 Data Set Scan Routine (Figure 4-42)

The Data Set Scan routine is entered from the Channel Scanner routine as a result of bit 0 of this channel's LCT command byte (see Figure 4-7) being set.

The routine takes the necessary action to read the status of this channel from the adapter. When the new status is received from the adapter, it is compared against the adapter status stored in the LCT. If the status is unchanged, a return to the Service Scan routine is performed. If the new status from the adapter has changed, the new status is written into the LCT, and the LCT channel command byte is examined to ascertain what action is necessary.



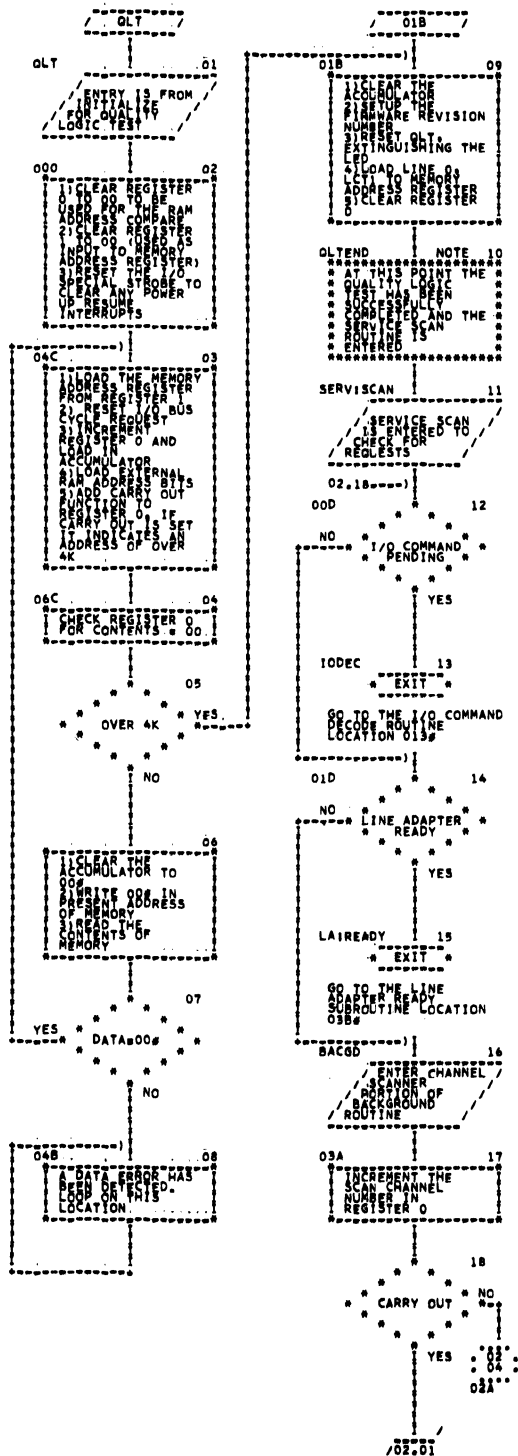


Figure 4-16 Quality Logic Test/Service Scan/Channel Scanner Routines (Sheet 1 of 2)

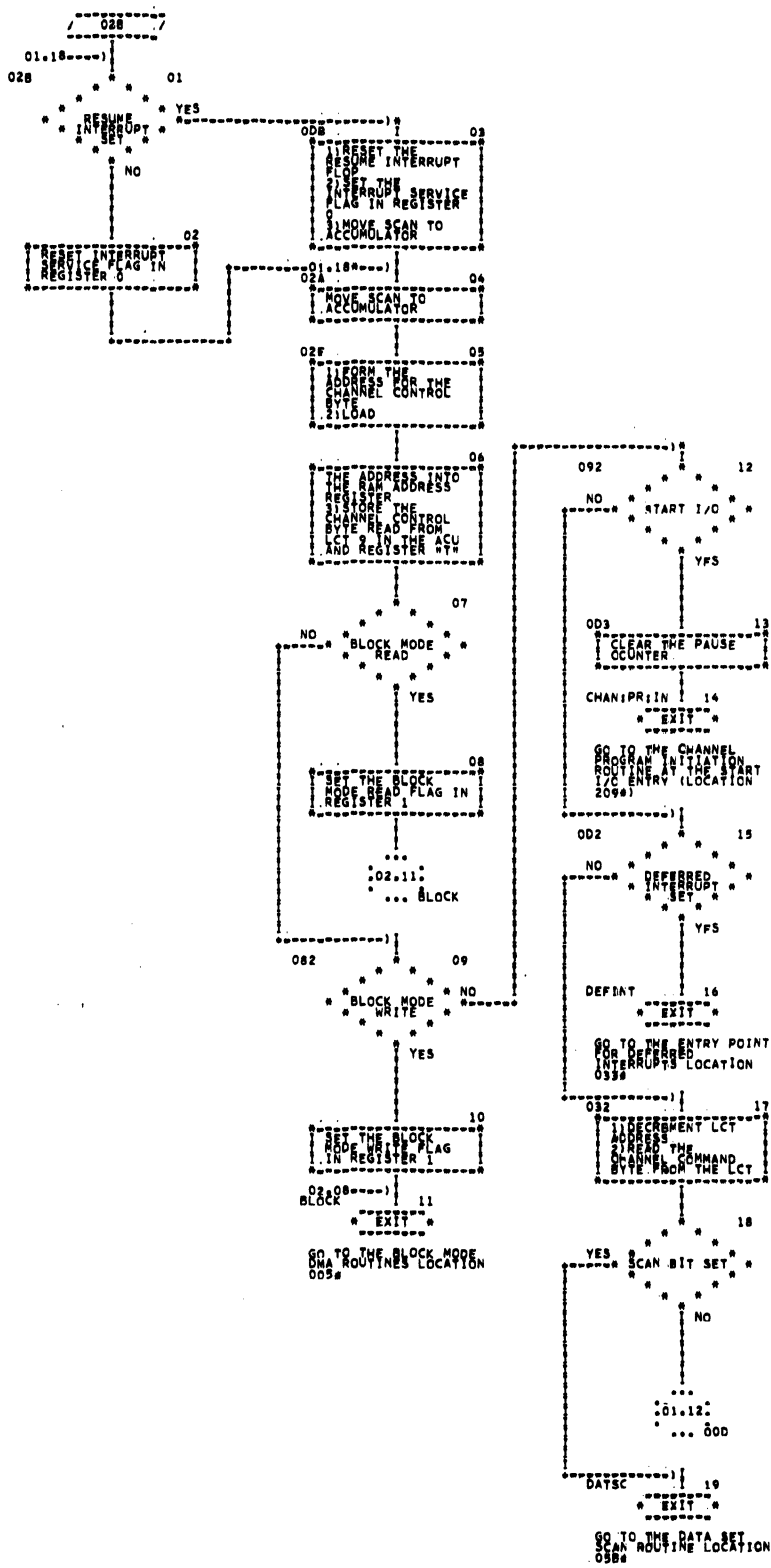


Figure 4-16 Quality Logic Test/Service Scan/Channel Scanner Routines (Sheet 2 of 2)

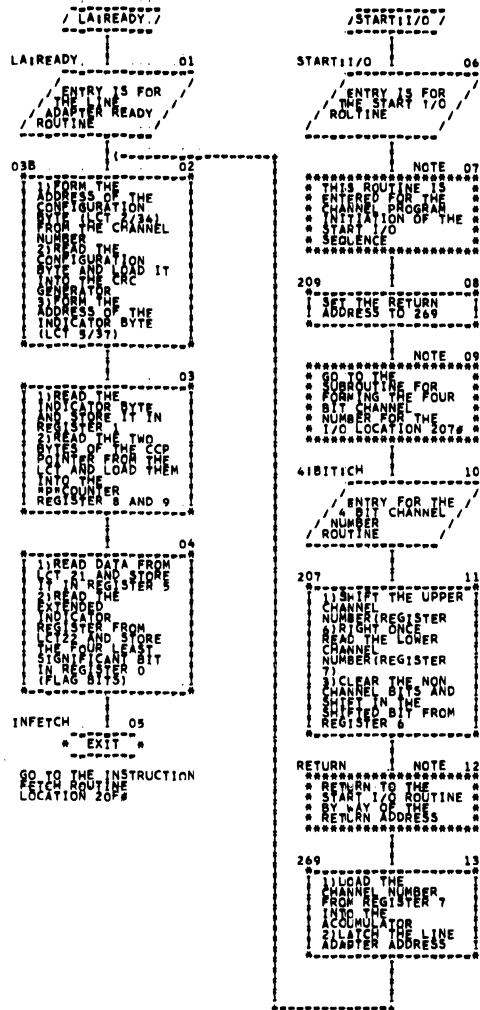


Figure 4-17 Line Adapter Ready/Start I/O/Form Four-Bit Channel Number for I/O Routines

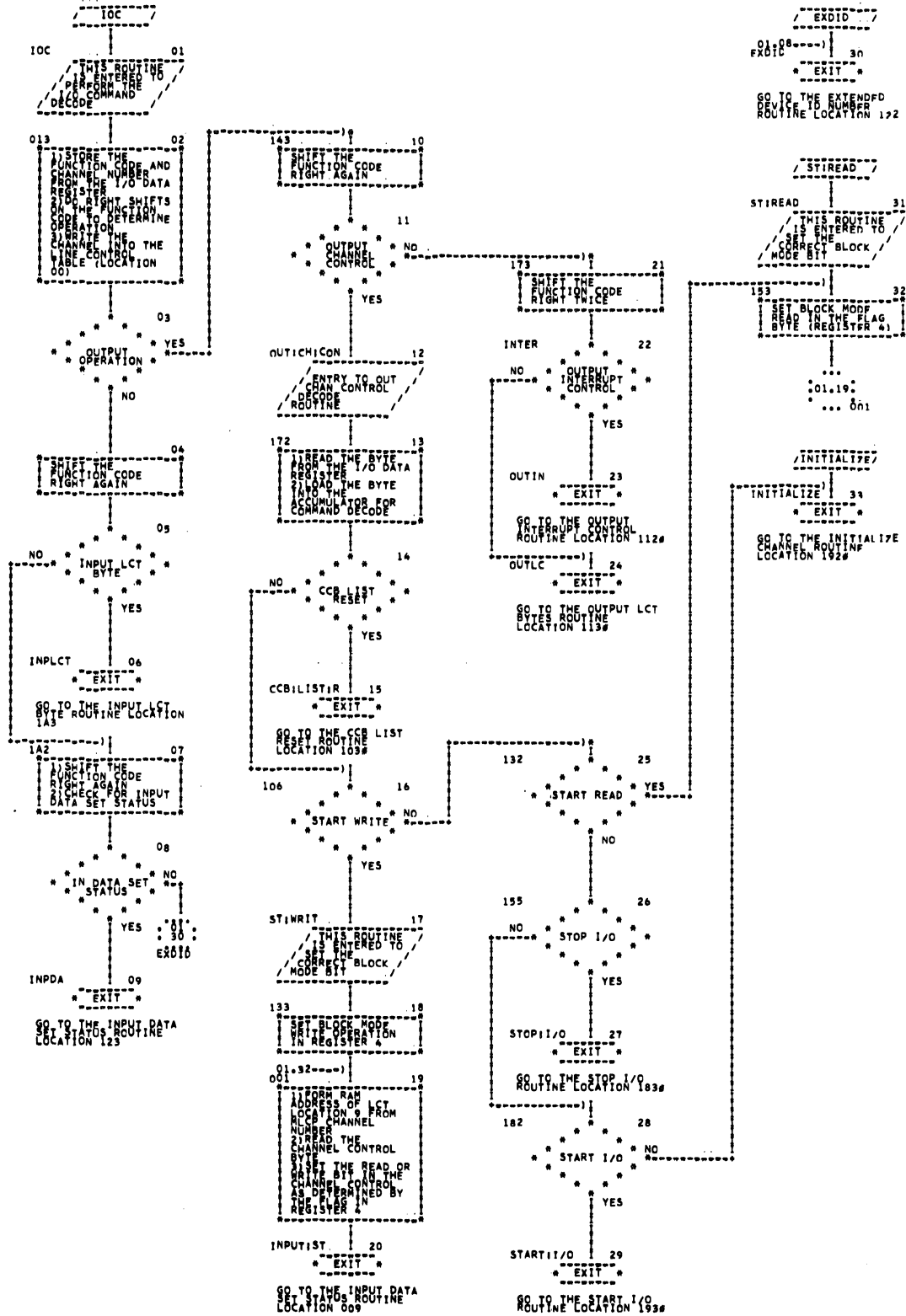


Figure 4-18 I/O Command Decode/Output Channel Control/Start Write/Read Routines

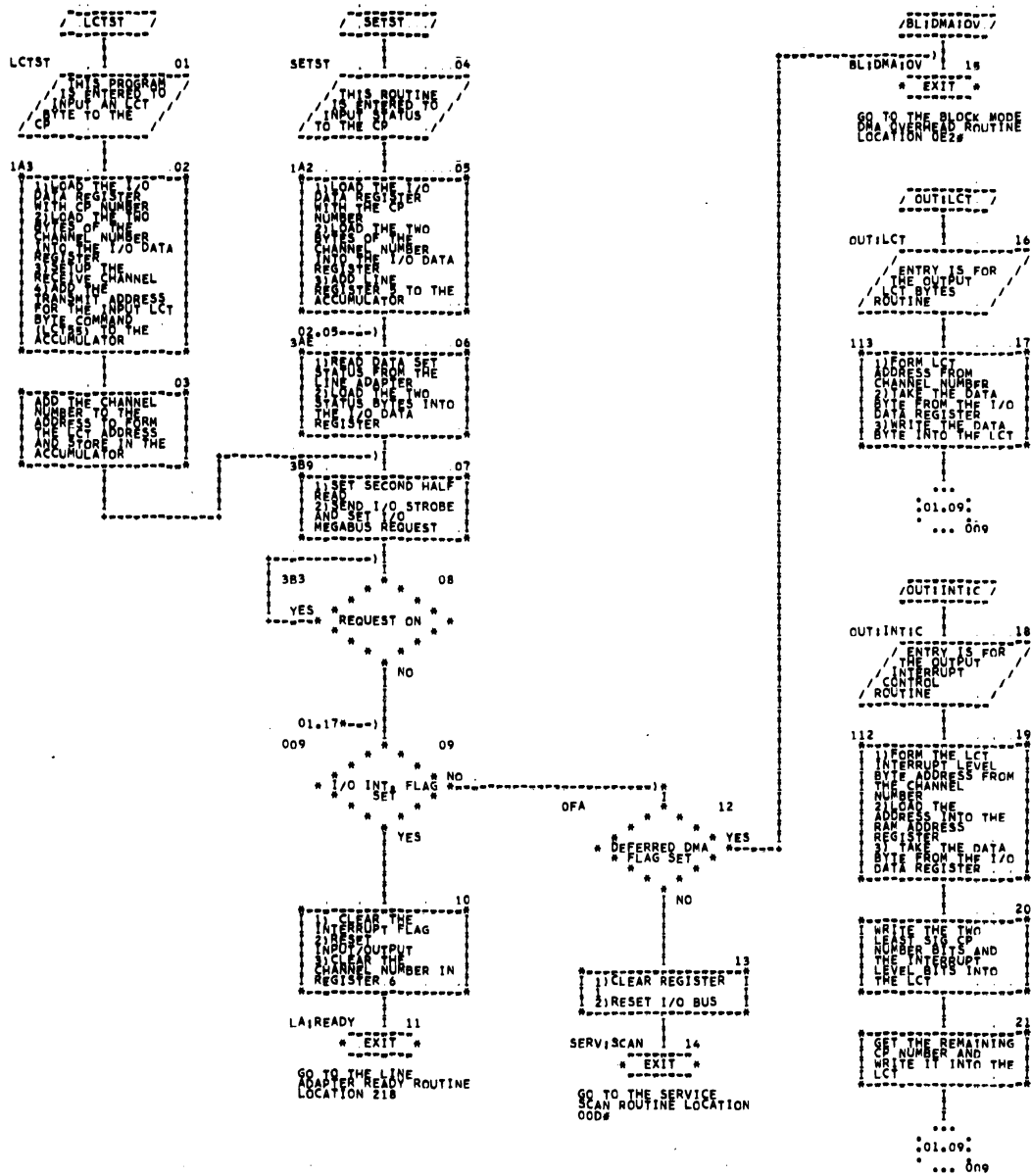


Figure 4-19 Input LCT Byte/Input Data Set Status/
Output Interrupt Control/Output LCT Bytes/
Set Start I/O in LCT 9/Input Extended Device
ID Number Routines (Sheet 1 of 2)

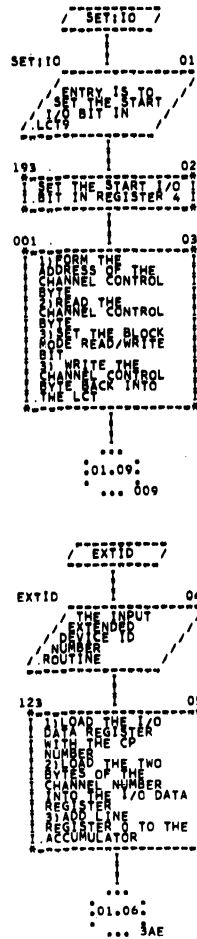


Figure 4-19 Input LCT Byte/Input Data Set Status/
 Output Interrupt Control/Output LCT Bytes/
 Set Start I/O in LCT 9/Input Extended Device
 ID Number Routines (Sheet 2 of 2)

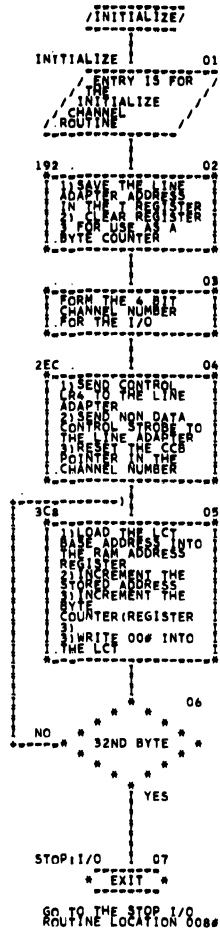


Figure 4-20 Initialize Channel Routine

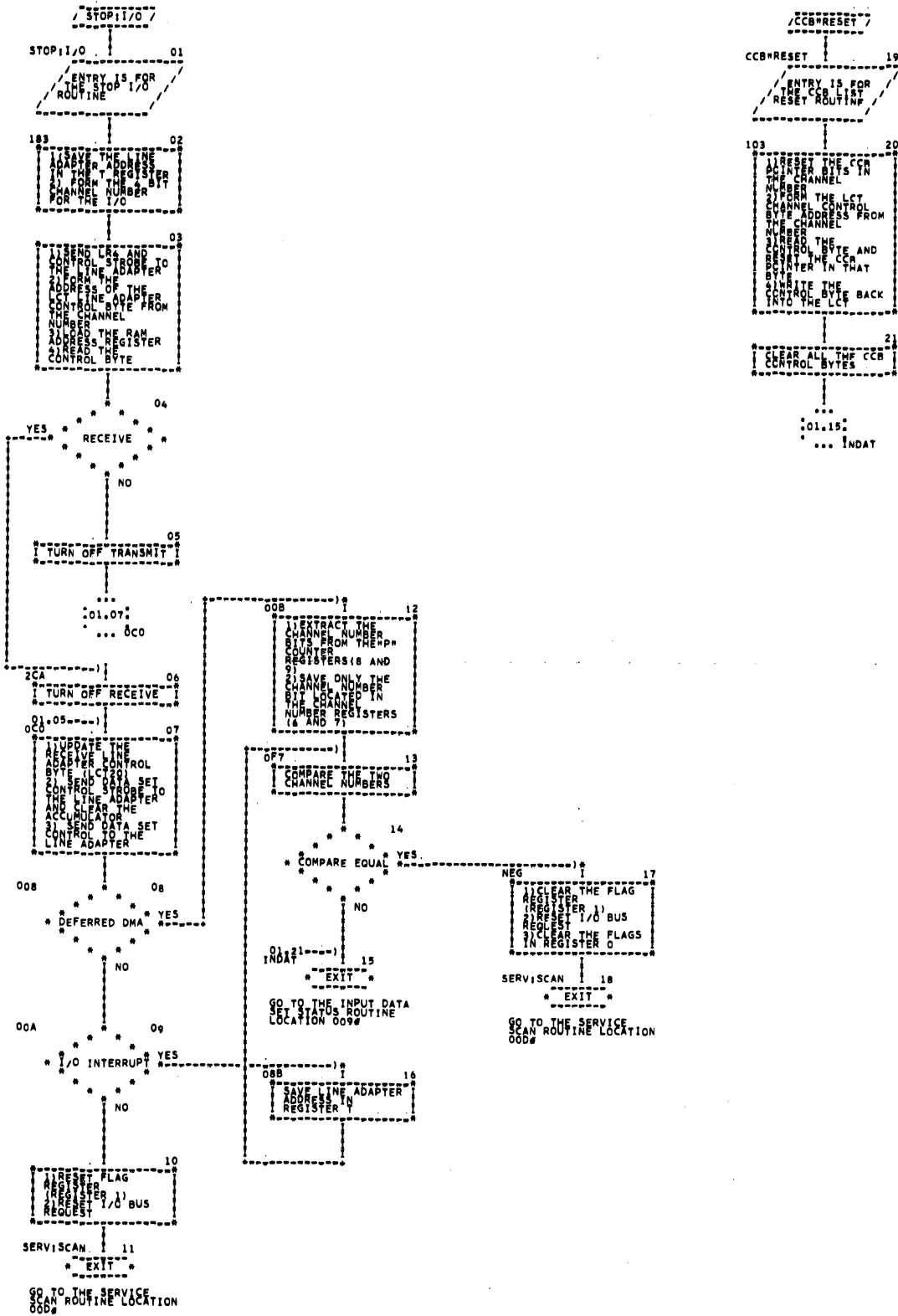


Figure 4-21 Stop I/O/CCB List Reset Routines

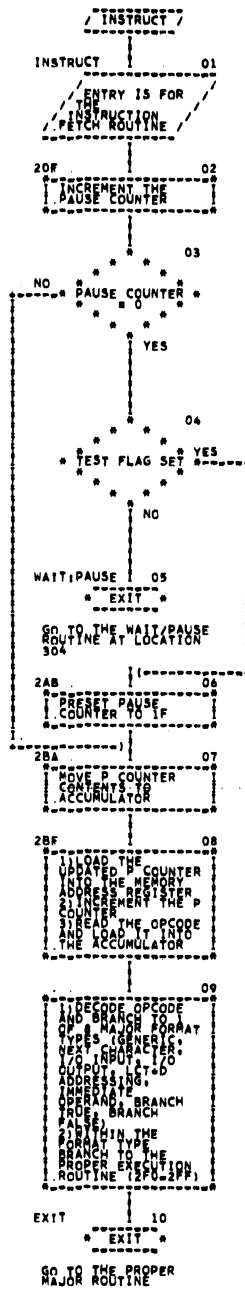


Figure 4-22 Instruction Fetch Routine

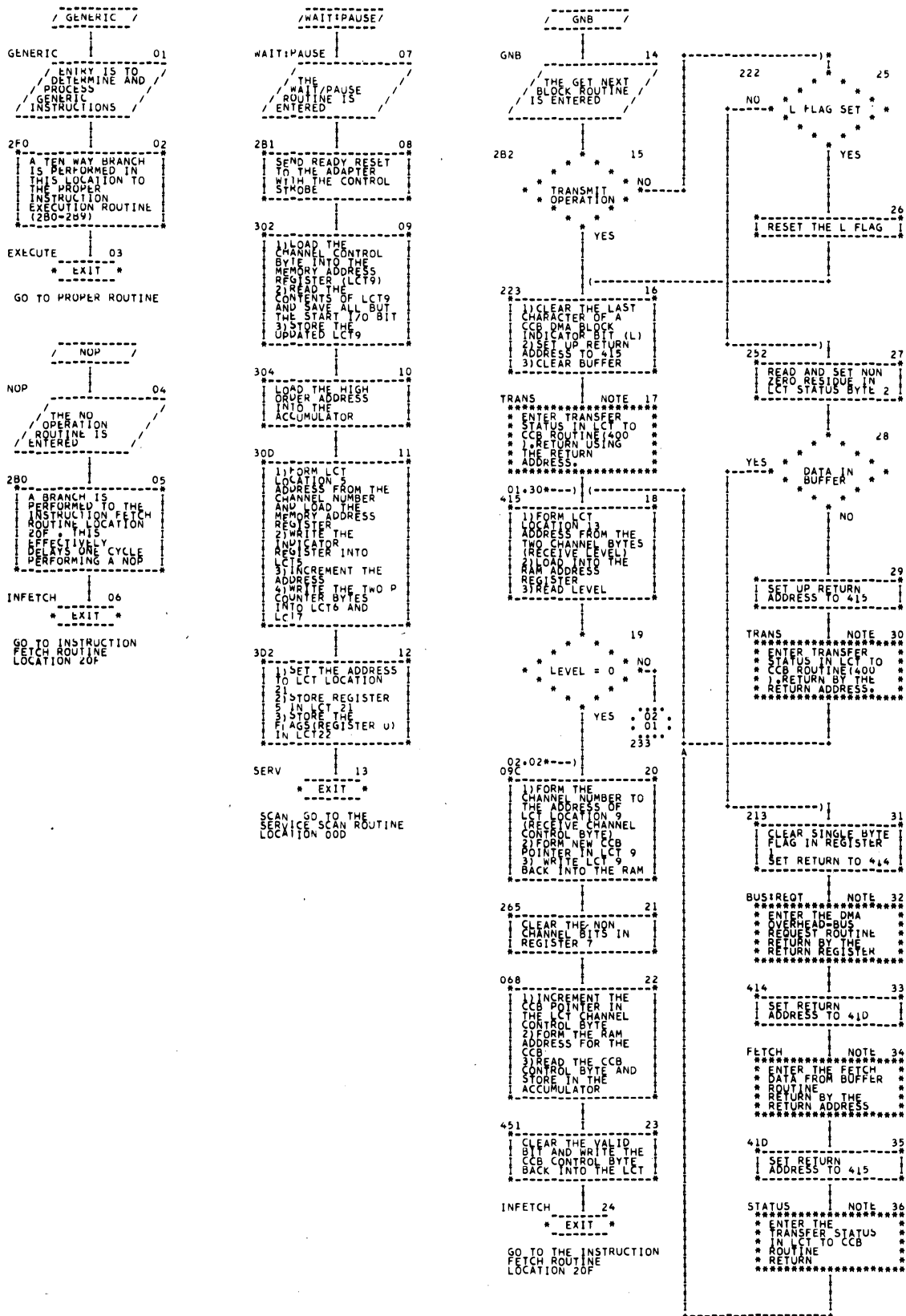


Figure 4-23 Generic Instruction Routines (Sheet 1 of 3)

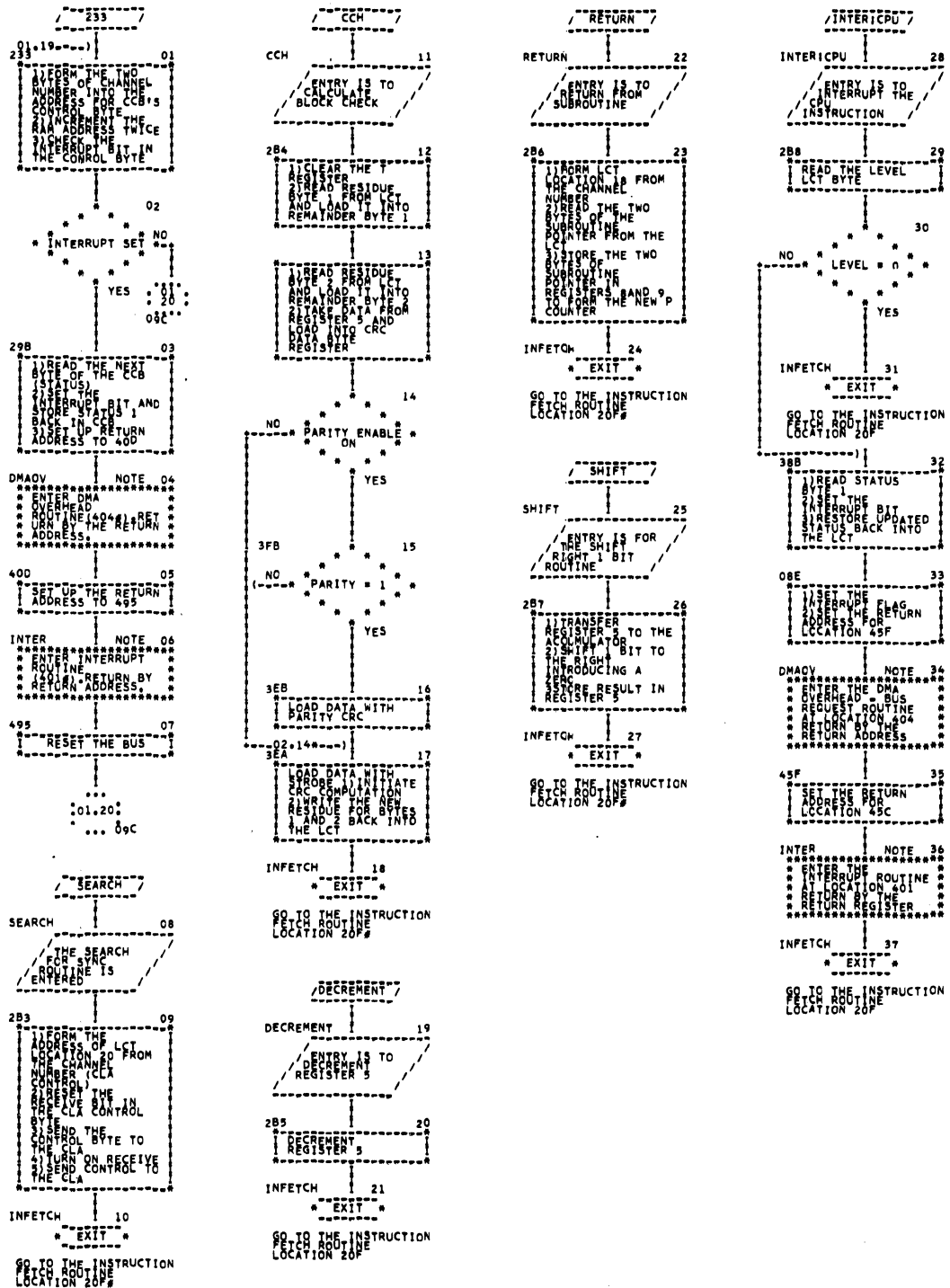


Figure 4-23 Generic Instruction Routines (Sheet 2 of 3)

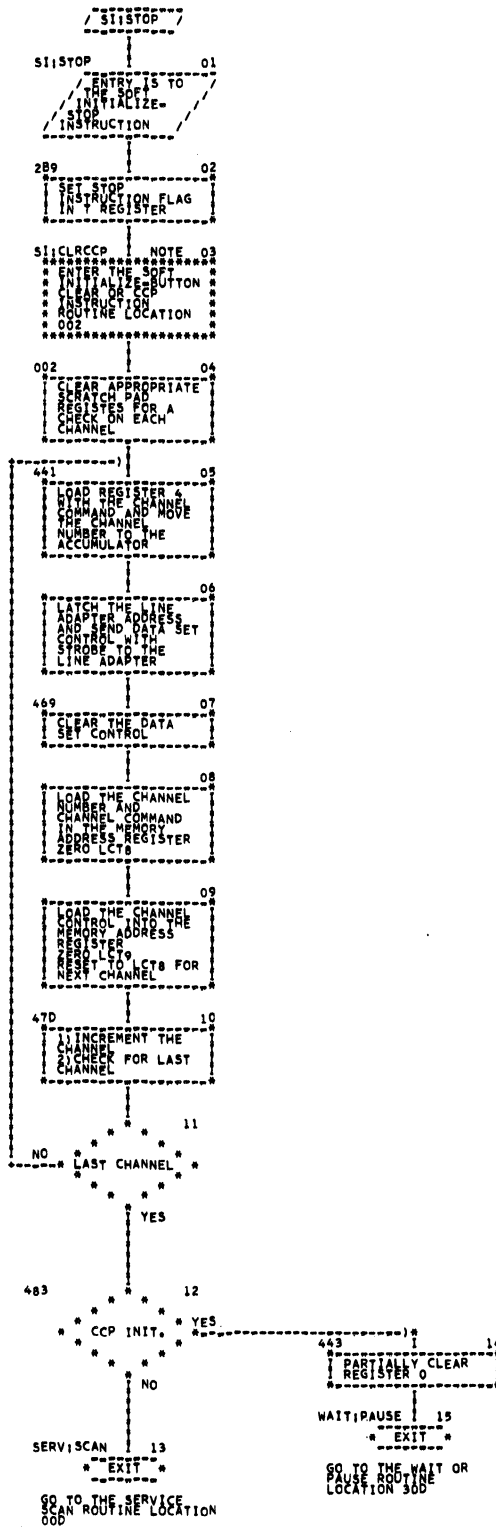


Figure 4-23 Generic Instruction Routines (Sheet 3 of 3)

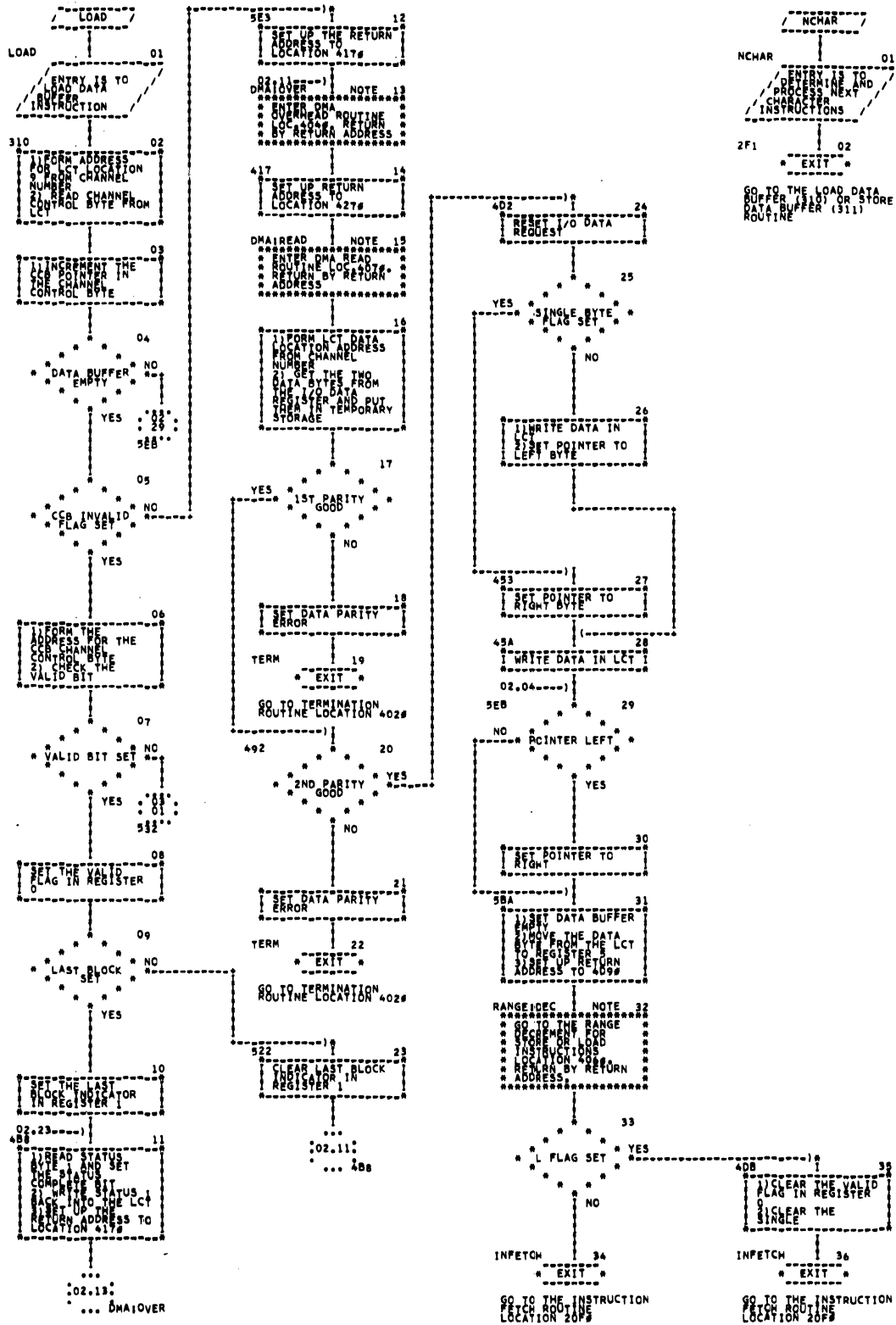


Figure 4-24 Next Character Instruction Routines (Sheet 1 of 3)

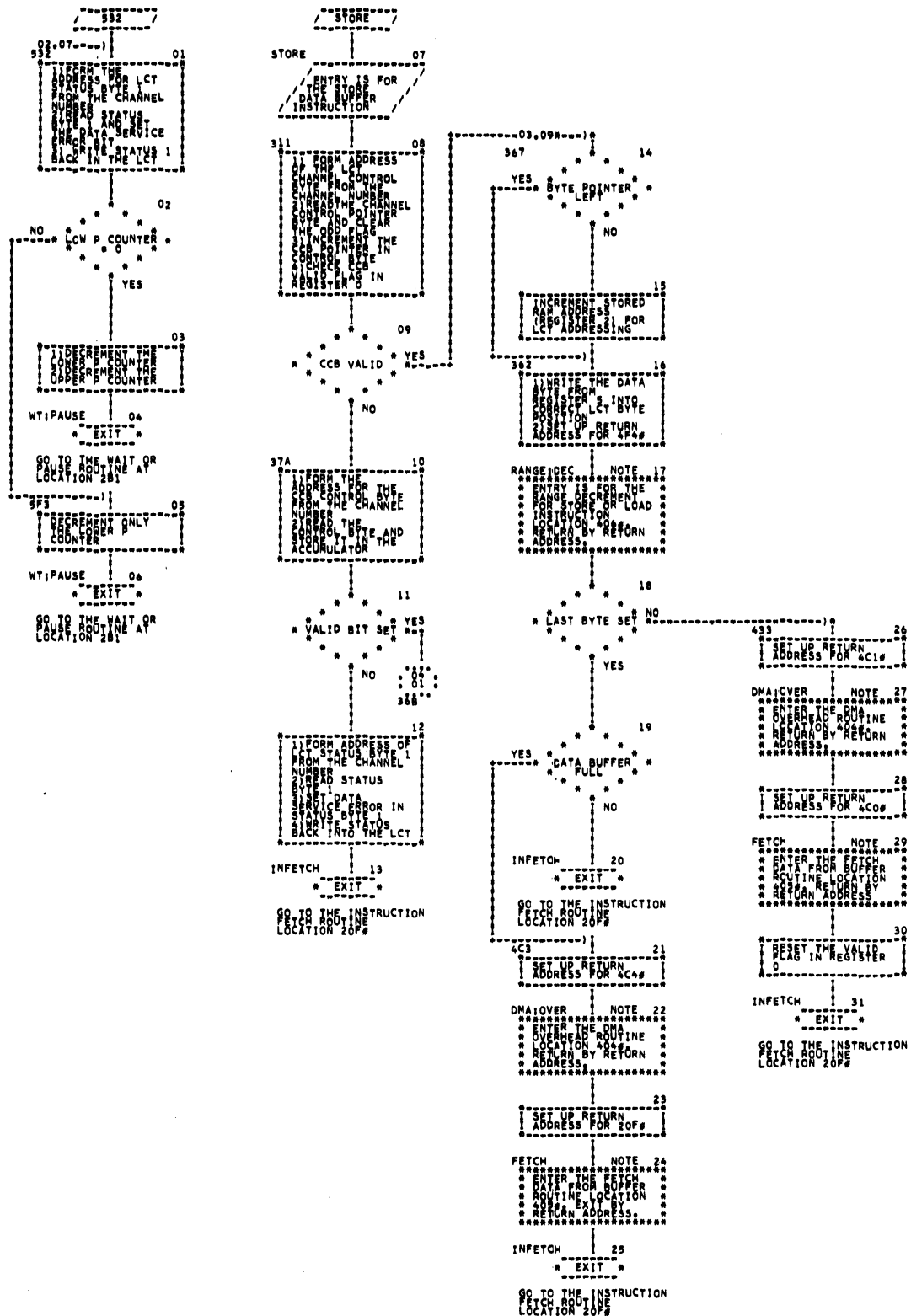


Figure 4-24 Next Character Instruction Routines (Sheet 2 of 3)

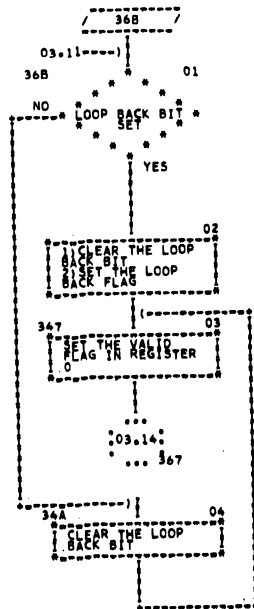


Figure 4-24 Next Character Instruction Routines
(Sheet 3 of 3)

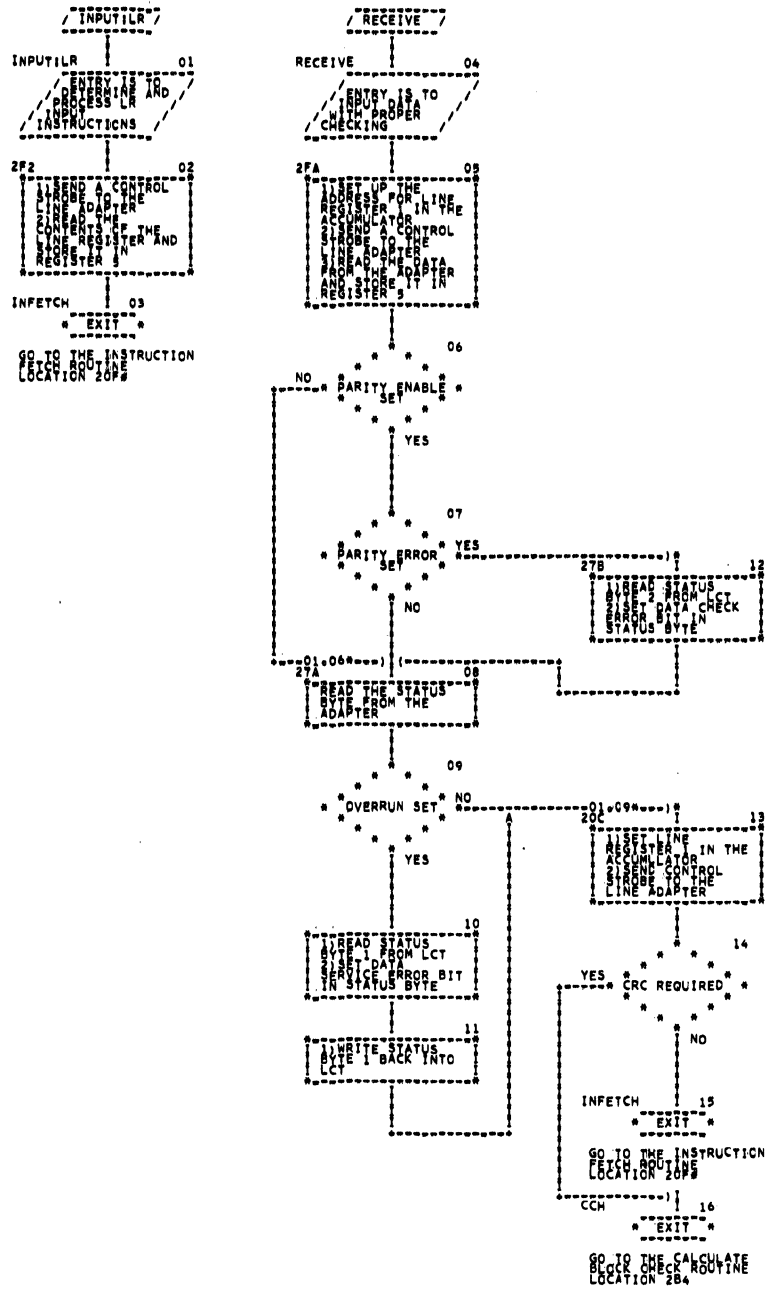


Figure 4-25 Input Line Register/Receive Routines

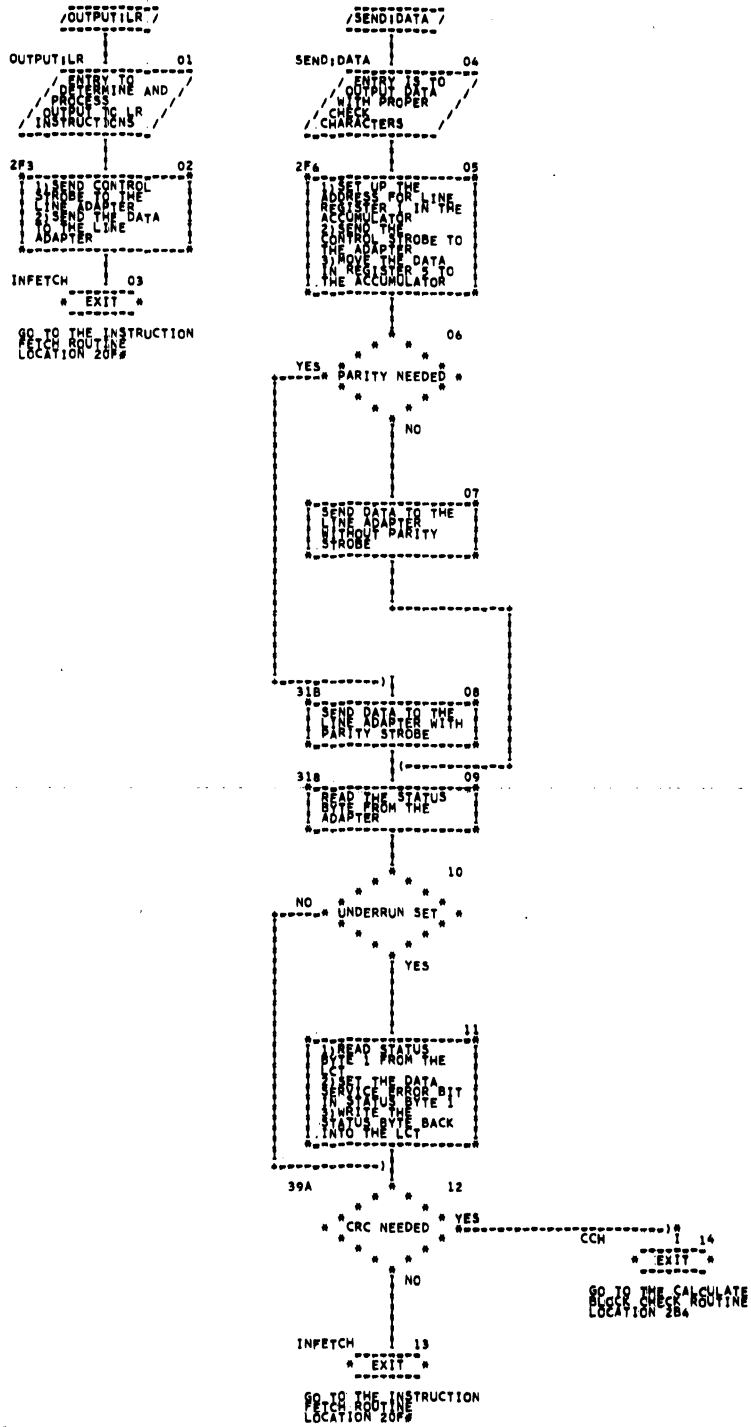


Figure 4-26 Output to Line Register/Send Routines

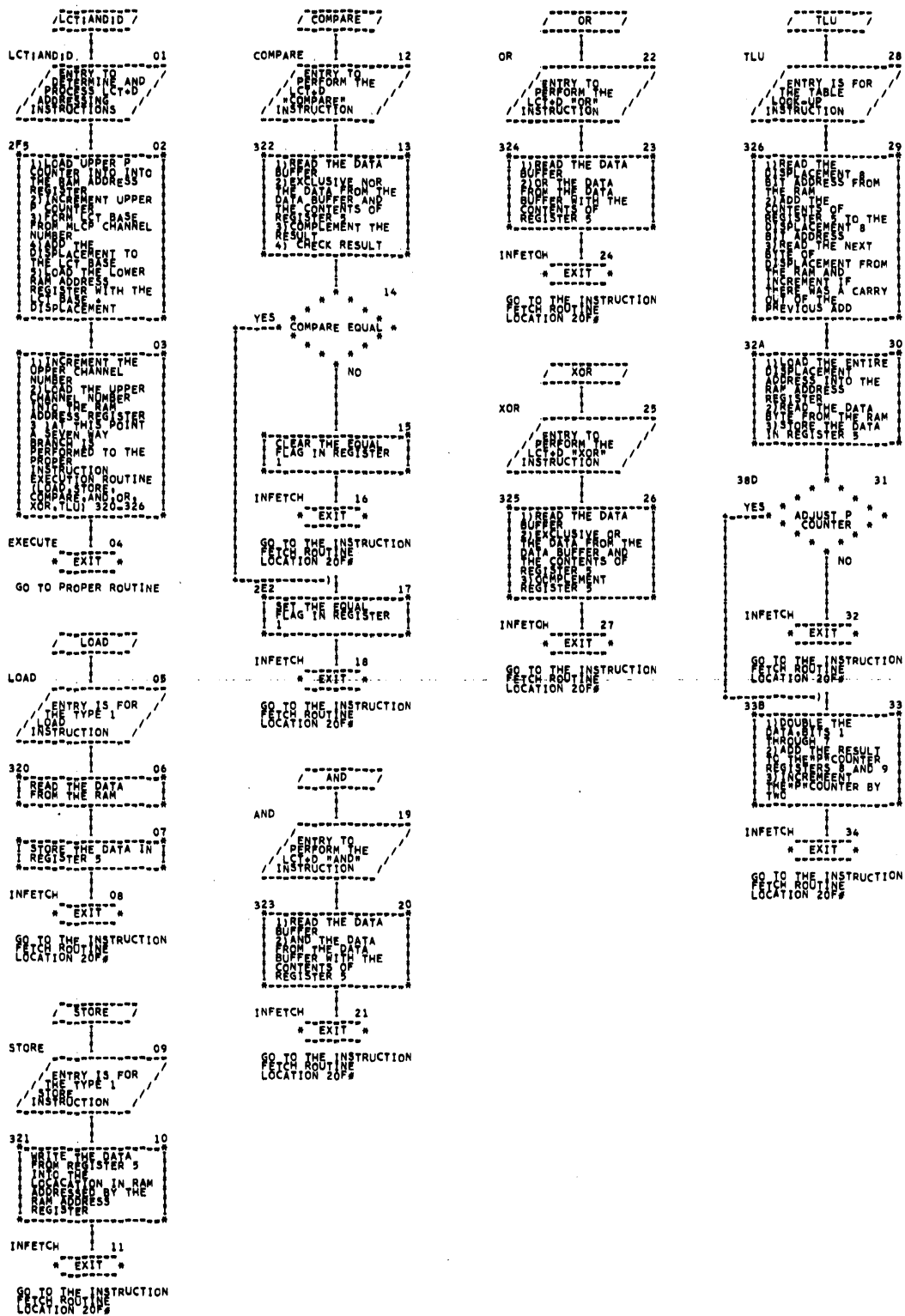


Figure 4-27 LCT and Displacement Addressing Instruction Routines

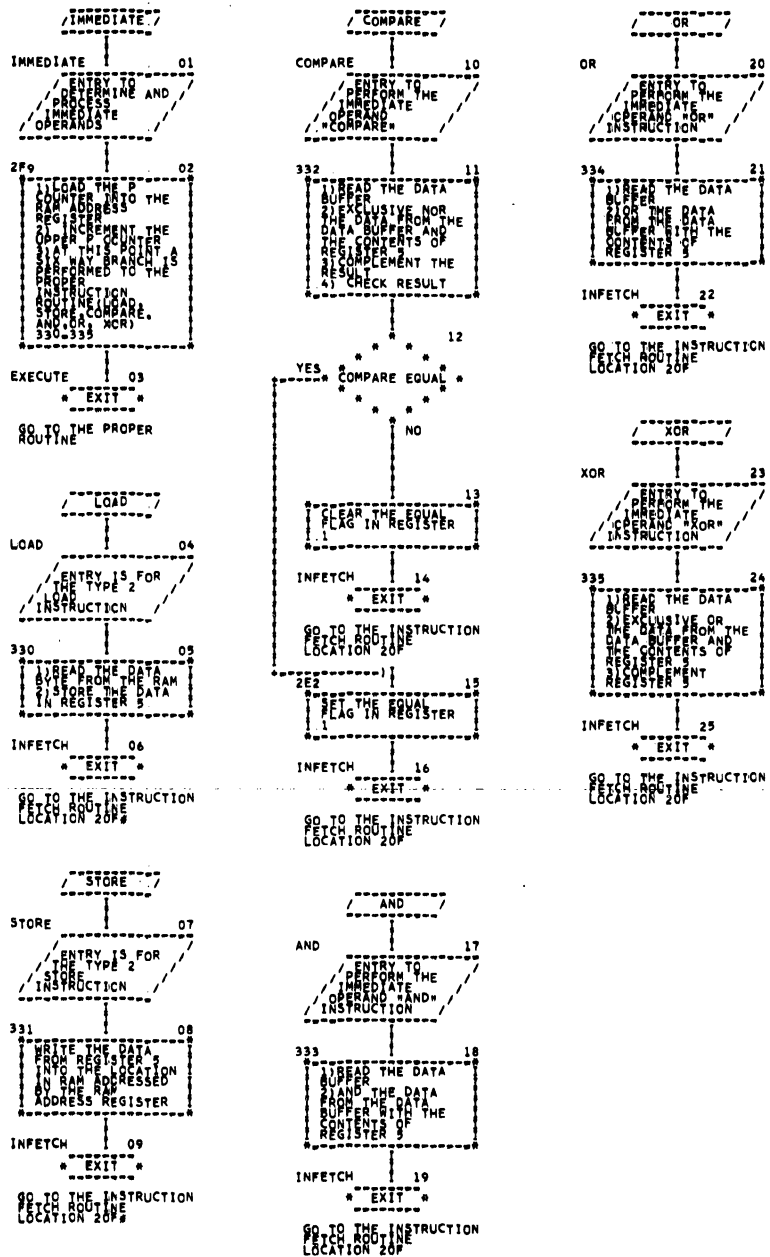


Figure 4-28 Immediate Operand Instruction Routines

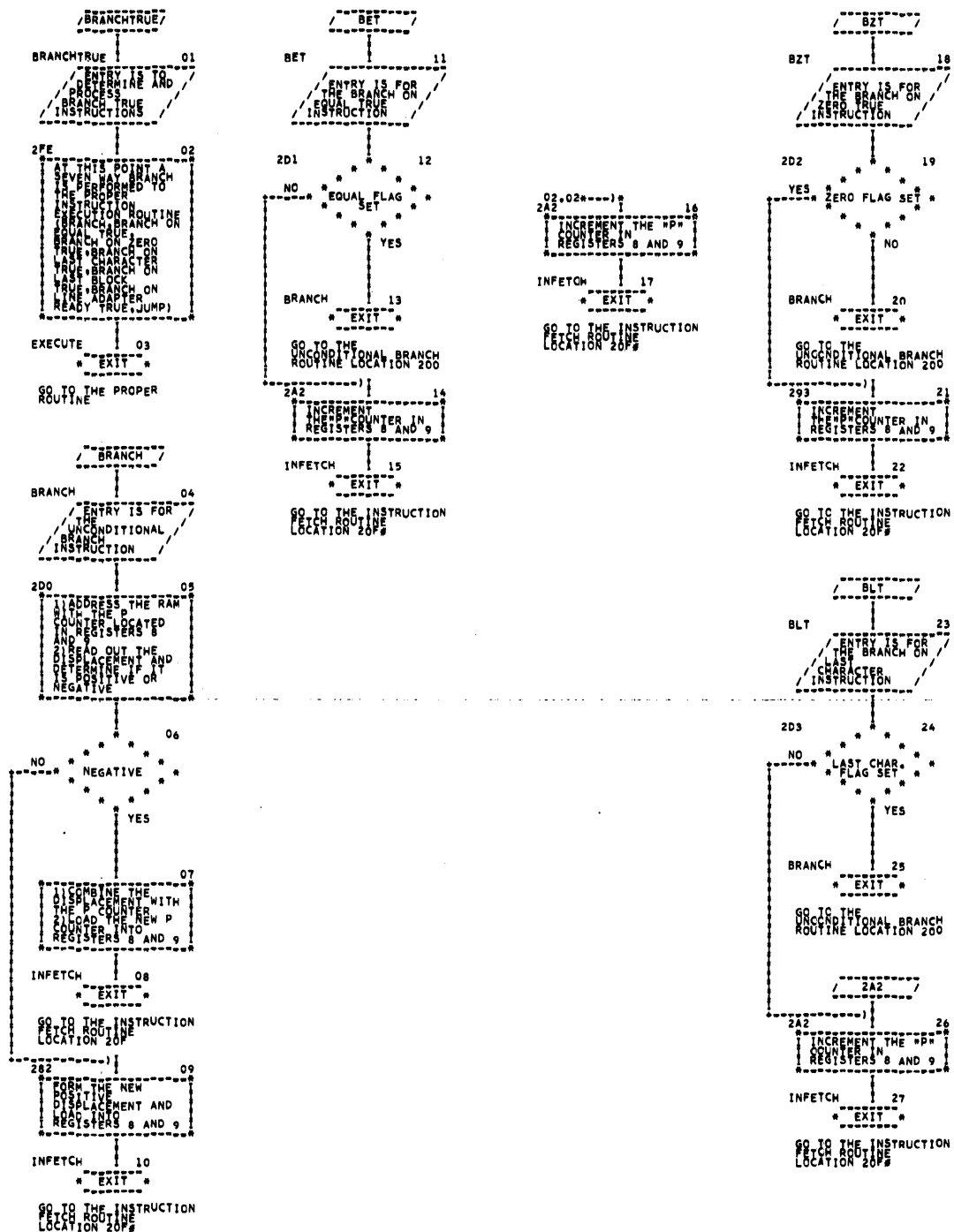


Figure 4-29 Branch True Instruction Routines (Sheet 1 of 2)

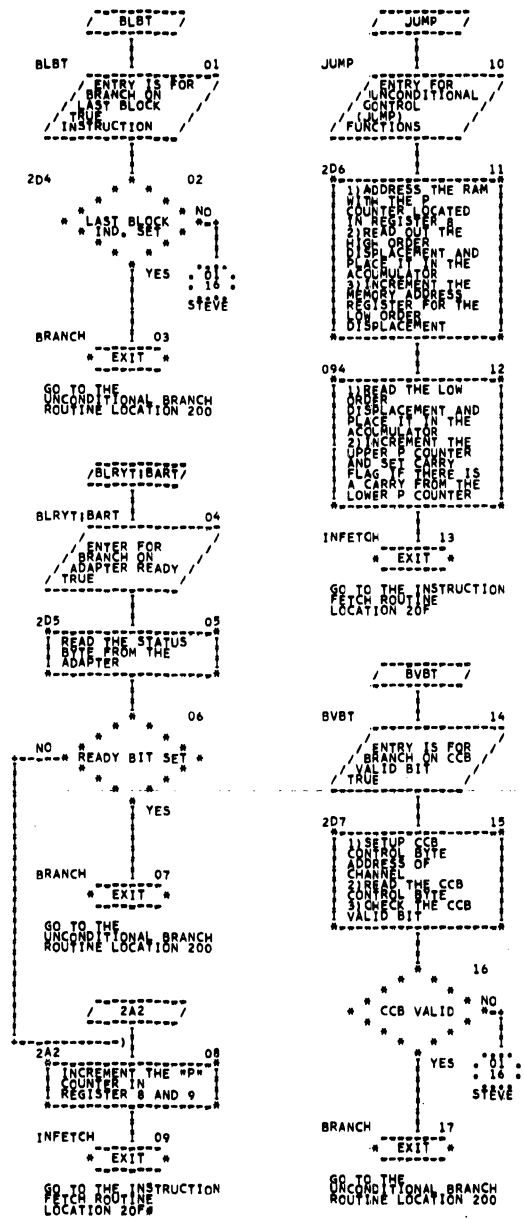


Figure 4-29 Branch True Instruction Routines
(Sheet 2 of 2)

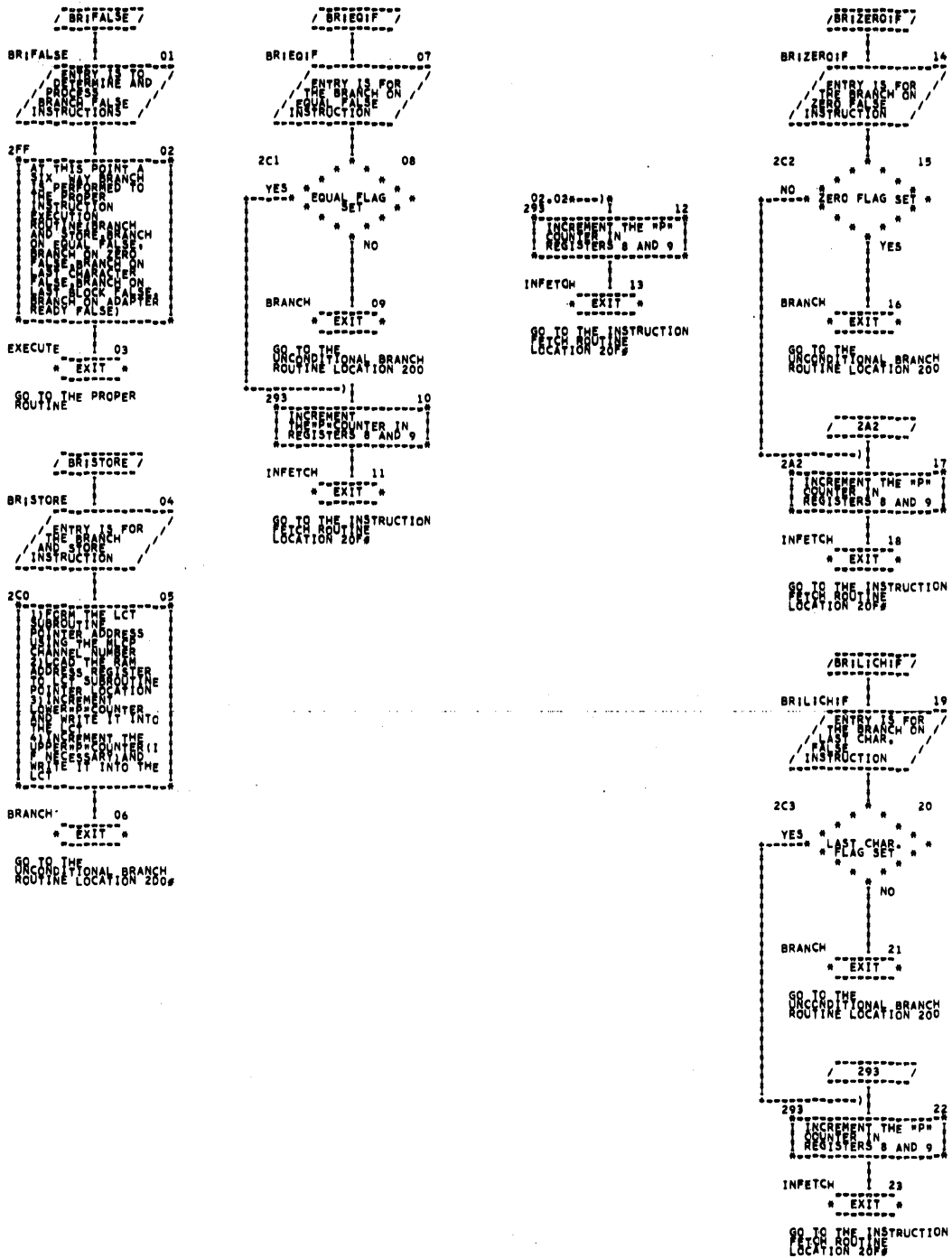


Figure 4-30 Branch False Instruction Routines (Sheet 1 of 2)

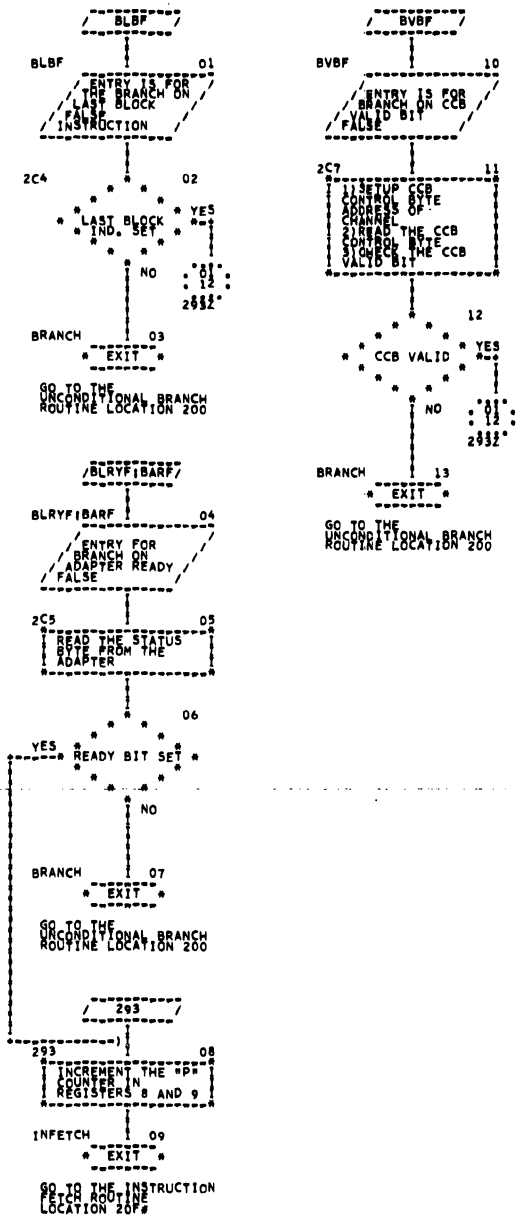


Figure 4-30 Branch False Instruction Routines (Sheet 2 of 2)

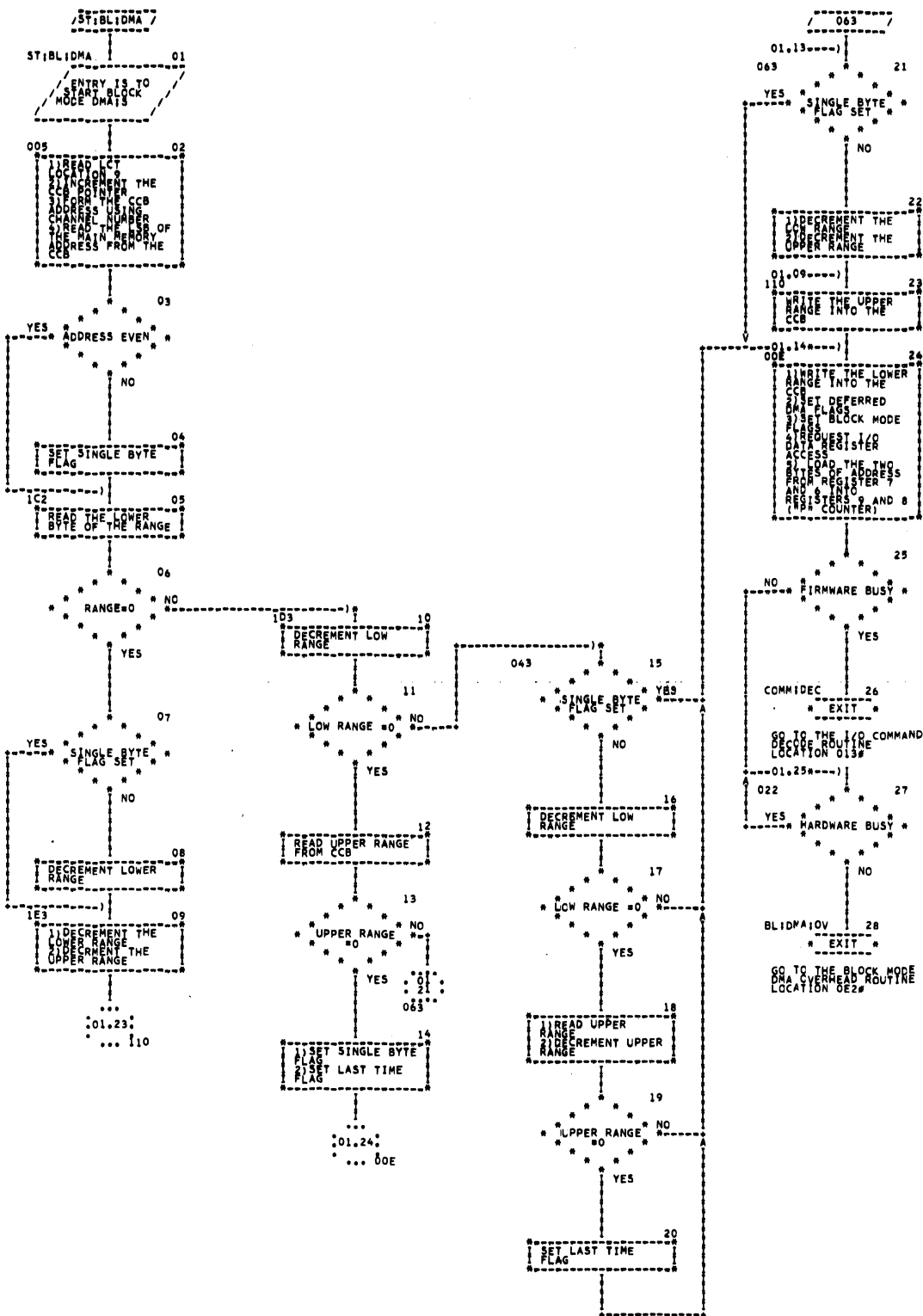


Figure 4-31 Start Block Mode DMA Routine

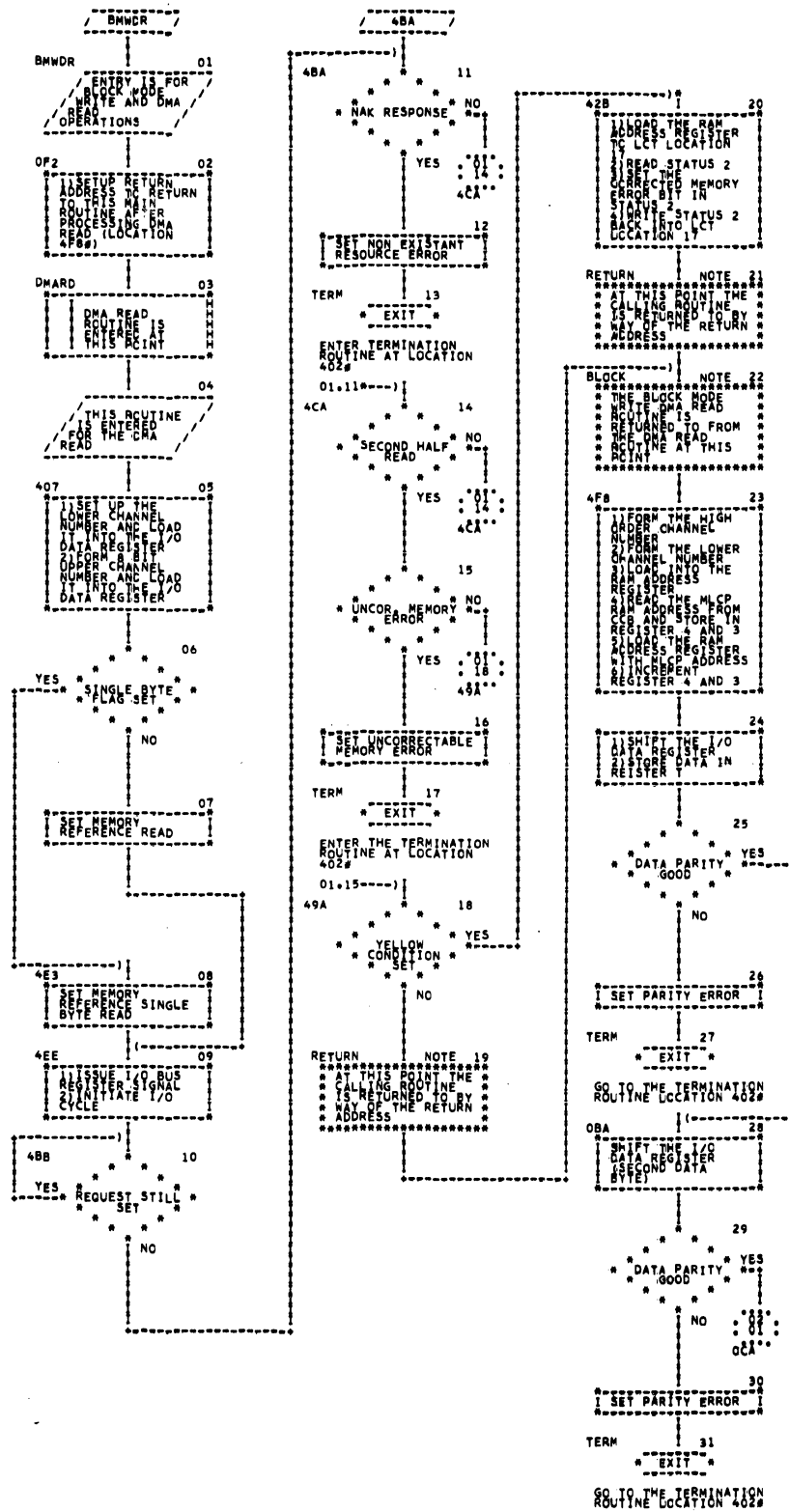


Figure 4-32 Block Mode Write DMA Read/DMA Read Routines (Sheet 1 of 2)

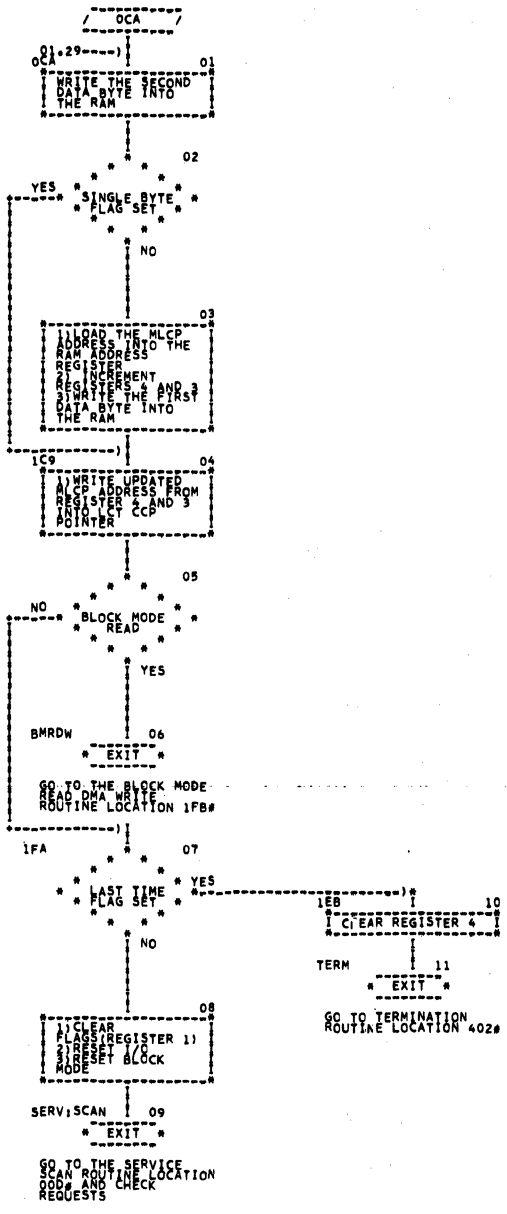


Figure 4-32 Block Mode Write DMA Read/DMA Read Routines (Sheet 2 of 2)

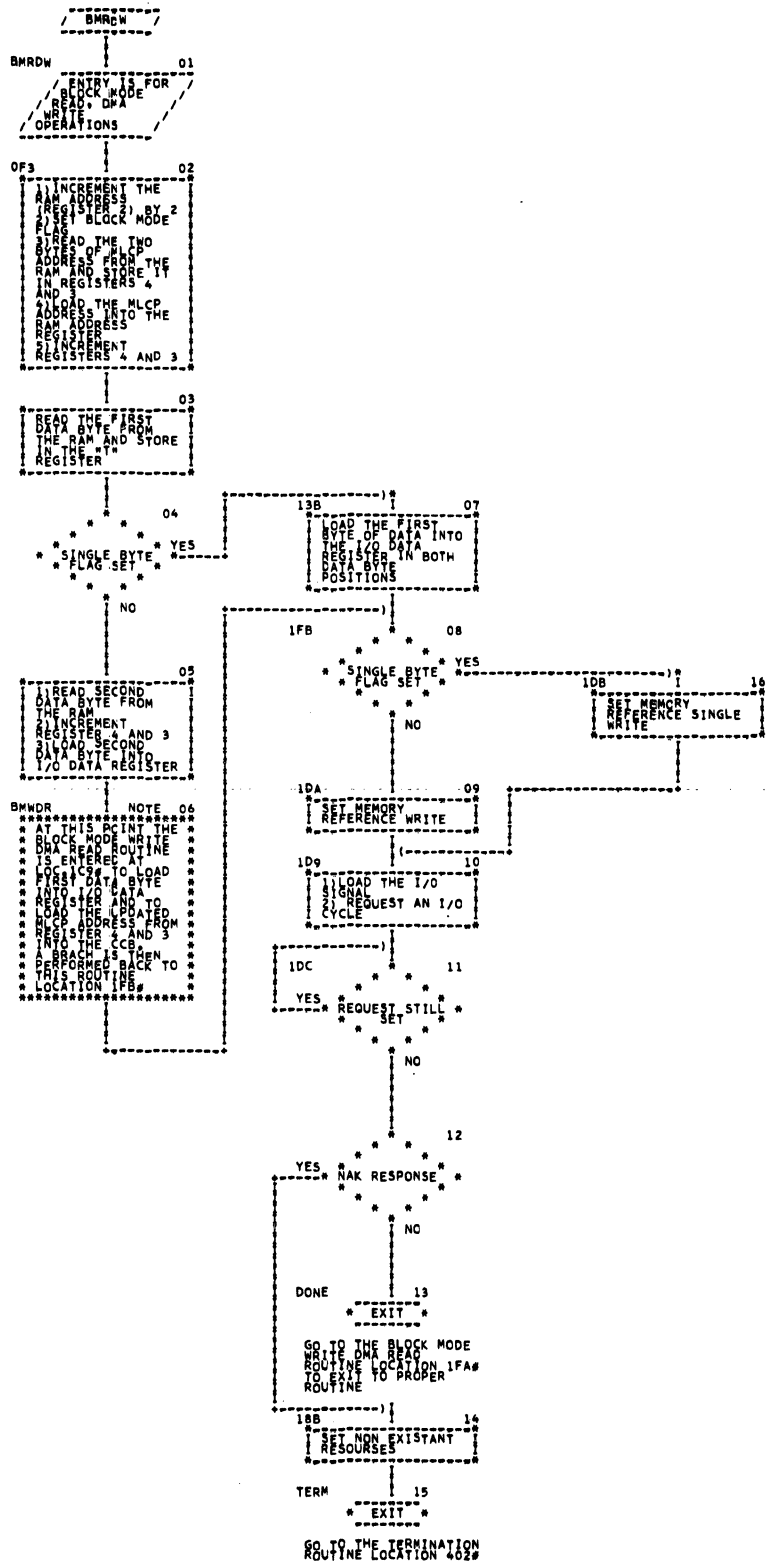


Figure 4-33 Block Mode Read DMA Write Routine

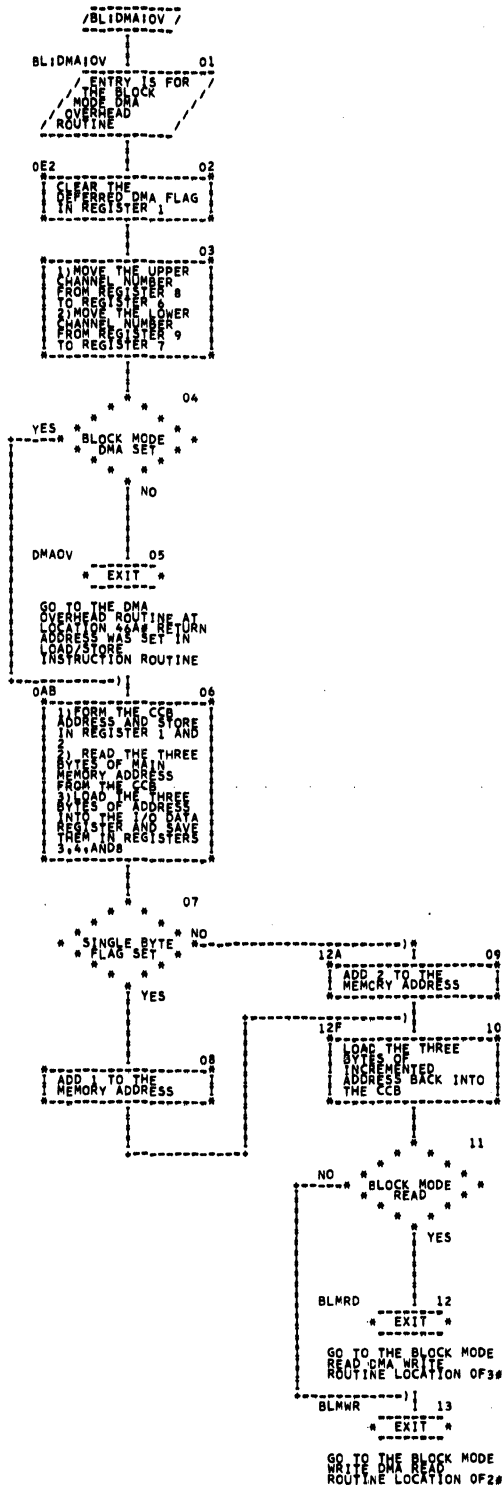


Figure 4-34 Block Mode DMA Overhead Routine

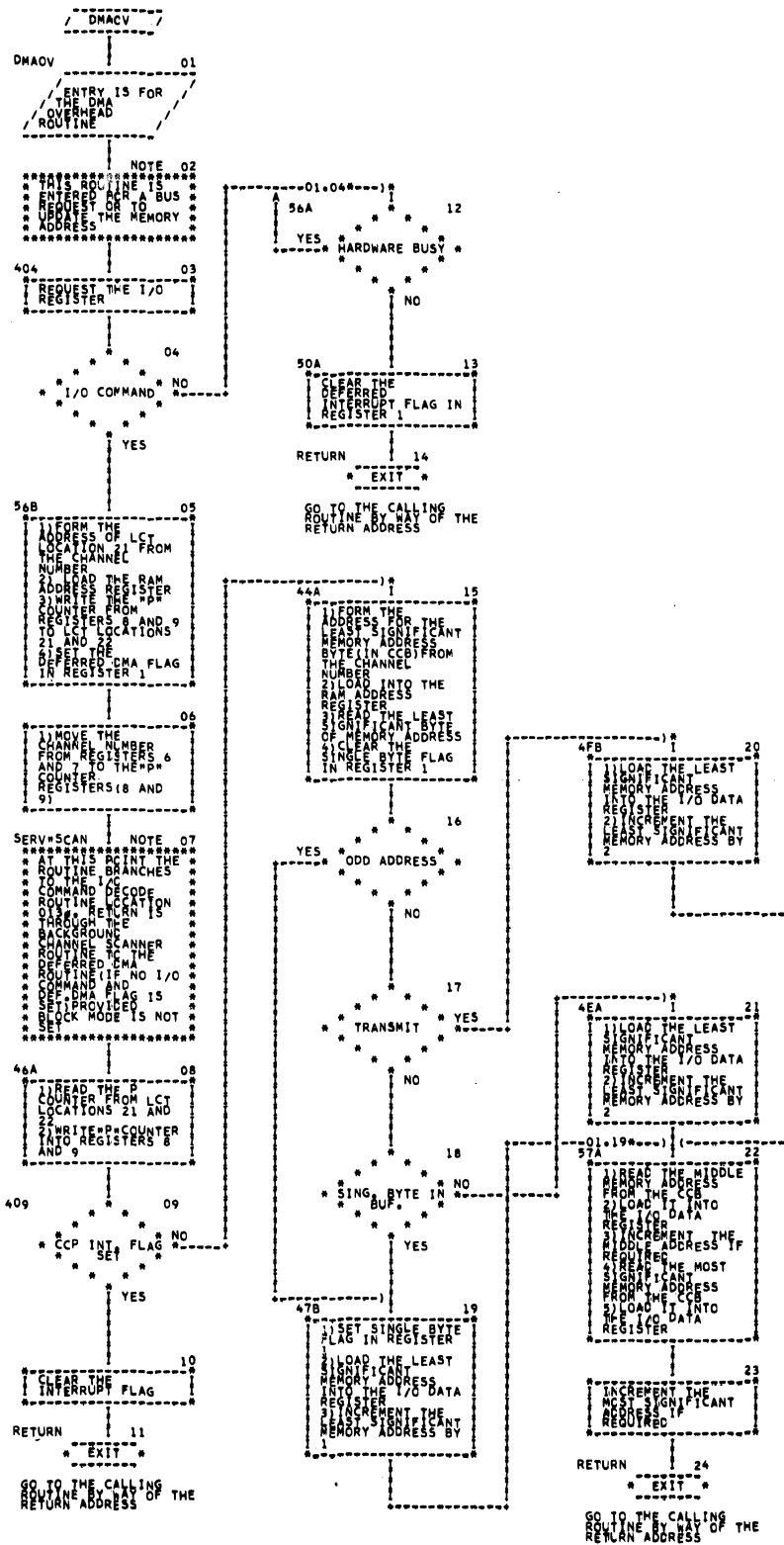


Figure 4-35 DMA Overhead Routine

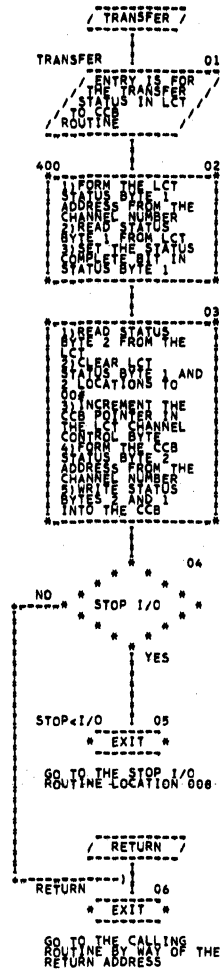


Figure 4-36 Transfer Status in LCT to CCB Routines

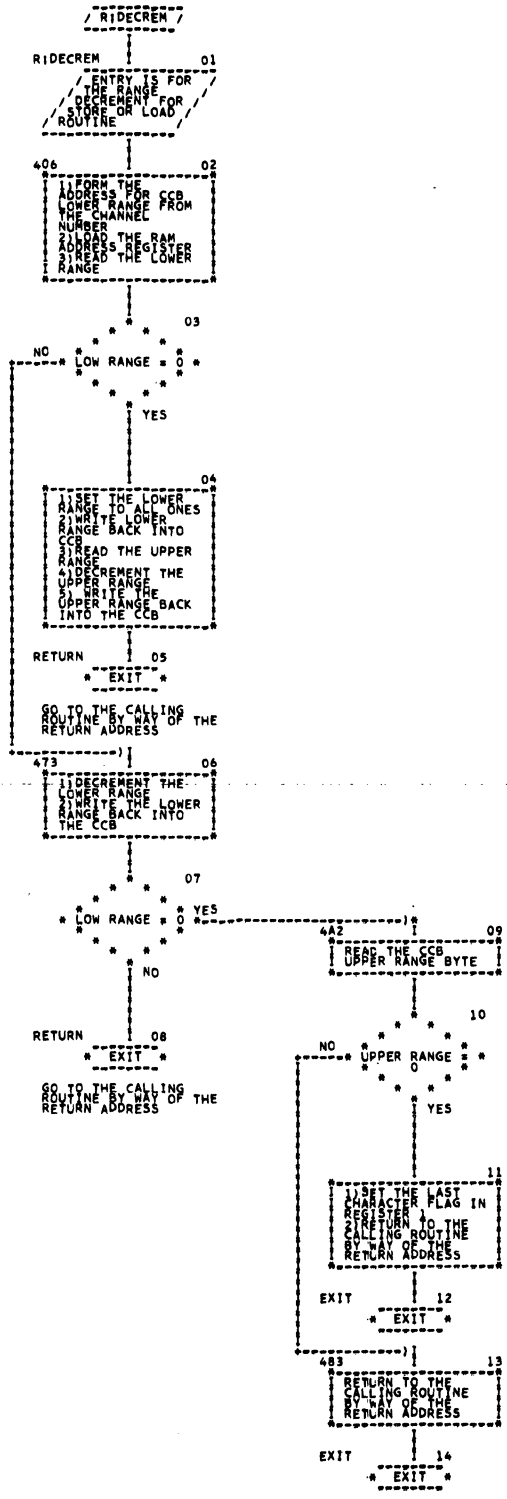


Figure 4-37 Range Decrement Routine

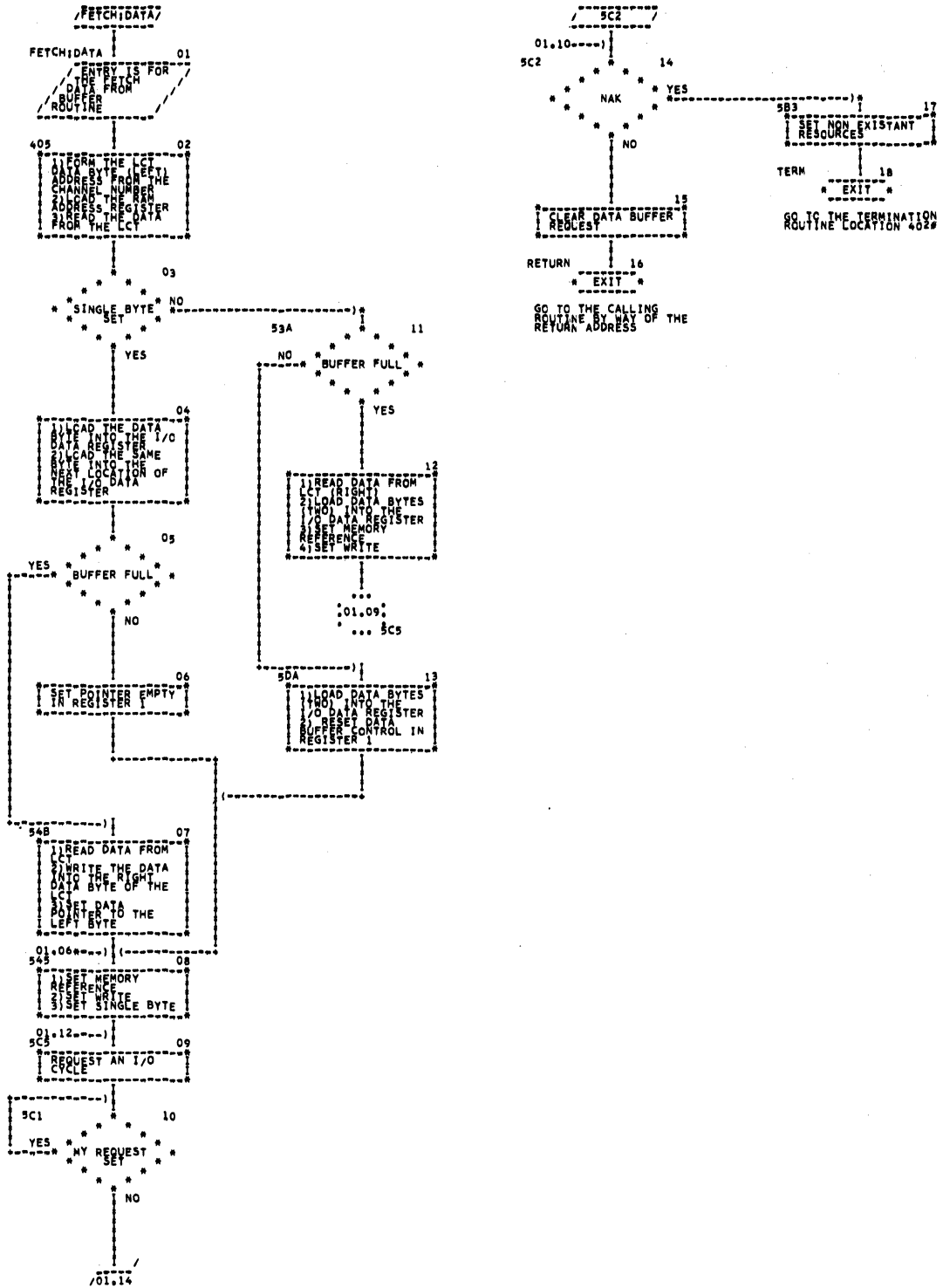


Figure 4-38 Fetch Data From Buffer Routine

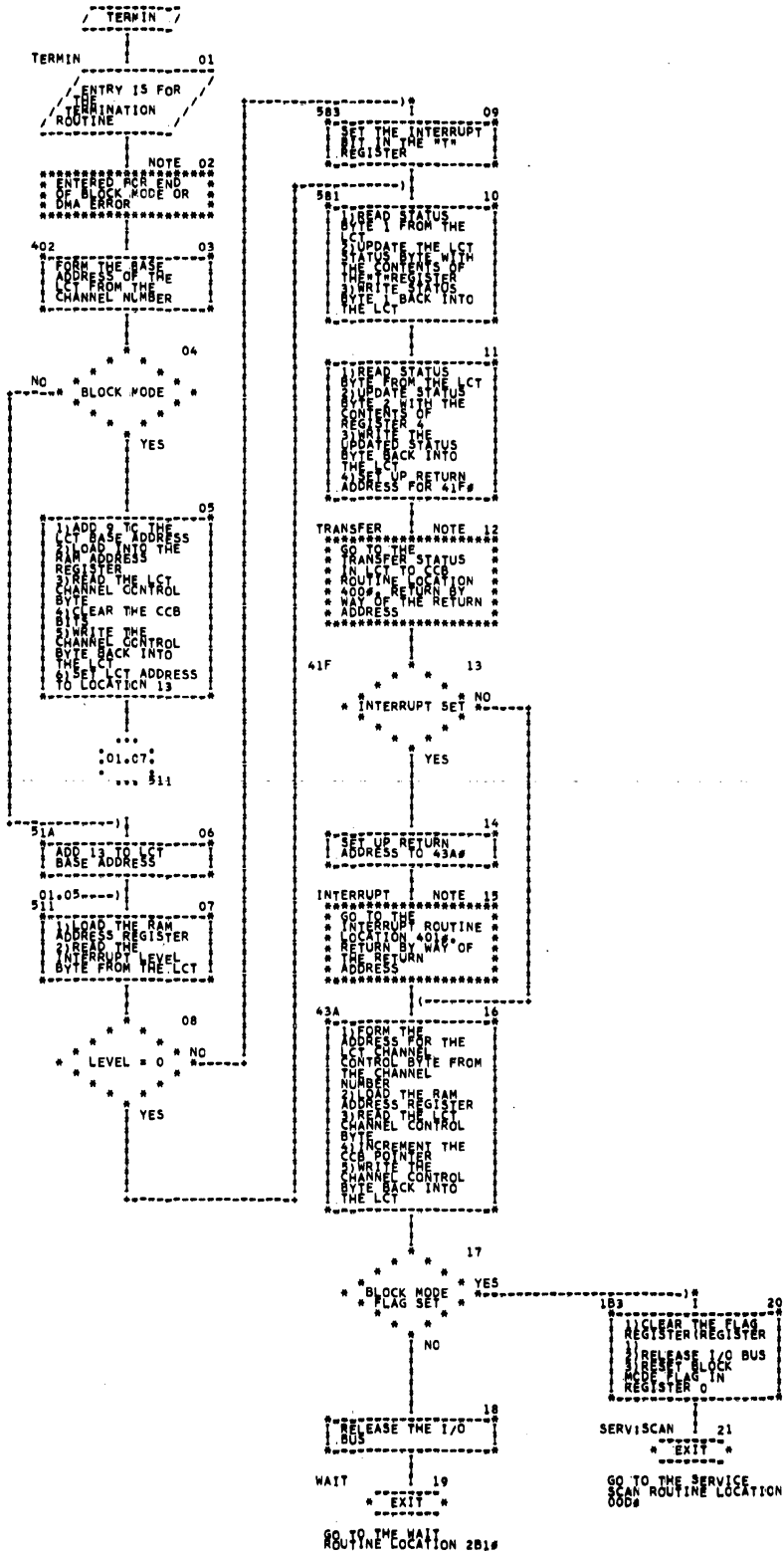


Figure 4-39 Termination Routine

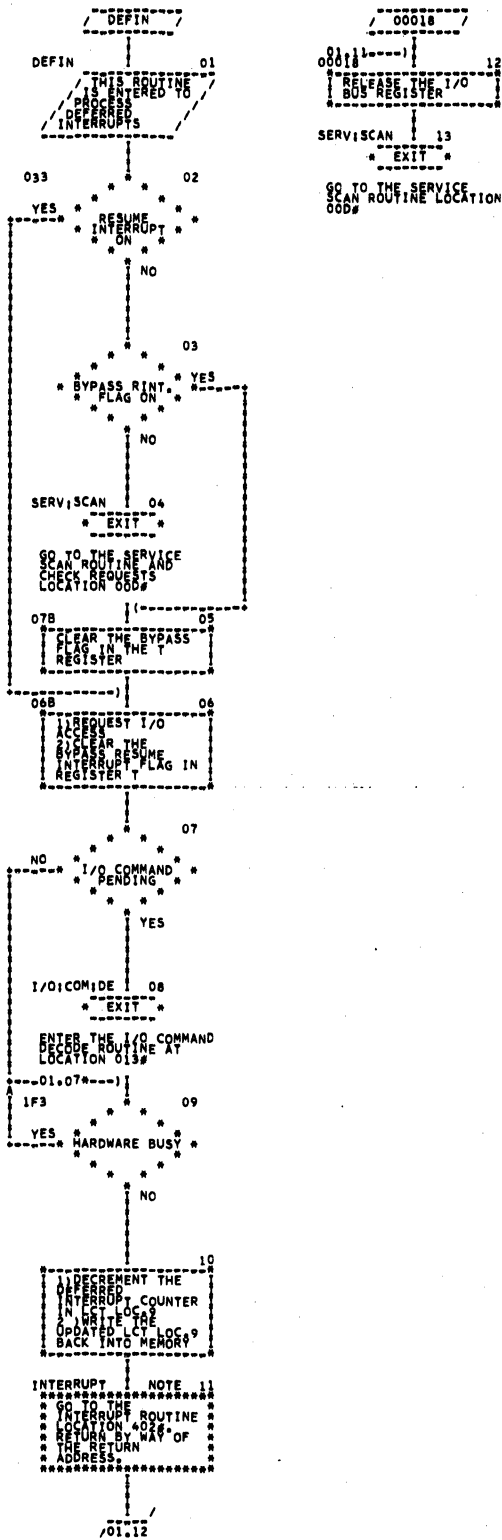


Figure 4-40 Deferred Interrupt Routine

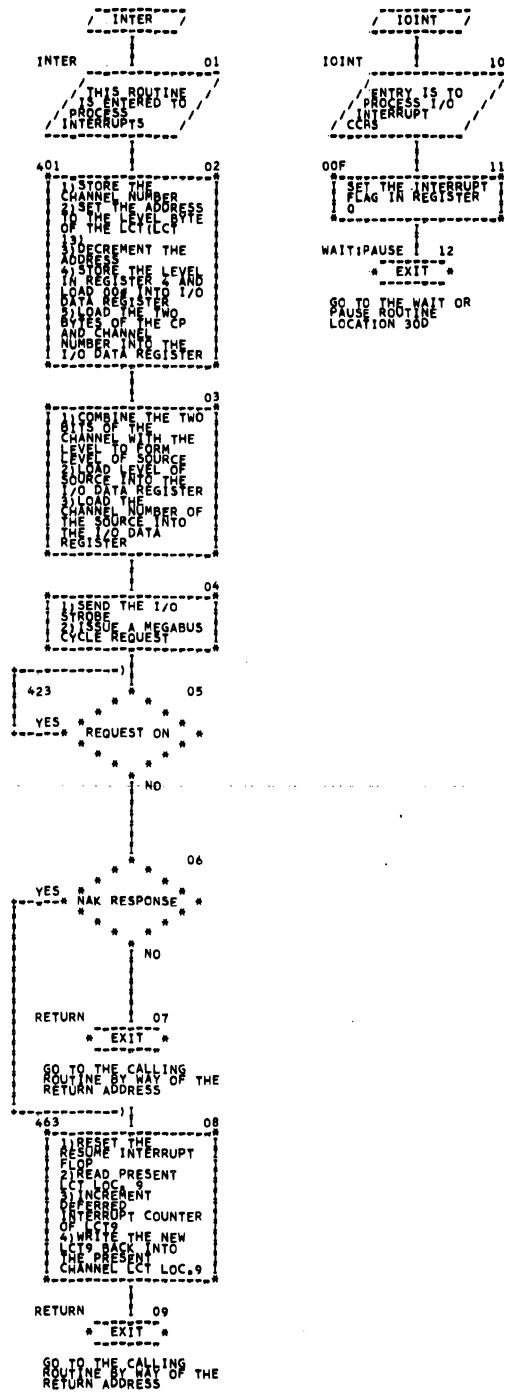


Figure 4-41 Interrupt I/O Interrupt CCP Routines

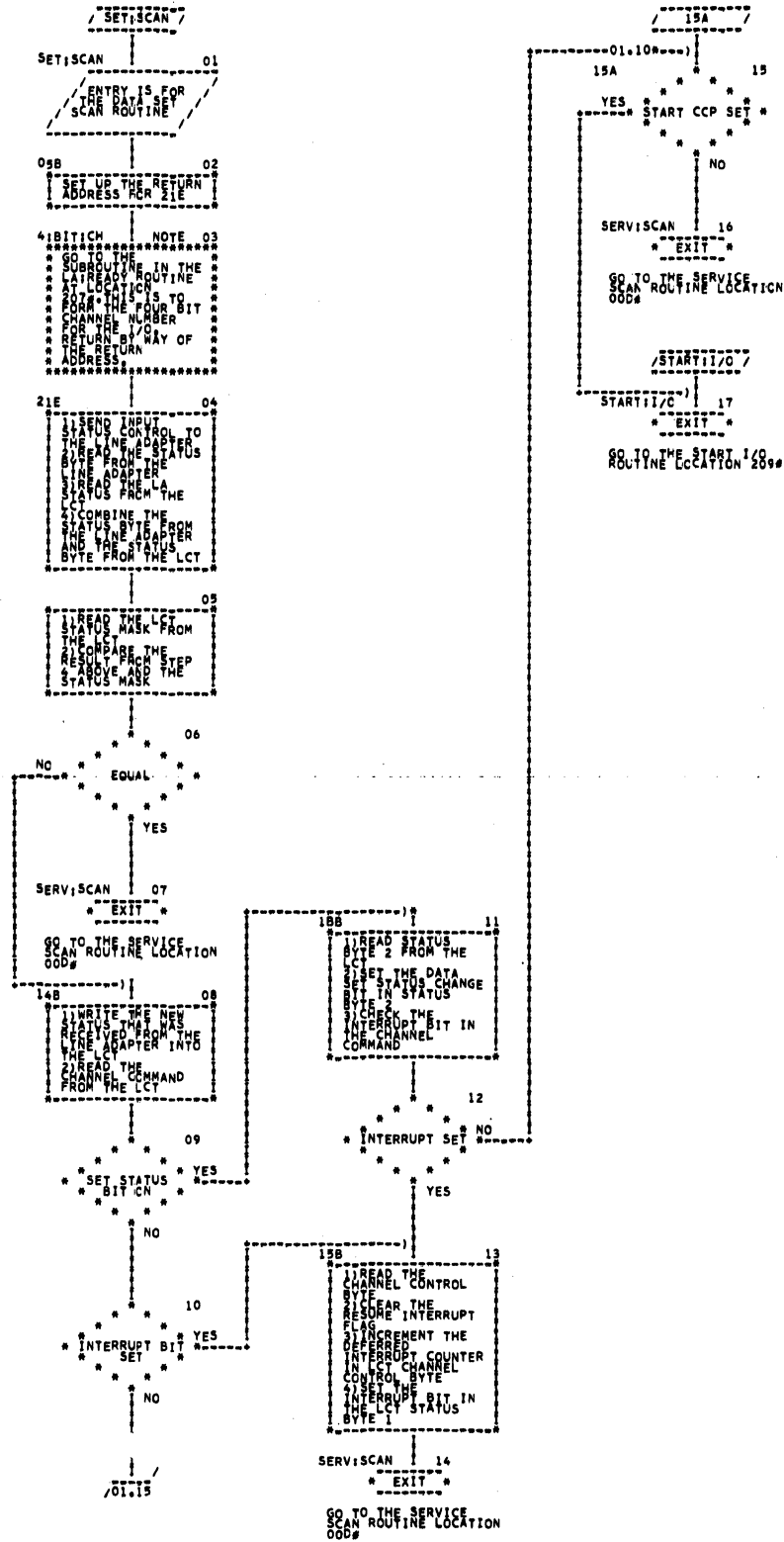


Figure 4-42 Data Set Scan Routine

PLEASE FOLD AND TAPE –
NOTE: U. S. Postal Service will not deliver stapled forms

**M&TO HARDWARE PUBLICATIONS
USER COMMENTS FORM**

DOCUMENT TITLE: _____

PART NO.: _____

ORDER NO.: _____

ERRORS:

HOW DO YOU USE THIS DOCUMENT?

THEORY _____

MAINTENANCE _____

TROUBLESHOOTING _____

OTHER: _____

DOES THIS MANUAL SATISFY YOUR REQUIREMENTS?

YES NO

IF NOT, PLEASE EXPLAIN _____

FROM: NAME _____ DATE _____

TITLE _____

COMPANY _____

ADDRESS _____

FIRST CLASS
Permit No. 39531
Waltham, Ma.

Business Reply Mail

NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY

HONEYWELL INFORMATION SYSTEMS INC.
200 SMITH STREET
WALTHAM, MA. 02154

MAIL STATION 872A
HARDWARE PUBLICATIONS, BILLERICA

Honeywell

PLEASE FOLD AND TAPE –
NOTE: U. S. Postal Service will not deliver stapled forms

M&TO HARDWARE PUBLICATIONS
USER COMMENTS FORM

DOCUMENT TITLE: _____

PART NO.: _____

ORDER NO.: _____

ERRORS:

HOW DO YOU USE THIS DOCUMENT?

THEORY _____

MAINTENANCE _____

TROUBLESHOOTING _____

OTHER: _____

DOES THIS MANUAL SATISFY YOUR REQUIREMENTS?

YES NO

IF NOT, PLEASE EXPLAIN _____

FROM: NAME _____ DATE _____

TITLE _____

COMPANY _____

ADDRESS _____

FIRST CLASS
Permit No. 39531
Waltham, Ma.



Business Reply Mail

NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES



POSTAGE WILL BE PAID BY

HONEYWELL INFORMATION SYSTEMS INC.
200 SMITH STREET
WALTHAM, MA. 02154

MAIL STATION 872A
HARDWARE PUBLICATIONS, BILLERICA

Honeywell

C

3

C

11
2

O

The Other Computer Company:
Honeywell

HONEYWELL INFORMATION SYSTEMS

In the U.S.A.: 200 Smith Street, MS 061, Waltham, Massachusetts 02154
In Canada: 2025 Sheppard Avenue East, Willowdale, Ontario

FL48, Rev. 1