DATAPOINT CORPORATION

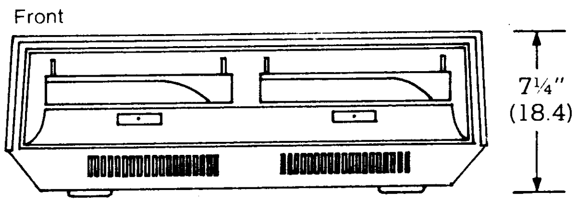
The leader in dispersed data processing ™

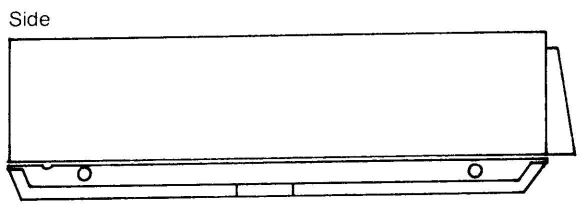
Product specification and hardware reference manual

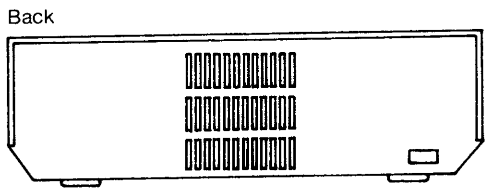
Datapoint 1800

Datapoint 1800 System

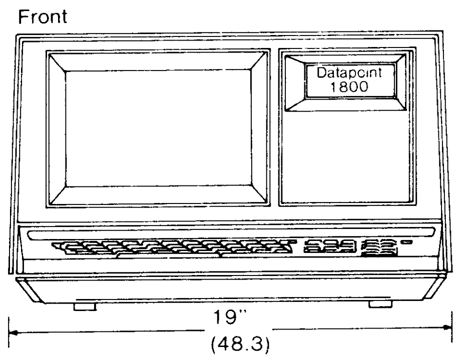
Diskette Module

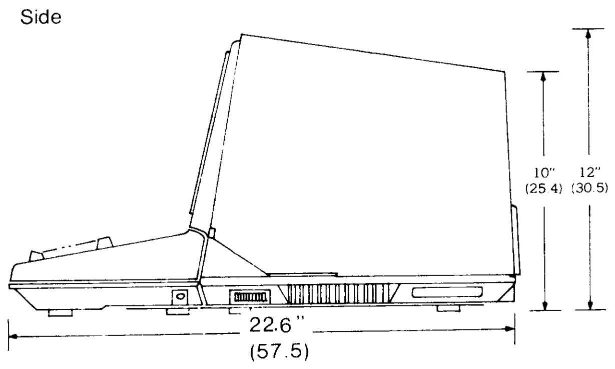


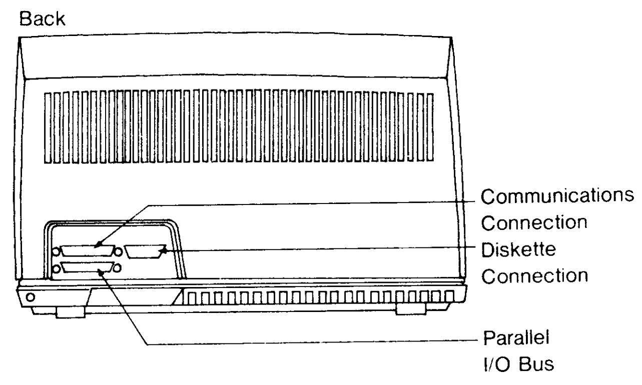




1800 Processor







June 1979
Document No. 60523
Copyright© 1979 Datapoint Corporation. Printed in U.S.A.

The "D" logo and Datapoint Corporation are trademarks of Datapoint Corporation, registered in the U.S. Patent Office.
"The leader in dispersed data processing" is a trademark of Datapoint Corporation.

PREFACE

The computer-oriented user will find this manual useful for evaluation of Datapoint 1800 system capabilities and limitations. However, only the hardware considerations are covered in this manual. The full utility of the Datapoint 1800 system cannot be appreciated until the available software support for the machine has been reviewed.

A complete family of software packages available for the Datapoint 1800 system includes high-level languages, operating systems, source code and text editors, communications programs, utility programs, etc. Reference should be made to the latest issue of the Datapoint Software Catalog for more complete information.

TABLE OF CONTENTS

PART 1
GENERAL FEATURES

1.1 Introduction

The Datapoint 1800 is a stand alone processor/diskette system. The unit provides a CRT, Keyboard, 8 bit high-speed bipolar processor, 64K of memory (4K-ROM and 60K RAM), and SDLC, BISYNC, GENSYNC, and ASYNC communications.

1.2 System Elements

There are four basic elements in the 1800 system plus the capability to interface to a number of external peripheral devices.

This chapter introduces the basic elements: CRT display, keyboard, processor and diskette storage drives. Further information may be obtained from succeeding chapters.

1.3 CRT Display

The CRT Display provides the following features:

a. 1920 characters;
b. 80-character by 24-line format;
c. Software-defined 128-character font;
d. 60 frames-per-second refresh rate (50 f.p.s. when using 50 Hertz power);
e. 5 x 7 matrix character generation;
f. 5 x 7 solid, blinking cursor, alternates with characters, nondestructive;
g. Direct processor control of CRT Display line by line;
h. Inverse video, character by character;
i. Split screen, blink, and page mode capability;

1.4 Keyboard

The removable keyboard, which may be placed up to 1 meter away from the CRT Display, provides a basic 55-key alphanumeric group, an 11-key numeric group and 10 programmable system control keys.

The keyboard provides a unique multikey roll-over characteristic giving maximum ease of typing. The control processor itself performs key scanning. An audible click and a variable-pitch "beep" for acoustic cues to the operator are available under program or software control.

1.5 Processor

The integral processor provides all control functions and includes:

* 8-bit memory word length (plus parity)
* 60K RAM memory
* "Pipelined" control processor
* Memory includes ROM for system functions
* Internal Communications module
* Internal Automatic Calling Unit interface
* 5500/6600 compatible I/O Bus
* Implements 5500 USER MODE instructions

The instruction set contains 5500 USER MODE instructions plus selected USER MODE instructions from the 6600. In addition, the 1800 processor characteristics provide:

* Direct control of flexible disk independent of 5500/6600-compatible I/O channel
* Memory, sector-table, and I/O parity checking
* Memory protection through sector table
* Based and sectored memory addressing

1.6 Memory

The memory module contains both random access memory (RAM) for program storage, and read only memory (ROM) for system functions such as initialization, Boot Block, memory test, keyboard and display routines, diskette routines, a debug program, and various test routines.

1.7 Diskette Drive System

A dual-drive diskette system capable of operation in both single and double-density modes is provided for program and data storage. Each drive accepts only qualified, double density diskette media, and provides:

a. .256 Megabyte storage in single density mode;
b. .512 Megabyte storage in dual density mode;
c. Direct processor controlled random access and data transfer.

One dual-drive diskette module provides a total of 1.024 Megabyte storage; an extension module is available to increase this to 2.048 Megabytes.

1.8 General Specifications

POWER REQUIREMENTS:
100/120/130/200/230/254 VAC (±10%), 60 or 50 Hz

Processor: 350 Watts, 1200 BTU/hr.
Diskette module: 250 Watts, 850 BTU/hr.

EQUIPMENT DIMENSIONS:
Width: 19.925 in. (45.7 cm)
Height: 10.5 in. (25.4 cm)
Depth: 22.65 in. (48.3 cm)
Weight: Processor — 46 lbs. (20.4 kg)
Diskette Module — 70 lbs. (31.8 kg)

OPERATING ENVIRONMENT:
10° to 38°C (50° to 100°F)
20 to 80% Relative Humidity (Non-Condensing)

1.9 Peripherals

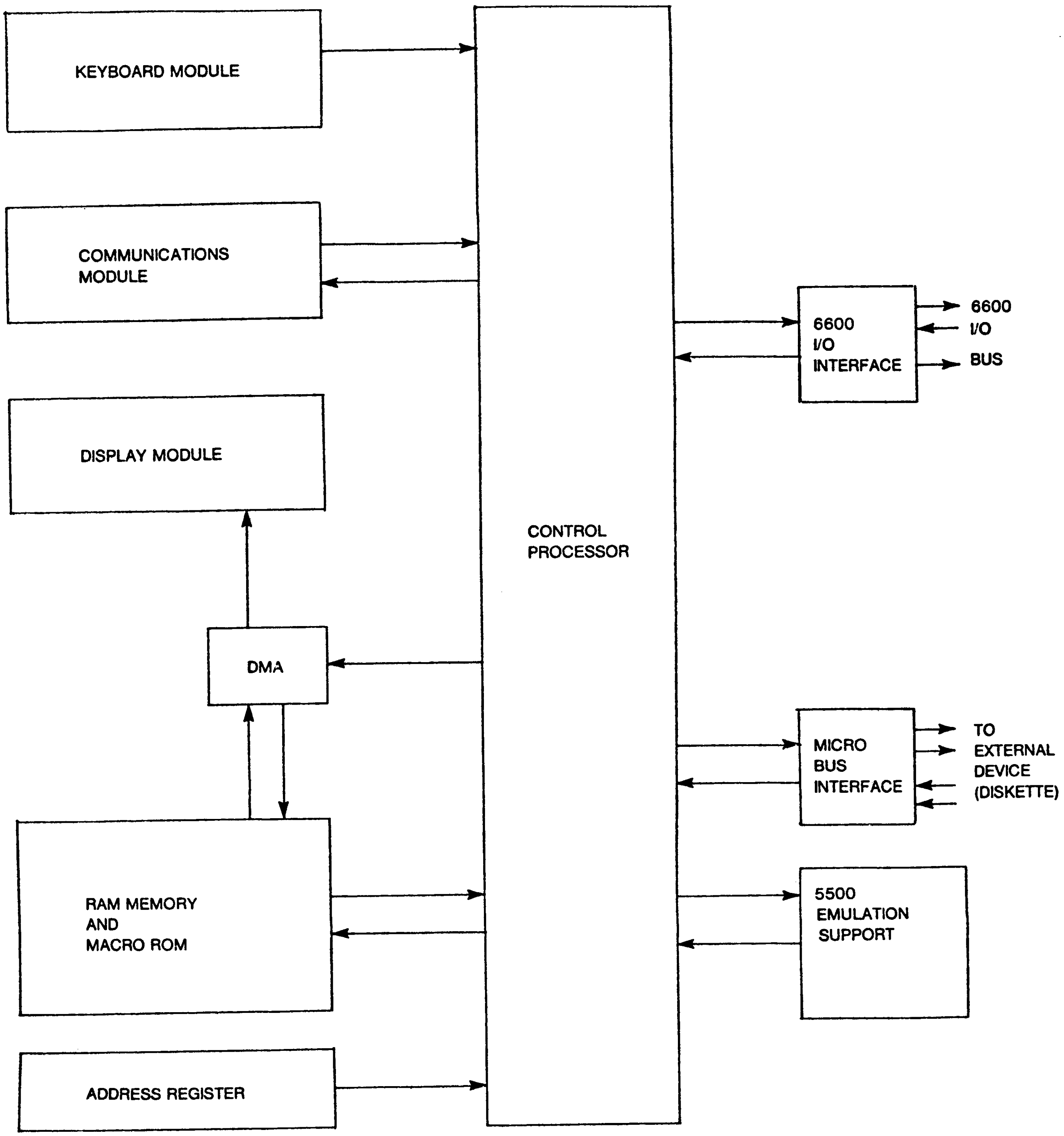
The 1800 will accommodate a wide variety of external peripherals — such as asynchronous and synchronous communications adaptors, printers, and magnetic tapes — by way of its standard Datapoint I/O bus.

Note, however, that the 1800 **cannot** accommodate peripherals that derive their power from its I/O bus. All peripherals attached to the I/O bus of the 1800 must contain their own separate power supplies; specifications for the transmitters and receivers of these devices will be found in Part 6 of this manual.

Refer to the Datapoint Equipment Catalog (Model Code 60001) for a complete description of peripherals.

1.10 Model Codes

1802 — 60K RAM Memory, internal communications adaptor, removable keyboard
1842 — 1 MB Dual-Density Diskette Drive Module



1800 BLOCK DIAGRAM
FIGURE 1-1

PART 2
KEYBOARD

2.1 General

The removable keyboard on the Datapoint 1800 performs the functions of data entry and processor control.

It provides for 76 key positions including a 10-key numeric pad, shift/shift-lock functions and the standard ASCII alphanumeric set, and features a unique multikey rollover characteristic to allow for maximum ease of typing.

The universal key coding is done in Programmable Read Only Memories (PROMS). The keyboard uses the same keycoding as the 1500 universal keyboard. It can be detached up to 1 meter from the processor. Status is provided to allow macro-programs to generate "repeat" functions if desired.

On the right side of the keyboard are 10 function keys, five of which are reserved for control over the processor as follows:

RESTART
Momentary contact switch which, when depressed, sets a status bit that may be tested at any time by the processor. When depressed together with the INTERRUPT key, RESTART causes a jump to a bootstrap routine in the ROM. To protect against accidental restarts it is necessary to depress both the RESTART and INTERRUPT keys to cause a restart. The order of depression is not significant as the restart occurs when either key is released (similar to the operation of the RESTART/RUN keys of a Datapoint 5500).

ATTENTION
Momentary contact switch which, when depressed, sets a status bit that may be tested at any time by the processor.

INTERRUPT
Momentary contact switch which, when depressed, sets a status bit that may be tested at any time by the processor. Used in conjunction with the RESTART key to effect a processor initialization.

KEYBOARD
Momentary contact switch which sets a status bit that may be tested at any time by the processor.

DISPLAY
Momentary contact switch with a function similar to that of KEYBOARD switch.

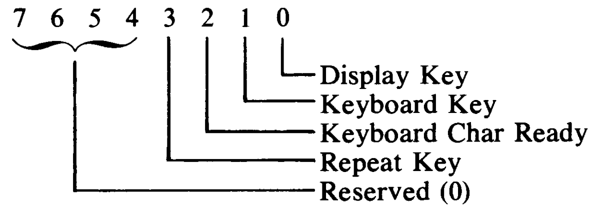
2.2 Keyboard Operation

The macro-program communicates to the keyboard module through several control/status bytes in system RAM. These control status bytes are:

LOCATION	DESCRIPTION
0167550	KBS2
0167551	KBS1
0167552	Keyboard Data
0167553	Frame Counter

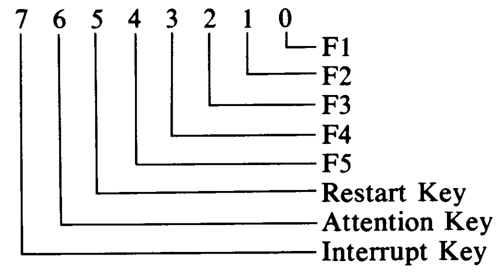
a. KBS1 — KBS1 is one of the keyboard status bytes. This byte contains status information about the function switches on the keyboard. When any of the ten function keys are depressed, the associated bit becomes a one until that key is released. When any other key is depressed, the data from the keyboard is put into the Keyboard Data byte (0167552) and the Keyboard Character Ready bit becomes a one and remains such until the software resets it. The repeat function bit becomes a one and remains a one as long as the key is depressed. The repeat bit, when active, means that the last key depressed is still pressed.

Bit Definitions:



b. KBS2 — The second keyboard status byte indicates the condition of keyboard switches F1 through F5. The corresponding status bit will be a one when the switch is being depressed and will remain a one until the switch is released — this is also true of the Display and Keyboard keys.

Bit Definitions:



c. Frame Counter — The Frame Counter is incremented once at the end of each screen frame and is used for timing the cursor blink rate.

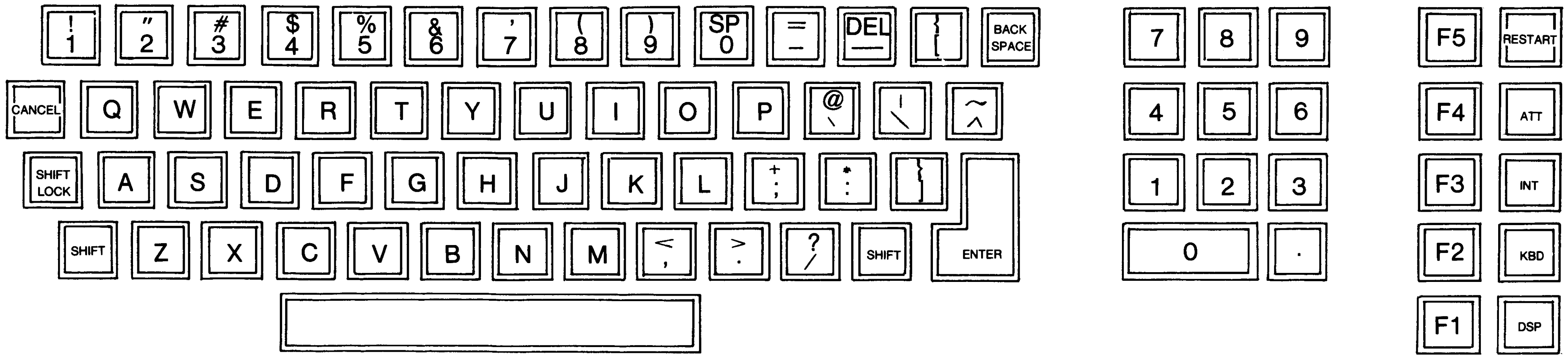
TABLE 2-1*
KEYBOARD CODING (ASCII)

A - 101	a - 141	0 - 060	: - 072
B - 102	b - 142	1 - 061	; - 073
C - 103	c - 143	2 - 062	< - 074
D - 104	d - 144	3 - 063	= - 075
E - 105	e - 145	4 - 064	> - 076
F - 106	f - 146	5 - 065	? - 077
G - 107	g - 147	6 - 066	[- 133
H - 110	h - 150	7 - 067	\ - 134
I - 111	i - 151	8 - 070	] - 135
J - 112	j - 152	9 - 071	^ - 136
K - 113	k - 153	Space Bar - 040	_ - 137
L - 114	l - 154	! - 041	@ - 100
M - 115	m - 155	" - 042	{ - 173
N - 116	n - 156	# - 043	` - 140
O - 117	o - 157	$ - 044	| - 174
P - 120	p - 160	% - 045	~ - 176
Q - 121	q - 161	& - 046	} - 175
R - 122	r - 162	' - 047	Enter - 015 (035 shifted)
S - 123	s - 163	(- 050	Bkspace - 010 (034 shifted)
T - 124	t - 164	) - 051	Del - 177
U - 125	u - 165	* - 052	Cancel - 033 (013 shifted)
V - 126	v - 166	+ - 053	Sp - 037 (shifted zero)
W - 127	w - 167	, - 054	
X - 130	x - 170	— - 055	
Y - 131	y - 171	. - 056	
Z - 132	z - 172	/ - 057	

NUMERIC PAD CODING

Symbol	Unshifted	Shifted
.	016	036
0	000	020
1	001	021
2	002	022
3	003	023
4	004	024
5	005	025
6	006	026
7	007	027
8	011	031
9	012	032

* All characters are represented in octal.



KEYBOARD LAYOUT
FIGURE 2-1

PART 3
DISPLAY

3.1 General Description

The Datapoint 1800 uses a magnetically deflected Raster Scan CRT. It provides for the display of 1920 characters organized as 24 lines of 80 characters each. It is refreshed from the processor memory through a direct memory access (DMA) channel under processor control. The display character generator is loadable, allowing any character set; it also incorporates an inverse video feature that displays characters as dark dots on a light background on a character by character basis (see Figure 3-1), under program control. Up to 128 different individual 5 x 7 dot matrix characters may be produced.

3.2 Display Operation

The macro-program communicates to the display by providing a list of pointers to the data it desires to display on the screen. The list of screen pointers is stored in memory in locations 0167554 - 0167637 and is defined as follows:

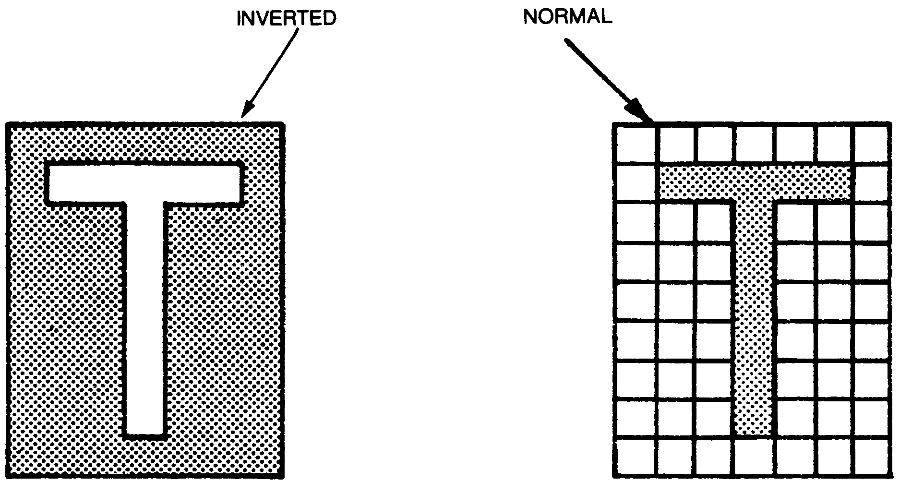
LOCATION	DEFINITION
0167554,55	Top Null Line
0167556,57	Display Line 0
0167560,61	Display Line 1
0167562-633	Display Lines 2 - 22
0167634,35	Display Line 23
0167636,37	Bottom Null Line

These locations all contain two-byte pointers to the first character in memory to be displayed in the first position of the respective line. The contents of the following 79 locations in memory will be displayed as the rest of each line. The pointers must specify memory locations that have existing physical memory (the display pointers are sectored and based) and the display pointer should not cause the display DMA to cross a sector boundary.

The character set is loaded by an instruction that is new with the 1800 series. It allows the character font for each ASCII code to be loaded from memory (see description of LODCF Instruction).

The inverse video feature is controlled by the MSB of each character to be displayed. If the MSB of the character as read out of memory by the DMA is a zero, the character is displayed normally. If the MSB is a one, the character is displayed as dark dots against a light background (see Figure 3-1).

A blinking cursor may be obtained by using one of the unused character codes (such as 0 to 0177) to form a solid cursor. Alternating the character with it at the cursor blink rate will produce a blinking cursor.



SHADED AREA REPRESENTS ILLUMINATED GREEN DOTS ON CRT

FIGURE 3-1

PART 4
COMMUNICATIONS MODULE

4.1 General Description

The Datapoint 1800 communications module is self contained within the basic 1800 unit and allows for communication under several disciplines. The module is fully buffered with a 128 character buffer on transmit as well as receive and can operate full duplex with internal or external clock up to 9600 baud. It provides automatic testing of received line signal detector and clear-to-send. It also provides Cyclical Redundancy Check error checking on both transmit and receive when needed; on transmit it automatically inserts the CRC bytes in the data block.

Finally, it provides for control of an Automatic Calling Unit for dialing.

4.2 Interface

The macro-program communicates to the communications module through status and control bytes located in system RAM, and special communication instructions.

See Appendix C for a complete description of the macro interface.

TABLE 4-1
SERIAL COMMUNICATIONS SPECIFICATIONS

Signal Levels RS232C compatible (enhanced drive capability)

Supports Synchronous and Asynchronous modems

Internal/External clock selection at modem connector

Data Rates (Baud) — 150, 300, 600, 1200, 2400, 4800, 9600 internal clock

Internal clock rate selected by jumpers

ASYNC operation provides limited baud rate selection by software

Up to 9600 baud on external clock

Firmware supports — SDLC, BISYNC, GENSYNC, and ASYNC

NRZ/NRZI — Selected by internal jumpers

ACU interface — Supports 801A, 801C or equivalent

Supervisory Channel control (ACU or Supv. not both)

See Appendix C for pin specifications.

PART 5
PROCESSOR

5.1 Macro-Instruction Set

The Datapoint 1800 macro-level processor executes Datapoint 5500 USER MODE instructions and some selected 6600 instructions plus the normal I/O instructions. Since the 1800 has no internal tape deck, none of the tape I/O commands are used. Refer to 5.7 for a description of the instruction set.

5.1.1 Macro-Level Interrupts

Certain fault conditions in the hardware and software operations will cause interrupts to occur. These interrupts cause calls to the interrupt vectors located in system RAM, which consist of jump pairs that can be overstored by the user program to change the action taken in response to the interrupt.

The interrupt vector locations are:

0167400	Memory Parity Fault
0167406	Input Parity Error
0167414	Output Parity Error
0167422	Write Violation
0167430	Access Violation
0167436	Instruction Violation
0167444	One millisecond
0167452	System Call
0167460	Break Point
0167466	Unimplemented Instruction
0167474	Sector Table Parity Error
	Special Vector Entries
0167502	Wait Called by Disk Drivers
0167505	Wait Called by Display Drivers
0167510	Beep Audio Channel Entry
0167516	Click Audio Channel Entry
0167524-36	Reserved

In addition, a fail-safe interrupt causes the equivalent of a power-on reset to the processor if macro-instruction execution is stopped for more than 30 milliseconds. The fail-safe also sets a status bit to the processor to differentiate it from a normal power-on reset, and displays on the processor screen a message: ET TIME OUT ERROR.

5.1.2 Macro-Instructions New for the 1800

In the 1800 there is a significant increase in the quantity of user information (registers, stack, etc.) stored in system RAM. To make dealing with this information easier, several instructions have been created to allow moving user data from registers to memory and back. Some of these are simply modified 5500 instructions (such as Betal), but others are new (System Move). All instructions new to the 1800 are explained in part 5.7.8 below.

5.1.3 Macro-Level Execution Rate

Execution speed on the 1800 relies on the processor time-sharing between its control operations and its instruction emulation. Moreover, DMA operations for the display reduce available memory time.

The following overheads need to be accounted for when calculating actual execution times during time-critical parts of programs. They are expressed on a microseconds per millisecond basis and represent worst cases; average overhead will be lower. Use of these overhead times will allow calculation of available execution margins in time-critical portions of programs.

5.1.4 Fixed Overhead

The fixed overhead occurs continuously and cannot be disabled. The following are peak, not average, values.

Display Control	21
Keyboard Scanning	25.3
Display DMA	121
Audio Channel	44 (even if not in use)

5.1.5 Variable Overhead

1800 variable overhead is only a factor when certain operations are being done. For example, if diskette operations are not in process, none of the diskette overhead will apply. Again, the following are peak values during one millisecond.

Diskette read or write 78.9 (single density)

Baud Rate —	9600	4800	2400	1200	600
SDLC Channel	127.45	68.20	41.35	32.40	23.45
BISYNC	53.85	28.00	21.55	19.40	17.25
ASYNC	64.75	34.15	23.50	19.95	16.40
GENSYNC	95.25	51.55	28.00	20.15	12.30

5.2 Memory

5.2.1 General

The Datapoint 1800 can address up to 64K bytes of memory, organized in 9 bit words (8 data + parity) composed of N channel, MOS random access memory and MOS ROM.

Due to the dynamic nature of the RAM used, it is necessary to periodically "refresh" the data in memory. This is performed automatically in the 1800 during the DMA cycles, eliminating any additional overhead for refresh. The display utilizes approximately 15% of the available memory cycles for the DMA operation required to refresh the screen. The parity bit structure allows for automatic detection of memory errors to prevent inadvertent operation with faulty memory. Even memory parity prevents reads from empty card positions causing a parity fault.

The 1800 may have up to 60K of RAM by using two of the 32K RAM cards — identical to those of the Datapoint 6600 except for jumper option — as well as a ROM memory card. Four K of RAM is overlapped by System ROM.

5.3 Memory Specifications

5.3.1 RAM

The RAM memory in an 1800 is provided in 2 cards of 32K bytes each, which provide a memory system with the following characteristics:

Memory Type	MOS Random Access
Memory Cycle Time	633 Nanoseconds (DMA)
Memory Cycle Time	723 to 814 Nanoseconds (Processor)

5.3.2 ROM

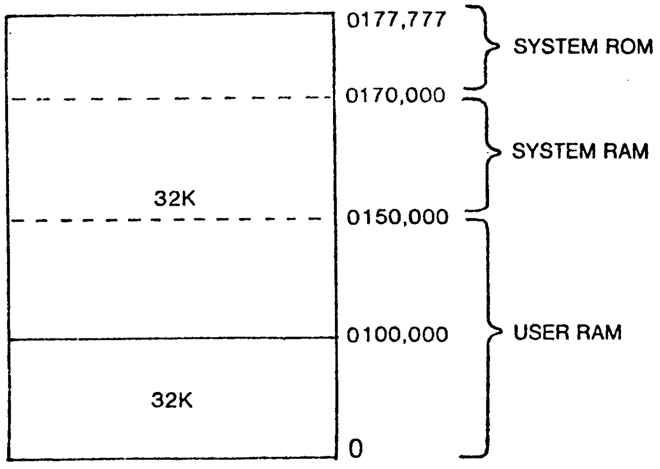
The ROM contains the following functions:

* Initialization
* Character Set Load
* Diskette Boot
* Diskette Drivers
* Display and Keyboard Drivers
* Diskette Adjustment Aids
* Limited Diskette Diagnostic Aids
* Debug
* Memory Test
* Error Message Routines

5.4 Memory Allocation

The memory in the 1800 is both ROM and RAM. The ROM memory is used for routines and system initialization, and resides from 0170000 to 0177777 octal.

The RAM memory occupies logical addresses 0 to 0167777 (see Figure 5.1). The area from logical 0150000 to 0167777 is designated for system use and contains the interrupt vectors, screen data, and macro-interface bytes for the internal modules.



LOGICAL MEMORY MAP
FIGURE 5-1 1800

5.4.1 Address Generation

User programs use what is known as a "logical" memory address. This is a 16-bit value created by the program and translated to the proper "physical" memory address by a mechanism in the processor. The translation mechanism uses a base register and a memory sector table as depicted in Figure 5.2.

If the logical memory address is between 0100000 and 0137777 its upper eight bits are added (two's complement) to the eight bit base register. Otherwise, the upper eight bits of the logical memory address are unchanged by the adder. The new 16-bit value consisting of the lower eight bits of the logical memory address and the eight bits from the adder is called the "based logical memory address." Note that the base register may be negative (two's complement) for creating based logical memory addresses lower than 0100000.

The upper four bits of the based logical memory address form an address for the 16-entry 8-bit sector table. This table divides the 64K based logical memory space into sixteen 4K byte sectors, each of which may be translated to any physical 4K memory section and may be protected from being accessed if the USER mode flag is set or from being written into regardless of the state of the USER mode flag. (Note that many people in the computer industry refer to the sector table as a page table. However, the reference has been changed here to avoid confusion with the term "page" used elsewhere to denote a 256 byte section of logical memory space starting at an address of 0 modulo 256.)

The sector table contains eight bits for each entry. Bit 1 (the next to the least significant) of a sector table entry contains a hardware-generated and checked parity bit. Any value loaded into this position is ignored since the hardware generates the proper parity bit when a sector table entry is loaded. If, during any memory access, there is not an odd number of one bits out of the eight sector table entry bit positions, a Sector Table Parity Error System Call interrupt will be generated to memory location 0167474. Bit 2 of a sector table entry is set to enable the sector to be read or written when the machine is in User Mode. Bit 3 is set to enable the sector to be written in either User or System Mode. Bits 4 through 7 of a sector table entry are used for physical memory address bits 12 through 15, and bit 0 of a sector table entry is used for physical memory address bit 16 — giving the processor a total of 17 bits of physical memory addressing capability. The 1800 offers 64K of memory.

From the address generation mechanism described above, two major benefits derive. The first is ease of reentrant coding for multiple user tasks. The programmer can load into the base register the base address (in multiples of 256 bytes) of his nonreentrant data area minus 0100000; thereafter, all references to logical memory addresses between 0100000 and 0100000 plus the length of his data area will automatically be translated into the proper based logical memory location. The second major benefit derives from the sector table. Besides providing the capability of implementing a completely protected monitor, the sector table makes it easy to run several independent partitions in memory at once.

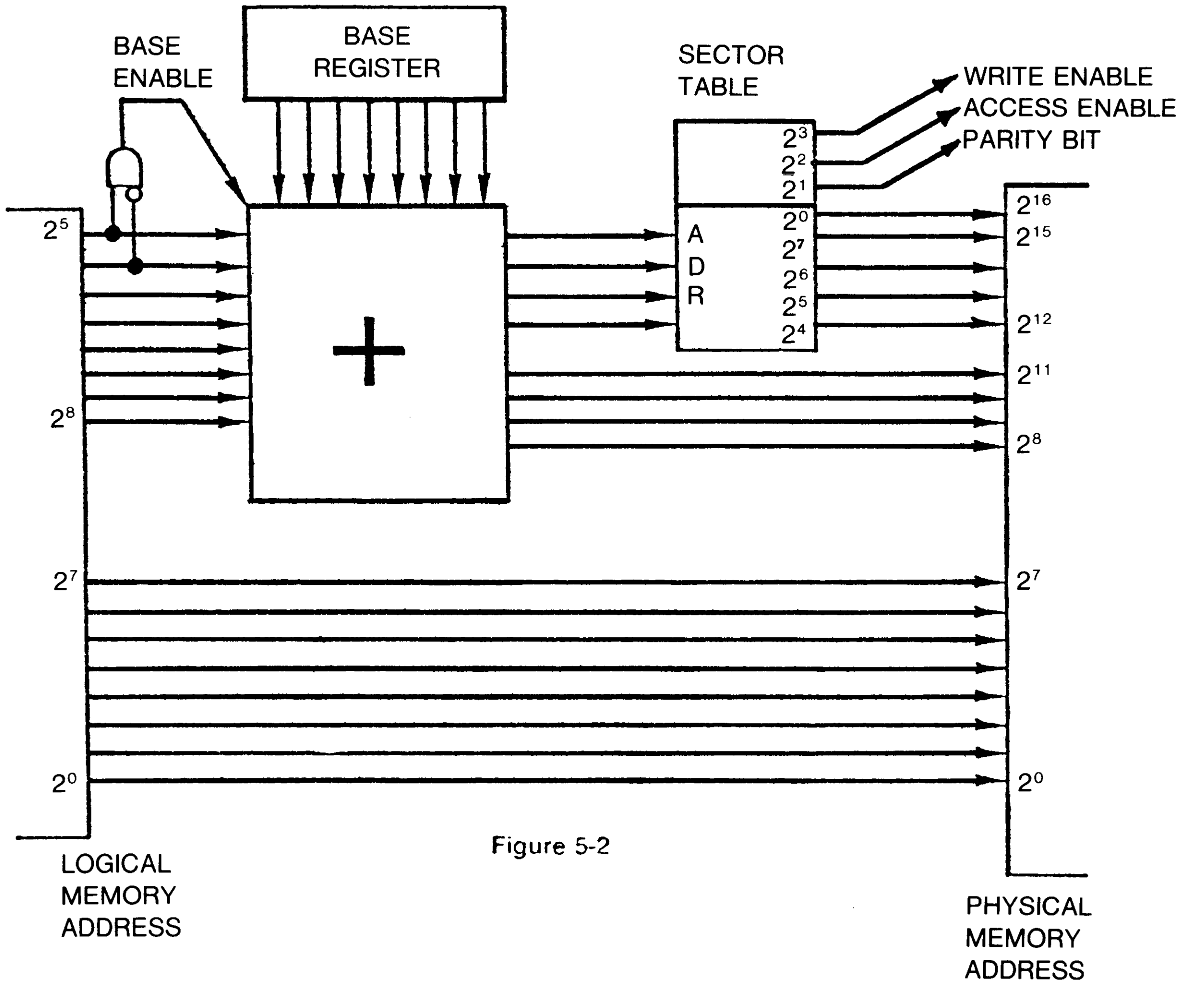


Figure 5-2

5.5 Processor Instructions

The Datapoint 1800 processor instructions have been divided into eight categories for convenience of presentation.

* Category one: All instructions contained in 1100 and 2200 system processors.
* Category two: 2200 system instructions which have been enhanced with additional register referencing capability.
* Category three: Multibyte (string) instructions.
* Category four: Instructions for saving and restoring the state of the processor.
* Category five: Address manipulation instructions.
* Category six: Operating system control instructions.
* Category seven: 6600 instruction set and instruction timing.
* Category eight: Instructions unique to the 1800.

5.5.1 Comparisons to 5500 System Instructions

The 1800, with some exceptions noted below, will execute the entire 5500 instruction set; it includes most 5500 instructions and some 6600 instructions. Most programs designed for earlier Datapoint processors will, therefore, execute normally on the 1800 without changes or with only minor changes.

However, the 1800 has no cassette decks, so operations dealing with cassettes are undefined. Moreover, the 1800 contains instructions which operate certain special peripherals, some of which have the same op-codes as 5500 cassette instructions; hence, programs that reference cassette decks **should not** be run on the 1800.

The 1800 has no HALT instruction. An attempt to execute a HALT (op-codes 000, 001, 0377) will cause an interrupt into ROM giving an error message: E8 INSTRUCTION ERROR INTERRUPT — if the system RAM vector has not been overwritten.

The 1800 CRT display and keyboard have no address which can be given with an EX ADR instruction. Instead, as explained in Part 2, the CRT display is driven directly out of fixed RAM memory locations, which are read or written to perform a desired function. Programs which directly address the CRT display or keyboard, therefore, should not be run on the 1800.

A new feature of the 1800 is its fully user-programmable audio channel. The Macro ROM uses this channel to execute EX BEEP, EX CLICK, and other, similar instructions. Users may employ this audio channel in any manner desired. A full description and operating procedures are given in Appendix D.

Finally, to communicate with certain highly specialized peripherals, such as its flexible diskette drives, the 1800 uses a special, dedicated direct processor control which is independent of its 5500/6600-compatible I/O channel. If the user must control the diskette drives directly from his program, eight basic instructions which exercise this direct processor control are available; a full discussion of them may be found in Appendix B.

5.6 Presentation Format

A description of each 1800 instruction is given below. In order to simplify the presentation, the following symbols and abbreviations are used:

Operation:	Symbolic representation of instruction description.
Op Code:	Operation Code, expressed in octal.
Timing:	Execution time in microseconds.
Length:	Number of bytes in the instruction. (Used when the length may not be especially obvious from the op code or the instruction diagram.)
Stack:	Number of stack entries.
Entry:	Conditions necessary before execution.
Exit:	Conditions existing after execution.
Algorithm:	Steps taken to perform the instruction execution.
( )	The contents of.
←	Is replaced by.
→	Is transferred to.
:	Is compared with.
V	Logical "Or" operation.
⊻	Logical "Exclusive Or" operation
Λ	Logical "AND" operation.
A	
B	
C	
D	
E	General purpose registers.
H	
L	
X	
M	Memory location designated by the contents of HL or the designated register pair.
P	Program counter.
STACK	Instruction counter pushdown queue.
(OP)	One of eight AULU operations (AD, AC, SU, SB, ND, XR, OR, CP).
(rs)	A source general register (ABCDEHL) (s=0 to 6).
(rd)	A destination general register (ABCDEHL) (d=0 to 6)
(r)	A general register (ABCDEHLX) (s or d=0 to 7).
(rp)	One of the pairs of registers (BC DE HL XA).
r	A register select op code. No byte is necessary for selection of the A register. Otherwise: B=111, C=062, D=113, E=174, H=115, L=176, X=117.
rp	A register pair select op code. No byte is necessary for the selection of HL. Otherwise: BC=062, DE=174, XA=022.
(vvv)	An 8-bit value used in an instruction.
(adr)	A 16-bit value used in an instruction with the LSP first, followed by the MSP.
(cf)	Control flags (C=0, Z=1, S=2, P=3) (often called flip-flops).
(exp)	External command, listed in Table 5-1.
data	An expression reducing to an 8-bit immediate value.
loc	An expression reducing to a 16-bit address.
(s)	Operand source; 0 through 6 refers to registers A through L, 7 refers to memory

5.7 Datapoint 1800 Instruction Set

5.7.1 Category 1 — 2200 System Instructions

LOAD IMMEDIATE **L(r)**

Op Code: 0d6 (vvv)
Timing: 4.05
Operation: (vvv) → (r)

Transfers the contents of the operand given in the instruction to the register specified by bits 3 - 5 of the instruction word.

7 6	5 4 3	2 1 0	7 0
0	d	6	OPERAND

1. d is the destination designator.
2. None of the flag flip-flops are changed.

LOAD **L(rd)M, L(rd)(rs), LM(rs)**

For L(rd) M:
Op Code: 3d7
Timing: 5.00
Operation: (M) → (rd) d≤6

For L(rd)(rs):
Op Code: 3ds
Timing: 2.65
Operation: (rs) → (rd) s≤6, d≤6

For LM(rs):
Op Code: 37s
Timing: 4.65
Operation: (rs) → (M) s≤6

Transfers the operand from the source specified by bits 0-2 of the instruction word to the destination specified by bits 3-5 of the instruction word.

7 6	5 4 3	2 1 0
3	d	s

1. The data source is unaffected.
2. s and d both = 7 results in a HALT instruction.
3. None of the flag flip-flops are changed.

ADD IMMEDIATE **AD data**

Op Code: 004 (vvv)
Timing: 5.30
Operation: (A) + (P+1) → A

Adds the contents of the operand to the contents of the A register and retains the sum in the A register.

7 6	5 4 3	2 1 0	7 0
0	0	4	OPERAND

1. Carry flip-flop set if add overflow occurs; otherwise carry is reset.
2. The Sign, Zero and Parity flip-flops indicate the status of the A register at completion.

ADD **AD(rs), ADM**

For AD(rs):
OP Code: 20s
Timing: 3.80
Operation: (A) + (rs) → A

For ADM:
Op Code: 207
Timing: 5.85
Operation: (A) + (M) → A

This instruction is identical to ADD IMMEDIATE with the exception of operand source.

7 6	5 4 3	2 1 0
2	0	s

s specifies the operand source.

ADD WITH CARRY IMMEDIATE **AC data**

Op Code: 014 (vvv)
Timing: 5.75
Operation: (A) + (P+1) + (Carry) → A

Adds the Carry bit and contents of the operand to the contents of the A register and retains the sum in the A register.

7 6	5 4 3	2 1 0	7 0
0	1	4	OPERAND

1. If add overflow occurs, the Carry flip-flop is set; otherwise Carry is reset.
2. The Sign, Zero, and Parity flip-flops indicate the status of the A register at completion.

ADD WITH CARRY **AC (rs), ACM**

For AC (rs) :
Op Code: 21s
Timing: 4.25
Operation: (A) + (Carry) + (rs) → A

For ACM:
Op Code: 217
Timing: 6.3
Operation: (A) + (Carry) + (M) → A

This instruction is identical to ADD WITH CARRY IMMEDIATE with the exception of operand source.

7 6	5 4 3	2 1 0
2	1	s

s specifies the operand source.

SUBTRACT IMMEDIATE **SU data**

Op Code: 024 (vvv)
Timing: 5.30
Operation (A) - (P+2) $\rightarrow$ A

Subtracts the contents of the operand from the contents in the A register and retains the difference in the A register.

7 6	5 4 3	2 1 0	7 0
0	2	4	OPERAND

1. The Carry flip-flop is set if underflow occurs, otherwise carry is reset.
2. The Zero, Sign and Parity flip-flops represent the status of the A register at completion.

SUBTRACT **SU(rs), SUM**

For SU(rs):
Op Code: 22s
Timing: 3.80
Operation: (A) - (rs) $\rightarrow$ A
For SUM:
Op Code: 227
Timing: 5.85
Operation: (A) - (M) $\rightarrow$ A

This instruction is identical to SUBTRACT IMMEDIATE with the exception of operand source.

7 6	5 4 3	2 1 0
2	2	s

s specifies the operand source.

SUBTRACT WITH BORROW IMMEDIATE **SB data**

Op Code: 034 (vvv)
Timing: 5.75
Operation: (A) - (P+1) - (Carry) $\rightarrow$ A

Subtracts the contents of the operand and the Carry bit from the contents of the A register, and retains the difference in the A register.

7 6	5 4 3	2 1 0	7 0
0	3	4	OPERAND

1. Sets the Carry flip-flop if underflow occurs; otherwise resets Carry.
2. The Zero, Sign, and Parity flip-flops represent the status of the A register at completion.

SUBTRACT WITH BORROW **SB(rs), SBM**

For SB(rs):
Op Code: 23s
Timing: 4.25
Operation: (A) - (rs) - (Carry) $\rightarrow$ A
For SBM:
Op Code: 237
Timing: 6.30
Operation: (A) - (M) - (Carry) $\rightarrow$ A

This instruction is identical to SUBTRACT WITH BORROW IMMEDIATE with the exception of the operand source.

7 6	5 4 3	2 1 0
2	3	s

s specifies the operand source.

AND IMMEDIATE **ND data**

Op Code: 044 (vvv)
Timing: 5.30
Operation: (A) $\wedge$ (P+1) $\rightarrow$ A

Forms the logical product of the contents of the A register with the contents of the operand and places the result in the A register.

7 6	5 4	3 2 1 0	7 0
0	4	4	OPERAND

1. Resets the Carry flip-flop upon completion.
2. The Zero, Sign and Parity flip-flops represent the status of the A register upon completion.

Sample Operation

(A Reg)	0 0 0 0 1 1 1 1
(P+1)	0 1 1 0 0 1 1 0
(A Reg)	0 0 0 0 0 1 1 0

AND **ND(rs), NDM**

For ND(rs):
Op Code: 24s
Timing: 3.80
Operation: (A) $\wedge$ (rs) $\rightarrow$ A
For NDM:
Op Code: 247
Timing: 5.85
Operation: (A) $\wedge$ (M) $\rightarrow$ A

This instruction is identical to AND IMMEDIATE with the exception of operand source.

7 6	5 4 3	2 1 0
2	4	s

s specifies the operand source.

OR IMMEDIATE **OR data**

Op Code: 064 (vvv)
Timing: 5.30
Operation: (A) V (P+1) $\rightarrow$ A

Forms the logical sum of the contents of the A register and the contents of the A register and the contents of the operand, and places the result in the A register.

7 6	5 4 3	2 1 0	7 0
0	6	4	OPERAND

1. Resets the Carry flip-flop upon completion.
2. The Zero, Sign and Parity flip-flops represent the status of the A register upon completion.

Sample operation:

(A Reg)	0 0 0 0 1 1 1 1
(P+1)	0 1 1 0 0 1 1 0
(A Reg)	0 1 1 0 1 1 1 1

OR **OR(rs), ORM**

For OR(rs):
Op Code: 26s
Timing: 3.80
Operation: (A) V (rs) $\rightarrow$ A
For ORM:
Op Code: 267
Timing: 5.85
Operation: (A) V (M) $\rightarrow$ A

This instruction is identical to OR IMMEDIATE with the exception of operand source.

7 6	5 4 3	2 1 0
2	6	s

s specifies the operand source.

EXCLUSIVE OR IMMEDIATE **XR data**

Op Code: 054 (vvv)
Timing: 5.30
Operation: (A) $\veebar$ (P+1) $\rightarrow$ A

Forms the logical difference of the contents of the A register and the operand, and places the result in the A register.

7 6	5 4 3	2 1 0	7 0
0	5	4	OPERAND

1. Resets the Carry flip-flop at completion.
2. The Zero, Sign and Parity flip-flops represent the status of the A register upon completion.

Sample Operation:

(A Reg)	0 0 1 1 0 1 0 1
(P+1)	0 1 0 1 1 1 0 0
(A Reg)	0 1 1 0 1 0 0 1

EXCLUSIVE OR **XR(rs), XRM**

For XR(rs):
Op Code: 25s
Timing: 3.80
Operation: (A) $\veebar$ (rs) $\rightarrow$ A
For XRM:
Op Code: 257
Timing: 5.85
Operation: (A) $\veebar$ (M) $\rightarrow$ A

This instruction is identical to EXCLUSIVE OR IMMEDIATE with the exception of operand source.

7 6	5 4 3	2 1 0
2	5	s

s specifies the operand source.

COMPARE IMMEDIATE **CP data**

Op Code: 074 (vvv)
Timing: 5.30
Operation: (A) : (P+1)

Compare the contents of the A register with the contents of the operand.

7 6	5 4 3	2 1 0	7 0
0	7	4	OPERAND

1. The flag flip-flops assume the same state as they would for a Subtract instruction.
2. The contents of the A register are unaffected.

COMPARE **CP(rs), CPM**

For CP(rs):
Op Code: 27s
Timing: 3.80
Operation: (A):(rs)
For CPM:
Op Code: 277
Timing: 5.85
Operation: (A):(M)

This instruction is identical to COMPARE IMMEDIATE with the exception of operand source.

7 6	5 4 3	2 1 0
2	7	s

s specifies the operand sources.

UNCONDITIONAL JUMP **JMP**

Op Code: 104 (adr)
Timing: 5.60
Operation: (adr) $\rightarrow$ P

An unconditional transfer of control. The second byte of the instruction represents the least significant portion of the jump address, while the third byte of the instruction represents the most significant portion.

			P+1	P+2
7 6	5 4 3	2 1 0	7 0	7 0
1	0	4	LSP	MSP
	Op Code		Address	

JUMP IF CONDITION TRUE **JT(cf) loc**

Op Code: 1 (c+4) 0 (adr)
Timing: 6.05
(2.90) if jump not taken)
Operation: If condition True, (adr) → P

Examines the designated flip-flop. If set, transfers control to (adr). If reset, executes the next sequentially available instruction.

			P+1	P+2
7 6	5 4 3	2 1 0	7 0	7 0
1	c+4	0	LSP	MSP
Op Code			Address	

1. c designates which flip-flop (condition) is to be tested.
2. The condition of the selected flip-flop is unchanged by this instruction.

JUMP IF CONDITION FALSE **JF(cf)**

Op Code: 1c0 (adr)
Timing: 6.05
(2.90 if jump not taken)
Operation: if condition False, (adr) → P

Examines the designated flip-flop. If reset, transfers control to (adr). If set, executes the next sequentially available instruction.

			P+1	P+2
7 6	5 4 3	2 1 0	7 0	7 0
1	c	0	LSP	MSP
Op Code			Address	

1. c designates which flip-flop (condition) is to be tested.
2. The condition of the selected flip-flop is unchanged by this instruction.

SUBROUTINE CALL **CALL**

Op Code: 106 (adr)
Timing: 10.70
Operation: P+3 → STACK, (adr) → P

Transfers the address of the next sequentially available instruction to the Pushdown Stack, and transfers control to address specified by the contents of the two memory locations immediately following the Op Code.

			P+1	P+2
7 6	5 4 3	2 1 0	7 0	7 0
1	0	6	LSP	MSP
Op Code			Address	

SUBROUTINE CALL IF CONDITION TRUE **CT(cf) loc**

Op Code: 1 (c+4) 2 (adr)
Timing: 11.15
(3.10 if condition not met)
Operation: If condition True, P+3 → STACK,
(adr) → P

Examines the designated flip-flop. If set, transfers the address of the next sequentially available instruction to the pushdown stack, and transfers control to (adr). If reset, executes the next sequentially available instruction.

1. c designates which flip-flop (condition) is to be tested.
2. The condition of the selected flip-flop is unchanged by this instruction.
3. The stack is open-ended in operation. If it is overfilled, the deepest address will be lost.

			P+1	P+2
7 6	5 4 3	2 1 0	7 0	7 0
1	c+4	2	LSP	MSP
Op Code			Address	

SUBROUTINE CALL IF CONDITION FALSE **CF (cf) loc**

Op Code: 1c2 (adr)
Timing: 11.15
(3.10 if condition not met)
Operation: If condition False, P+3 → STACK,
(adr) → P

Examines the designated flip-flop. If reset, transfers the address of the next sequentially available instruction to the pushdown stack, and transfers control to (adr). If set, executes the next sequentially available instruction.

			P+1	P+2
7 6	5 4 3	2 1 0	7 0	7 0
1	c	2	LSP	MSP
Op Code			Address	

1. c designates which flip-flop (condition) is to be tested.
2. The condition of the selected flip-flop is unchanged by this instruction.
3. The stack is open-ended in operation. If it is overfilled, the deepest address will be lost.

SUBROUTINE RETURN **RET**

Op Code: 007
Timing: 7.10
Operation: (STACK) → P

Transfers control to the address specified by the most recent entry in the pushdown Stack. Deletes the most recent entry from the stack.

7 6	5 4 3	2 1 0
0	0	7

The effect of attempting more Return instructions than the stack is capable of handling is undefined.

SUBROUTINE RETURN IF CONDITION TRUE RT(cf)
Op Code: 0 (c+4) 3
Timing: 7.45
(2.60 if condition not met)
Operation: If condition True, (STACK) → P

Examines the designated flip-flop. If set, transfers control to the address specified by the most recent entry in the pushdown stack and deletes the most recent entry in the stack. If reset, executes the next sequentially available instruction.

7 6	5 4 3	2 1 0
0	c+4	3

1. c designates which flip-flop (condition) is to be tested.
2. The condition of the selected flip-flop is unchanged by this instruction.
3. The effect of attempting more Return instructions than the Stack is capable of handling is undefined.

SUBROUTINE RETURN IF CONDITION FALSE RF (cf)
Op Code: 0c3
Timing: 7.45
(2.60 if condition not met)
Operation: If condition False, (STACK) → P

Examines the designated flip-flop. If reset, transfers control to the address specified by the most recent entry in the pushdown Stack and deletes the most recent entry in the Stack. If set, executes the next sequentially available instruction.

7 6	5 4 3	2 1 0
0	c	3

1. c designates which flip-flop (condition) is to be tested.
2. The condition of the selected flip-flop is unchanged by this instruction.
3. The effect of attempting more Return instructions than the Stack is capable of handling is undefined.

SHIFT RIGHT CIRCULAR **SRC**
Op Code: 012
Timing: 3.30
Operation: An → A (n-1), A0 → A7, A0 → Carry

Shifts the contents of the A register right in a circular fashion. Shifts the least significant bit into the most significant bit position. Upon completion of the operation, the Carry flip-flop is equal to the most significant bit.

7 6	5 4 3	2 1 0
0	1	2

The Zero, Parity and Sign flip-flops are not affected by this instruction.

SHIFT LEFT CIRCULAR **SLC**
Op Code: 002
Timing: 3.10
Operation: A(n-1) → A(n), A7 → A0, A7 → Carry

Shifts the contents of the A register left in a circular fashion. Shifts the most significant bit into the least significant bit position. Upon completion of the operation, the Carry flip-flop is equal to the least significant bit.

7 6	5 4 3	2 1 0
0	0	2

The Zero, Parity and Sign flip-flops are not affected by this instruction.

NO OPERATION **NOP**
Op Code: 300
Timing: 2.15
Operation: P+1 → P

No operation is performed

7 6	5 4 3	2 1 0
3	0	0

The Zero, Parity and Sign flip-flops are not affected by this instruction.

POP **POP**
Op Code: 060
Timing: 6.75
Operation: (STACK) → H, L

Transfers the most recent stack into the H & L register. H=MSP, L=LSP.

7 6	5 4 3	2 1 0
0	6	0

PUSH **PUSH**
Op Code: 070
Timing: 6.80
Operation: H, L → Stack

Transfers the contents of the H & L registers onto the pushdown stack. H=MSP, L=LSP.

7 6	5 4 3	2 1 0
0	7	0

ENABLE INTERRUPTS **EI**
Op Code: 050
Timing: 2.20

Following the next instruction, will allow the interrupts to occur until a DISABLE INTERRUPT instruction is executed.

Priv. Note: If USER mode is set, this instruction will cause a privileged instruction interrupt to occur.

7 6	5 4 3	2 1 0
0	5	0

DISABLE INTERRUPTS **DI**
Op Code: 040
Timing: 3.20

Prevents interrupts from occurring until an ENABLE INTERRUPT instruction is executed.

Priv. Note: If USER mode is set, this instruction will cause a privileged instruction interrupt to occur.

7 6	5 4 3	2 1 0
0	4	0

SELECT ALPHA MODE **ALPHA**
Op Code: 030
Timing: 34.05 (if in ALPHA, 3.50)

Selects the ALPHA MODE registers and control flip-flops.

Priv. Note: If USER mode is set, this instruction will cause a privileged instruction interrupt to occur.

7 6	5 4 3	2 1 0
0	3	0

SELECT BETA MODE **BETA**
Op Code: 020
Timing: 34.15 (if in Beta, 3.50)

Selects the BETA MODE registers and control flip-flops.

Priv. Note: If USER mode is set, this instruction will cause a privileged instruction interrupt to occur.

7 6	5 4 3	2 1 0
0	4	0

INPUT **INPUT**
Op Code: 101
Timing: 6.125
Operation: (I/O Bus) $\rightarrow$ A

Transfers the contents of the I/O Bus to the A register.

Priv. Note: If USER mode is set, this instruction will cause a privileged instruction interrupt to occur.

7 6	5 4 3	2 1 0
1	0	1

EXTERNAL COMMAND **EX(exp)**
Op Code: 121 to 153
Timing: 11.70
Operation: Performs I/O control according to (exp)

These instructions perform the functions necessary for control of the I/O System and external devices. Note that for code 0121 (EX ADR) the value in A is also written to memory in location 0167652.

Priv. Note: If USER mode is set, this instruction will cause a privileged instruction interrupt to occur.

7 6	5 4 3	2 1 0
0 1	X X X	X X 1

Table 5-1 is a list of the External Commands. For a detailed discussion of their use, reference should be made to Part 6 (Input/Output Operations) and to descriptions of the separate external devices. External Commands 155 - 177 are not listed, as they apply to systems with integral cassette units.

TABLE 5-1
EXTERNAL COMMANDS

EX(exp)

(exp)	OCTAL CODE	COMMAND	DESCRIPTION
ADR	121	Address	Selects device specified by A register
STATUS	123	Sense Status	Connects selected device status to input lines
DATA	125	Sense Data	Connects selected device data to input lines
WRITE	127	Write Strobe	Signals selected device that output data word is on output lines
COM1	131	Command 1	Outputs a control function to selected device
COM2	133	Command 2	Outputs a control function to selected device
COM3	135	Command 3	Outputs a control function to selected device
COM4	137	Command 4	Outputs a control function to selected device
BEEP	151	Beep	Activates tone producing mechanism
CLICK	153	Click	Activates audible click producing mechanism

NOTE: All instructions in this table are privileged instructions.

5.7.2 Category 2 — Augmented Category 1 Instructions

LOAD REGISTER FROM MEMORY USING BC, DE, OR XA FOR THE ADDRESS **L(rd)M (rp)**

Op Code: I rp I 3d7
Timing: 6.95
Operation: (M) → (rp), d < 7
Length: 2 bytes
Example: LEM BC

Identical to the L(rd) M instruction except that the specified register pair, instead of HL, is used for the memory address.

LOAD MEMORY FROM REGISTER USING BC, DE, OR XA FOR THE ADDRESS **LM(rs) (rp)**

Op Code: I rp I 37s
Timing: 6.60
Operation: (rs) M, s < 6
Length: 2 bytes
Example: LMB DE

Identical to the LM(rs) instruction except that the specified register pair, instead of HL, is used for the memory address.

ARITHMETIC AND LOGICAL OPERATIONS TO OTHER THAN THE A REGISTER

Mnemonics:	Examples:
(op)(rs) (r)	ADAB adds A to B
(op)M (r)	ADMC adds (HL) to C
(op)(r) (vvv)	SUC 20 subtracts 20 from C
SRC (r)	SRCB shifts B right
SLC (r)	SLCD shifts D left

Op Codes:	I r I 2ps,	I r I 2p7,	I 0p4,	I r I 012,	I r I 002
Timing:	5.90	7.95	7.40	5.20	5.40

Length: Add 1 byte to the equivalent category 1 instruction.

Identical to the equivalent category 1 arithmetic operations except that the specified register, instead of the A register, is used.

SHIFT RIGHT EXTENDED **SRE, SRE(r)**

For SRE:
- Op Code: 032
- Timing: 3.65
- Operation: An $\rightarrow$ A(n-l), Carry $\rightarrow$ A7, A0 $\rightarrow$ Carry
- Length: 1 byte

For SRE(r):
- Op Code: | r | 032
- Timing: 5.75
- Operation: (r)n $\rightarrow$ (r)(n-1) Carry $\rightarrow$ (r)7, (r)0 $\rightarrow$ Carry
- Length: 2 bytes

The register is shifted right one place with the left hand bit being replaced by the Carry and the Carry being replaced by the right-hand bit.

I/O USING OTHER THAN THE A REGISTER **IN(r), EX(rs) (exp)**

For IN(r):
- Op Code: | r | 101
- Timing: 8.225
- Operation: (I/O Bus) $\rightarrow$ (r)
- Length: 2 bytes

For EX(rs) (exp):
- Op Code: | r | 121, | r | 123, etc.
- Timing: 13.80
- Operation: Performs I/O control with specified register according to (exp)
- Length: 2 bytes

Identical to the 2200 I/O operations except that the specified register, instead of the A register, is used.

Priv. Note: If USER mode is set, these instructions will cause a privileged instruction interrupt to occur.

PARITY CHECKING INPUT **PIN, PIN(r)**

For PIN:
- Op Code: 103
- Timing: 6.575
- Length: 1 byte

For PIN(r):
- Op Code: | r | 103
- Timing: 8.675
- Length: 2 bytes

Identical to the INPUT instruction except that if the nine bits of the I/O Bus contain an even number of ones, an interrupt will occur.

Priv. Note: If USER mode is set, these instructions will cause a privileged instruction interrupt to occur.

PUSH USING BC, DE, OR XA **PUSH(rp)**

- Op Code: | r | 070
- Timing: 7.45
- Operation: (rp) $\rightarrow$ STACK
- Length: 2 bytes

Pushes the specified register pair onto the Stack.

PUSH IMMEDIATE **PUSH loc**

- Op Code: 051 (adr)
- Timing: 10.70
- Operation: (adr) $\rightarrow$ STACK
- Length: 3 bytes

Pushes the contents of the operand onto the Stack.

POP USING BC, DE, OR XA **POP(rp)**

- Op Code: | r | 060
- Timing: 9.20
- Operation: (STACK) $\rightarrow$ (rp)
- Length: 2 bytes

Pops the stack into the specified register pair.

5.7.3 Category 3 — Multibyte (string) Operations

BLOCK TRANSFER OR BLOCK TRANSFER REVERSE **BT, BTR**

For BT:
- Op Code: 021
- Timing: If a match — 1.85 + (N•5.55)
  If no match — 2.35 + (N•5.55)
  (where N = number of bytes transferred)
- Length: 1 byte

For BTR:
- Op Code: 111 021
- Timing: If a match — 3.95 + (N•5.75)
  If no match — 4.45 + (N • 5.75)
  (where N = number of bytes transferred)
- Length: 2 bytes

The Block Transfer instructions move the number of bytes specified in the C register from the field pointed to by HL to the field pointed to by DE while adding the contents of the A register to each byte transferred. BT causes the pointers to be incremented after each transfer while BTR causes the pointers to be decremented after each transfer. If the B register is not zero, the transfer will stop after a character which is equal to the 2's complement of the B register is stored in the destination field (stops after the matching character is moved).

Entry:	HL = location of first byte.
	DE = location of first destination byte.
	C = number of bytes to move (C = 1 to 255; 0 for 256).
	B = 2's complement of terminating character if not 0.
	A = 8-bit value added to each byte as it is moved (for de-zoning and zoning decimal numbers).
Exit:	HL = location past last source byte read.
	DE = location past destination byte written.
	A = entry value.
	B = entry value.
	C = zero or count before terminator character found.
	Condition flags are altered.

BLOCK CONVERT **BCV**

Op Code: 062 021
Timing: If a match — 3.95 + (N•7.45)
If no match — 4.45 + (N•7.45)
(where N = number of bytes converted)
Length: 2 bytes

Block Convert is a variation of Block Transfer where the field pointed to by the DE registers is translated byte-by-byte using the translate table pointed to by the HL registers.

Entry:
HL = location of the translate table.
DE = location of the first byte to be translated.
C = number of bytes to move.
B = 2's compliment of terminating character if not 0.
A = no entry value used.

Exit:
HL = undefined.
DE = location past last destination byte.
A = LSB of last table position used for translation.
B = entry value.
C = zero or count before termination character found.

Algorithm:
1. Get the byte pointed to by DE.
2. Get A to the result of the byte added to L.
3. Get the byte pointed to by HA. This is the table's translated byte.
4. Store the translated byte where DE points.
5. Increment DE.
6. B is added to the translated byte.
7. Stop if the Carry and Zero conditions are True — a match is found.
8. Decrement the C register.
9. Go to Step 1 if result is non-zero.

BINARY FIELD ADD WITH CARRY OR SUBTRACT WITH BORROW **BFAC, BFSB**

For BFAC:
Op Code: 011
Timing: 2.35 + (C•6.70)
Length: 1 byte

For BFSB:
Op Code: 031
Timing: 2.35 + (C•6.90)
Length: 1 byte

These instructions take the field pointed to by HL and either add it to or subtract it from the field pointed to by DE, leaving the result in the field pointed by DE. The fields may be 1 through 16 bytes in length.

Entry:
HL = location of right hand byte of the operand field.
DE = location of right hand byte of the accumulator field.
C = the field width (1 through 16; 0 or 16 implies 16).
Carry = carry or borrow into the operation.

Exit:
HL = location to left of the left hand byte of the operand field.
DE = location to left of the left hand byte of the Accumulator field.
C = indeterminate.
Carry = carry or borrow out of the operation (all the condition flags are altered).

Algorithm:
1. Get the byte pointed to by HL.
2. Add it with carry or subtract it with borrow from the byte pointed to by DE and store the result where DE points.
3. Decrement HL and DE by one.
4. Decrement the lower 4 bits of C register by one.
5. Go to step 1 if the lower 4 bits of the C register is not now zero.

Stack: 2 entries used.

BLOCK COMPARE **BCP**

Op Code: 041
Timing: If a match — 2.35 + (N•5.45)
If no match — 1.85 + (N•5.45)
(where N = number of bytes compared)
Length: 1 byte

This instruction matches two strings of bytes from right to left until either a mismatch is found or the specified maximum number of bytes has been scanned.

Entry:
HL = location of left hand byte of the subtracting field.
DE = location of left hand byte of the field subtracted from.
C = the maximum number of bytes to scan (1 thru 255;0 implies 256).

Exit:
IF A MISMATCH WAS FOUND:
HL = location after the mismatch in the subtracting field.
DE = location after the mismatch in the field subtracted from.
C = entry value minus number of bytes that matched.
Condition flags all reflect the result of the subtract instruction that found the two bytes differing.
IF ALL BYTES MATCHED
HL = location after the last byte in the subtracting field.
DE = location after the last byte in the field subtracted from.
C = zero.
Condition flags are altered.

Algorithm:
1. Get the byte pointed to by HL.
2. Subtract the byte pointed to by DE from it.
3. Increment DE and HL.
4. Exit if the Zero condition is False.
5. Decrement C.
6. Go to Step 1 if the lower 4 bits of the C register are not equal to zero.
7. Exit with the Zero condition True.

DECIMAL FIELD ADD WITH CARRY **DFAC**

Op Code: 111 041
Timing: 4.45 + (C•7.45)
Length: 2 bytes

This instruction takes the field of zoned BCD digits pointed to by HL and adds it to the field of zoned decimal digits pointed to by DE, leaving the result in the field pointed to by DE. The zone bits of the result field are set to the zone bits in the B register. The fields may be 1 through 16 bytes in length.

Entry: Same as for the BFAC instruction except
B = output zoning (right 4 bits must be 0; left 4 bits must be other than 0000).

Exit: Same as for the BFAC instruction except
A register is destroyed.
B = entry value.

Algorithm:
1. Get the byte pointed to by HL.
2. Add it with carry to the byte pointed to by DE.
3. Strip away the zone bits.
4. Clear the Carry and go to step 7 if the result is less than 10.
5. Subtract 10 from the result and set the Carry.
6. Set the zoning bits.
7. Store the result where DE points.
8. Decrement the H & L registers by one.
9. Decrement the C register by one.
10. Go to step 1 if the lower 4 bits of the C register are now now zero.

NOTE: The binary values for the zoned BCD digits with xxxx not equal to 0000 are as follows (the digits are not packed, i.e., only one digit per byte):

0:xxxx0000	5:xxxx0101
1:xxxx0001	6:xxxx0110
2:xxxx0010	7:xxxx0111
3:xxxx0011	8:xxxx1000
4:xxxx0100	9:xxxx1001

DECIMAL FIELD SUBTRACT WITH BORROW **DFSB**

Op Code: 062 041
Timing: 4.45 + (C•7.65)
Length: 1 byte

This instruction takes the field of zoned BCD digits pointed to by HL and subtracts it from the field of zoned BCD digits pointed to by DE, leaving the result in the field pointed to by DE. The zone bits of the two fields must be identical. The zone bits of the result field are set to the zone bits in the B register. The fields may be 1 through 16 bytes in length.

Entry: Same as for the DFAC instruction.
Exit: Same as for the DFAC instruction.
Algorithm:
1. Get the byte pointed to by HL.
2. Subtract it, with borrow, from the byte pointed to by DE.
3. Go to Step 5 and clear the Carry if the byte result is not negative.
4. Add 10 to the result and set the Carry.
5. Set the zone bits to those in the B register.
6. Store the result where DE points.
7. Decrement HL and DE by one.
8. Decrement the C register by one.
9. Go to Step 1 if the lower 4 bits of the C register is not now zero.

BINARY FIELD SHIFT LEFT **BFSL**

Op Code: 075
Timing: 2.35 + (C•4.45)
Length: 1 byte

This instruction shifts a field of bytes in memory left one bit position as if all of the bytes made up one continuous word.

Entry: HL = location of right-hand byte of the field.
C = the field width (1 through 16; 0 or 16 implies 16).
Carry = bit shifted out on the left.

Exit: HL = location left of the left-hand byte of the field.
C = indeterminate.
A = indeterminate.
Carry = bit shifted out on the left.
All other flags are indeterminate.

BINARY FIELD SHIFT RIGHT **BFSR**
Op Code: 111 075
Timing: 4.45 + (C•4.55)
Length: 2 bytes

This instruction is similar to BFSL except the shift is in the opposite direction.

Entry:	HL = location of left-hand byte of the field.
	C = the field width (1 through 16; 0 or 16 implies 16).
	Carry = bit shifted in on left.
Exit:	HL = location right of the right-hand byte of the field.
	C = indeterminate.
	A = indeterminate.
	Carry = bit shifted out on the right.
	All other flags are indeterminate.

MULTIPLE INPUT **MIN**
Op Code: 111 061
Timing: 10.125 + (C•8.30)
Length: 2 bytes

This instruction moves the number of bytes specified in the C register from a buffered input device to the field pointed to by L. The number of bytes moved is the number in the C register modulo 16. To make transferring up to 256 bytes easy yet interruptible, the full eight-bit value of the C register is retained during loop counting and exit is made with the C register containing its entry value minus the number of bytes transferred, HL containing its entry value plus the number of bytes transferred, and the Zero condition code reflecting the eight-bit result of the last decrementation of the C register. Thus the interruptible loop for transferring the number of bytes indicated by the eight bit value in the C register yet not inhibiting interrupts would appear as follows:

```
LOOP    LA      DEVADR
        DI
        EX      ADR
        EX      DATA
        MIN
        EI
        JFZ     LOOP
```

Note that the device must be readdressed for each execution of the MIN instruction since an interrupt could cause some other device to be addressed. The MIN instruction causes a parity checking input strobe to be executed every 8.30 microseconds. This execution operates without regard to any status bits of any kind. It is included for use with 5500 system I/O devices with parity generation and faster buffers allowing them to be used at data rates equivalent to DMA channels. The MIN instruction has all of the advantages of a non-I/O device interrupting system (lower software overhead in high throughput situations, superior control over the occurrence of events allowing probability of correctness in the program logic and repeatability of even occurrence, and simpler hardware using lower speeds and noise filtered buses) and yet achieves DMA throughout rates.

Priv. Note:	If USER mode is set, this instruction will cause a privileged instruction interrupt to occur.
Entry:	HL = location of first destination byte.
	C = number of bytes to move (this number is taken modulo 16 and if it is 0 modulo 16 then 16 bytes will be moved).
Exit:	HL = location of entry value plus number of bytes moved.
	C = entry valve minus number of bytes
Algorithm:	1. Execute a parity checking INPUT.
	2. Store the byte where HL points.
	3. Increment HL.
	4. Decrement C using the ALU.
	5. Exit if the lower 4 bits of the C register is zero.
	6. Go to step 1.

MULTIPLE OUTPUT **MOUT**
Op Code: 111 071
Timing: 10.125 + (C•9.05)
Length: 2 bytes

This instruction is similar to the MIN instruction except for timing and the direction of information flow. MOUT moves the number of bytes specified in the C register from the field pointed to by HL to a buffered output device. A byte is written using the EX WRITE strobe every 9.05 microseconds.

5.7.4 Category 4 — Processor State Save and Restore Instructions

STACK STORE **STKS**
Op Code: 065
Timing: 2.35 + (N•9.50)
Length: 1 byte

The Stack Store instruction POPs a specified number of stack entries and stores them (LSB followed by MSB) in the field pointed to by HL.

Entry:	HL = first location in the storage area.
	C = the number of entries to be stored (1 through 16; 0 or 16 implies 16).
Exit:	HL and C indeterminate.
	Condition flags unchanged.

STACK LOAD **STKL**

Op Code: 111 065
Timing: 4.45 + (N●10.60)
Length: 2 bytes

The Stack Load instruction pushes onto the stack the specified number of entries from the field pointed to by HL. Upon entry HL points to the right hand byte and entries are loaded in reverse order to allow restoring the stack from locations stored using the STKS instruction.

Entry:	HL = last location in the storage area.
	C = the number of entries to be PUSHED (1 through 16; 0 or 16 implies 16).
Exit:	HL = indeterminate.
	C = indeterminate.
	Condition flags unchanged.

REGISTER STORE **REGS**

Op Code: 055
Timing: 21.60
Length: 1 byte

The Register Store instruction stores all of the registers for the currently selected mode (Alpha or Beta) in the field pointed to by the top entry of the stack. This entry points to the right hand byte of the field and the registers are stored in reverse order moving to the left (XLHEBCBA from right to left). When the instruction terminates, the top entry of the stack points to the left of the left hand byte in the field. For example, if entry is made with the top entry of the stack pointing to location 02007 (octal), the registers are stored as follows:

02000:A
02001:B
02002:C
02003:D
02004:E
02005:H
02006:L
02007:X

In the above example, the top entry of the stack will be 01777 when the instruction terminates. Neither the contents of the registers nor the condition flags for the given mode are altered by this instruction.

REGISTER LOAD **REGL**

Op Code: 111 065
Timing: 19.15
Length: 2 bytes

The Register Load instruction loads all the registers for a given mode (Alpha or Beta) from the field pointed to by HL. Upon entry, HL points to the right hand byte of the field. The registers are loaded in reverse order moving to the left in the field. In this manner, the registers can be reloaded from values stored by the REGS instruction. In the example given for the REGS instruction were entered with HL = 02007, the registers shown would be loaded from the locations shown. The condition flags are not altered by this instruction.

CONDITION CODE SAVE **CCS, CCS(r)**

Op Code: 042, I r I 042
Timing: 2.85, 4.95
Length: 1 byte or 2 bytes if I r I specified.

This instruction loads the register (r) with a value such that if the value is added to itself using the AD operation, the condition flags will all be restored to their state before the CCS instruction was executed. The logic equations for the value loaded into (r) are:

A7 = Carry
A6 = Sign
A5 = A4 = A3 = A2 = O
A1 = Not Zero and Not Sign
A0 = Not Zero and Not Parity

This instruction does not alter the state of any of the condition flags. If (r) is not specified, the A register is used.

5.7.5 Category 5 — Address Manipulation Instructions

INCREMENT REGISTER PAIR **INCP**

Mnemonics	Op Codes	Timing
INCP HL	015	3.60
INCP HL,2	117 015	5.70
INCP HL,A	017	3.65
INCP BC	062 015	5.55
INCP BC,2	113 015	5.70
INCP BC,A	062 017	5.60
INCP DE	174 015	5.55
INCP DE,2	115 015	5.70
INCP DE,A	174 017	5.60
INCP XA	022 015	5.55
INCP XA,2	111 015	5.70
INCP XA,A	022 017	5.60

These instructions increment the indicated register pair by either one, two or the contents of the A register. The increment value is added to the LSP register and then the carry is added to the MSP register. All conditions are indeterminate. The A register is not changed, except in the XA case.

DECREMENT REGISTER PAIR — DECP

Mnemonics	Op Codes	Timing
DECP HL	035	3.60
DECP HL,2	117 035	5.55
DECP HL,A	037	3.65
DECP BC	062 035	5.55
DECP BC,2	113 035	5.70
DECP BC,A	062 037	5.60
DECP DE	174 035	5.55
DECP DE,2	115 035	5.70
DECP DE,A	174 037	5.60
DECP XA	022 035	5.55
DECP XA,2	111 035	5.70
DECP XA,A	022 037	5.60

These instructions decrement the indicated register pair by either one, two, or the contents of the A register. The decrement value is subtracted from the LSP register and then the borrow is subtracted from the MSP register. All condition flags are determinate. The A register is not changed, except in the XA case.

DOUBLE LOAD — DL

Mnemonics	Op Codes	Timing
DL DE,HL	047	7.00
DL BC,HL	111 047	9.80
DL BC,BC	062 047	8.95
DL BC,DE	113 047	9.10
DL DE,BC	174 047	8.95
DL DE,DE	115 047	9.10
DL HL,BC	176 047	8.95
DL HL,DE	117 047	9.10
DL HL,HL	057	7.00

These instructions load the register pair specified by the first operand from the memory location pointed to by the register pair specified by the second operand. The LSP register (C,E, or L) is loaded from the specified memory location and the MSP register (B,D, or H) is loaded from the next higher memory location. Note that indirect addressing can be accomplished by loading a register pair from the locations that the pair specify (DL HL, HL for example).

DOUBLE STORE — DS

Mnemonics	Op Codes	Timing
DS DE,HL	027	6.60
DS BC,HL	111 027	9.40
DS BC,DE	113 027	8.70
DS DE,BC	174 027	8.55
DS HL,BC	176 027	8.55
DS HL,DE	117 027	8.70

These instructions store the register pair specified by the first operand into the memory locations pointed to by the register pair specified by the second operand. The LSP register (C,E, or L) is stored in the specified memory location and the MSP register (B, D, or H) is stored in the next higher location.

PAGED LOAD — PL

Mnemonics	Op Codes	Timing
PL A,(loc)	105 LSP	6.35
PL B,(loc)	114 LSP	6.35
PL C,(loc)	124 LSP	6.35
PL D,(loc)	134 LSP	6.35
PL E,(loc)	144 LSP	6.35
PL H,(loc)	154 LSP	6.35
PL L,(loc)	164 LSP	6.35

These instructions load the specified register from the memory location specified by the LSP given in the instruction and the X register.

PAGED STORE — PS

Mnemonics	Op Codes	Timing
PS A,(loc)	107 LSP	5.70
PS B,(loc)	116 LSP	5.70
PS C,(loc)	126 LSP	5.70
PS D,(loc)	136 LSP	5.70
PS E,(loc)	146 LSP	5.70
PS H,(loc)	156 LSP	5.70
PS L,(loc)	166 LSP	5.70

These instructions store the specified register in the memory location specified by the LSP given in the instruction and the MSP given in the X register.

DOUBLE PAGED LOAD — DPL

Mnemonics	Op Codes	Timing
DPL BC,(loc)	111 124 LSP	10.05
DPL DE,(loc)	113 144 LSP	10.05
DPL HL,(loc)	115 164 LSP	10.05

These instructions load the specified register pair from the memory locations specified by the LSP given in the instruction and the MSP given in the X register. The C,E, or L register is loaded from the specified memory location and the B, D, or H register is loaded from the next higher location.

DOUBLE PAGED STORE — DPS

Mnemonics	Op Codes	Timing
DPS BC,(loc)	111 126 LSP	9.15
DPS DE,(loc)	113 146 LSP	9.15
DPS HL,(loc)	115 166 LSP	9.15

These instructions store the specified register pair in the locations specified by the LSP given in the instruction and the MSP given in the X register. The C, E, or L register is stored in the specified location and the B, D, or H register is stored in the next higher location.

INCREMENT AND DECREMENT INDEX — INCI, DECI

Mnemonics	Op Codes	Timing
INCI (disp),(index)	0005 LSP(i)	11.70
DECI (disp),(index)	025 LSP(i)	12.00
INCI*(disp),(index)	111 005 LSP MSP(i)	15.30
DECI*(disp),(index)	111 025 LSP MSP(i)	15.60

The processor has a construct called an index which is a 16-bit value kept in memory. The concept is similar to index registers except that all the values are kept in the page of memory pointed to by the X-register. The index is specified by a single byte in the instructions (shown as (i) above) which points to the memory location containing the LSP of the index value, the MSP being in the next higher memory location ((i) specifies the LSP of the index address while the X register specifies the MSP of the index address). The instruction also contains a displacement (shown on (disp) above) that is either one or two bytes in length (depending upon the op code). These instructions either increment of decrement the value of the index by the value of the displacement. The Carry condition flag reflects the carry or borrow from the incrementation or decrementation. The rest of the condition flags are indeterminate.

LOAD FROM INDEX INCREMENTED OR DECREMENTED — LFII, LFID

Mnemonics	Op Codes	Timing
LFII BC,(disp),(index)	062 005 LSP(i)	12.30
LFID BC,(disp),(index)	062 025 LSP(i)	12.40
LFII BC,*(disp),(index)	113 005 LSP MSP(i)	13.80
LFID BC,*(disp),(index)	113 025 LSP MSP(i)	13.90
LFII DE,(disp),(index)	174 005 LSP(i)	12.30
LFID DE,(disp),(index)	174 025 LSP(i)	12.40
LFII DE,*(disp),(index)	115 005 LSP MSP(i)	13.80
LFID DE,*(disp),(index)	115 025 LSP MSP(i)	13.90
LFII HL,(disp),(index)	176 025 LSP(i)	12.30
LFID HL,(disp),(index)	117 005 LSP MSP(i)	12.40
LFII HL,*(disp),(index)	117 005 LSP MSP(i)	13.80
LFID HL,*(disp),(index)	117 025 LSP MSP(i)	13.90

These instructions are similar to the INCI and DECI instructions except that they load the specified pair of registers with the result of adding or subtracting the displacement to or from the index value of the index. The condition flags are similarly affected.

5.7.6 Category 6 — Operating System Control

BASE REGISTER LOAD — BRL, BRL(r)

Op Code: 072, I r I 072
Timing: 4.85, 6.95
Length: 1 or 2 if I r I specified

This instruction loads the base register from the specified register. Note that the base register cannot be saved. For this reason, loading the base register will normally be a monitor function, allowing the monitor to keep within itself the value of the base register for user state storage purposes. This instruction will cause a privileged instruction interrupt if the USER mode flag is set. If (r) is not specified, the A register is used. This value is also written to memory in location 0167653.

NOP JUMP — NOJ

Op Code: 045
Timing: 2.55
Length: 3 bytes

This instruction causes no operation to be performed. It is useful for overstoring jump instructions which might be executed while being overstored. The procedure to overstore a jump instruction would be to first overstore the op code with an 045 (NOP jump) and then update the address portion. Then the op code could be overstored with the appropriate jump instruction. The primary use of this instruction is for overstoring the interrupt vector jump instructions for the interrupts which cannot be disabled (such as memory parity fault) and which might happen while the jump is being overstored.

SYSTEM CALL — SC

Op Code: 067
Timing: 11.30

This instruction causes the USER mode flag to be cleared, the last entry in the sector table to be set to the last 4K section of physical memory space with access protection, and a CALL performed to location 0167452 (in System RAM). It is how the user would communicate with an operating system that employed the USER mode.

USER RETURN — UR

Op Code: 111 102
Timing: 10.00

This instruction is identical to the RETURN instruction (op code 007) except that additionally the USER mode flag is set.

SECTOR TABLE LOAD **STL**

Op Code: 077
Timing: 8.00 + (C•2.60) + 0.45 if C>8
(or + 3.65 if C = 0)
Length: 1 byte

This instruction loads up to the first 15 entries in the sector table. This table contains six bits for each entry. The two right-hand bits are not used and should always be set to zero. Bit 2 is set for access enable. Bit 3 is set for write enable. The left-hand four bits are used to map that entry into a particular 4K section of physical memory space. This instruction will cause a privileged instruction interrupt if the user mode flag is set.

Entry:	HL = location of first byte in table of up to 15 to load.
	C = number of entries to load (0 to 15).
Exit:	No registers or conditions changed.

BREAKPOINT **BP**

Op Code: 052
Timing: 11.70
Length: 1 byte

This instruction is similar to a System Call (SC) instruction except the call is performed to location 0167460 of system RAM, which will cause entry into the system DEBUG routine if the Sector location is not changed.

ENABLE INTERRUPTS AND JUMP **EJMP**

Op Code: 111 050
Timing: 9.05
Length: 4 bytes

This instruction is identical to the Enable Interrupts (EI) instruction except that additionally a jump is performed to the (LSP, MSP) address.

ENABLE INTERRUPTS AND RETURN **EUR**

Op Code: 062 050
Timing: 11.05
Length: 2 bytes

This instruction is identical to the combination of the Enable Interrupts, Set USER Mode Flag and Return instructions.

5.7.7 Category 7 — 6600 Type Instructions

LOAD REGISTER FROM MEMORY **LRM**

Op Code: rp 3d7
Timing: 5.00 (6.95 if imp-reg specified)

This instruction (in 2200 instruction set) will actually complete the load of memory contents into the specified register before memory fault/protection is checked. This allows violation of access protection as well as testing memory containing parity faults (and being able to read the data). If a violation occurs, the standard vector (as in other instructions) is called.

DOUBLE PAGED LOAD REVERSED **DPLR(rp) ,loc**

Mnemonic	Op Code
DPLR BC,loc	062 114 LSP
DPLR DE,loc	174 134 LSP
DPLR HL,loc	176 154 LSP

Timing: 10.05

These instructions load the specified register pair from the memory locations specified by the LSP given in the instruction and the MSP given in the X register. The B, D, or H register is loaded from the specified memory location and the C, E, or L register is loaded from the next higher location. Note that this is similar to the 5500 DPL instruction except that the order in which the registers are loaded is reversed.

DOUBLE PAGED STORE REVERSED **DPSR(rp) , loc**

Mnemonic	Od Code
DPSR BC,loc	062 116 LSP
DPSR DE, loc	174 136 LSP
DPSR HL,loc	176 156 LSP

Timing: 9.15

These instructions store the specified register pair into the locations specified by the LSP given in the instruction and the MSP given in the X register. The B, D, or H register is stored into the specified memory location and the C, E, or L register is stored into the next higher location. Note that this is similar to the 5500 DPS instruction except the order in which the registers are stored is reversed.

SECTOR TABLE LOAD STARTING AT OFFSET — STLO(r)

Mnemonic	Op Code
STLOA	022 077
STLOB	111 077
STLOC	062 077
STLOD	113 077
STLOE	174 077

Timing: 10.45 + (C•2.60) + 0.45 if C>B
5.75 if C = 0

The Sector Table in the 1800 contains eight bits for each entry. Bit 0 of a Sector Table entry is explained below; bit 1 of a Sector Table entry contains a hardware generated and checked parity bit. Any value loaded into this bit position is ignored since the hardware generates the proper parity bit when a Sector Table entry is loaded.

If, during any memory access, the number of one bits out of eight Sector Table entry bit positions is incorrect, a Sector Table Parity Error System Call interrupt will be generated to memory location 0167474.
Bit 2 of a Sector Table entry is set to enable the sector to be read or written when the machine is in USER mode. Bit 3 is set to enable the sector to be written in either USER or SYSTEM mode. Bits 4 through 7 are used for physical memory address bits 12 through 15, and bit 0 is used for physical memory address bit 16 — giving the 1800 17 bits of physical memory address to accommodate a possible 128K of physical memory space.

STLO(r) is similar to the 5500 STL instruction except that the upper four bits of the specified register (A, B, C, D, or E) determine where in the Sector Table the loading is started (the lower four bits can be any value). For example; if the STLOA instruction is performed with the A register containing a 060 (octal) and the C register containing a 5, Sector Table entries 3 through 7 will be loaded.

Note that though sector table loads can wrap around through the top entry, the top entry always points to the system ROM sector at 0170000 through 0177777, is not access enabled, and not write enabled. This condition is automatically forced at the end of all sector table loads.

Entry:	HL = location of first byte in a table of up to 15 Sector Table entries to be loaded.
	C = number of entries to be loaded (0 through 15; the upper 4 bits of C can be any value).
	(r) = starting Sector Table entry (upper four bits 0 through 15; and lower 4 bits of (r) can be any value; (r) can be A, B, C, D, or E).
Exit:	Sector Table loaded, no registers or condition flags changed.

5.7.8 Instructions New to the 1800

PROCESSOR TYPE INFORMATION — INFO

Op Code: 111 010
Timing: 5.60

This instruction is used to differentiate the 1800 from other Datapoint processors. In the 5500 this instruction performs no operation. In the 1800 it loads a 2 into the A register and the revision number of the micro-code ROM into the B register. None of the condition code flags and none of the other registers are affected by this instruction. To determine the type of Datapoint processor, the following sequence is suggested:

```
XRA                    determine if 2200
LLA
LHA
DECP    HL             (this is a NOP on a 2200)
JFC     IMA2200        I'm a 2200
XRA                    determine if 5500 or 1800
INFO
CP      1
JTC     IMA5500        I'm a 5500
JTZ     IMA6600        I'm a 6600
CP      3
JTC     IMA1800        I'm an 1800
JMP     IMA????
```

PROCESSOR TYPE CAPABILITIES — INFOx

Mnemonic	Op Code
INFO2	062 010
INFO3	113 010
INFO4	174 010
INFO5	115 010
INFO6	176 010
INFO7	117 010
INFO8	022 010

Timing: 6.15

These instructions return control bits which define the capabilities of the processor, including definitions of in-board communications modules, interface capabilities, interface code for special devices and other future expansions products. The result is always returned in registers B, C, D, and E. At present they are always zero except in the INFO2 B register, where eight control bits have been defined as follows:

B0	Micro-bus instructions available
B1	reserved
B2	Dual density diskette instructions available
B3	zero
B4	reserved
B5	reserved
B6	standard I/O bus and interface instructions available
B7	internal communication instructions available

Note that this instruction on a 2200, 5500 or 6600 type processor will not change registers as described. On the 2200 and 5500 a NOP is done; on the 6600 the operation is like the INFO instruction, changing the A and B registers.

SYSTEM STATUS INFORMATION **SYSTATx**

Mnemonic	Op Code	Timing	Status
SYSTAT1	111 157	5.85	PSW
SYSTAT2	062 157	6.25	SNID
SYSTAT3	113 157	6.25	* reserved *
SYSTAT4	174 157	6.25	* reserved *
SYSTAT5	115 157	6.25	* reserved *
SYSTAT6	176 157	6.25	* reserved *
SYSTAT7	117 157	6.25	* reserved *
SYSTAT8	022 157	6.25	* reserved *

System status is designed for future expansion. The B and C registers are loaded with zero for most of the instructions, excepting the first two at present. SYSTAT1 returns in the B register the value of the internal Processor Status Word, containing bits defining Alpha/Beta Mode, Interrupts Enabled/Disabled, User/Privileged Mode, etc., as described below. For SYSTAT2, the SNID switch register value is returned in the B register; it is used by the communication subsystem and other modules as needed.

LOAD CHARACTER FONT **LODCF**
Op Code: 155
Timing: 22.50 + (A●0.15) + (A/10)●0.4
(subtract-1 if A ≠ zero and A MOD 10 = zero)

The character font for the code given in the A register (0 to 0177) is loaded into the RAM/font memory. The HL register pair points to a 7-byte list in memory specifying the font. Each row of a font is on one of the bytes with HL pointing to the font for the bottom row, HL+1 for the next row up and HL+6 for the top row of the character font. HL is left undefined after the instruction is executed; the condition flags are not changed.

PERFORM AUDIO **ACDO**
Op Code: rp 151
Timing: 5.75 (if no audio in progress)
5.20 (if audio in progress)

The register pair points to a string to be used for audible sound generation. It is not used if sound generation is already in progress. See Appendix D for further discussion of audio instructions.

PERFORM AUDIO OVERRIDE **ACDOO**
Op Code: rp 153
Timing: 6.05

The register pair points to a string to be used for audible sound generation. This instruction is illegal in USER mode, and in privileged mode it forces the string pointed to by the register pair to be used for sound generation.

AUDIO CLICK FROM ROM **CLICKR**
Op Code: 111 153
Timing: 6.00

Performs the same operation as EX CLICK except no references to RAM are made by the channel command words. This instruction is used for ROM error routines which must assume that even the memory used by the regular EX CLICK is bad.

ALPHA & BETA, SAVE & LOAD SUB-INSTRUCTIONS

Mnemonic	Op Code	Timing
SYSSAV	062 020	19.35 (-0.10 if Beta saved)
ALPHAL	111 030	21.70
BETAL	111 020	21.70

These are extensions of the ALPHA and BETA instructions. The 1800 processor has only one set of "User Registers": A, B, C, D, E, H, L, X, and the condition flags. Consequently, Alpha/Beta mode switching is done by storing the contents of this set of registers in the System Save Area of memory while loading the contents of the other set from System Save Area to replace it, when switching to the opposite mode. The SYSSAV instruction saves the contents of whichever register set is currently in use.

SYSTEM RETURN **SYSRET**
Op Code: 062 030
Timing: 30.55 (+0.05 if to Beta)

All system calls (BP, SC, 1-millisecond, Memory Fault, etc.) cause to be stored in the System Save Area an internal status word containing bits that define Alpha/Beta mode, Interrupts Enabled/Disabled, Privileged/User Mode and other control information. The System Return instruction will reload this control status word into the internal control register, as well as reload the correct register set (Alpha or Beta), thereby destroying previous contents of the user registers regardless of which mode the system was in. In addition, it will perform a Return operation, popping the return address off the stack. This privileged instruction is useful for 1-millisecond interrupt routines.

SYSTEM SAVE AREA MOVE **SYSMOV**

Op Code: rp 065
Timing: 7.35

The System Save Area contains the 32-entry-deep stack, the Alpha and Beta register sets, and the saved Internal Status Word. This area is on a 128-byte boundary with the last 64 bytes used for the stack and some of the rest used for the other saved information — Processor Status Word, Alpha and Beta Register Save Areas.

SYSMOV allows the Save Area to be moved anywhere in memory, provided that the memory location physically exists, is access-enabled and writeable. On entry, the register pair must point to a new System Save Area and to the new top of Stack. On exit, the register pair points to the old System Save Area, specifically the old top of Stack. To enforce correctness of the given stack pointer, Bit 6 of the LSB is forced to a one, and Bit 0 to zero. This ensures that the stack pointer given is in the correct half of the 128-byte System Save Area.

SYSMOV aids in fast state-swapping by obviating the STKS and STKL instructions which do approximately the same things but take much longer.

Note that the SYSSAV, SYSMOV combination will save all state information and SYSMOV, SYSRET — if entered from interrupt — will restore it between multiple tasks.

The System Save Area

The following diagram can be used as an aid in understanding the memory allocation of the System Save Area:

0177	
	Stack Area (32 two-byte entries)
0100	
0077	Program Status Word
0076	Alpha Flags Save byte
0075	Alpha X register Save byte
	Alpha B, C, D, E, H, and L Save Area
0066	Alpha A register Save byte
0065	Beta Flags Save byte
0064	Beta X register Save byte
	Beta B, C, D, E, H, and L Save Area
0055	Beta A register Save byte
0054	
	unassigned
0000	

Program Status Word

The Program Status Word has these bit definitions:

Bit 0	Interrupt Enabled
Bit 1	Base Register Disabled
Bit 2	User Mode
Bit 3	reserved
Bit 4	I/O Bus in Data Mode
Bit 5	Repeated Instruction (internal)
Bit 6	reserved
Bit 7	Beta Mode

PART 6
INPUT/OUTPUT

6.1 General

The Datapoint 1800 communicates with and exercises program control over external devices via an Input/Output System Bus. All external devices are connected to the I/O Bus in parallel, "daisy chain" fashion.

Each external device is assigned an Input/Output 'address" unique to it alone. One device at a time is designated by the processor as currently addressed, and only communication between the processor and that device is possible. All others on the I/O Bus are logically, although not electrically, disconnected from the I/O Bus.

The 1800 does not supply power, except on four +5 V I/O bus pin connectors for I/O bus drivers. The I/0 +5 is fused at two amps. A light-emitting diode, visible from the rear with the housing installed, indicates when the fuse is blown.

WARNING!

Note particularly that the standard Datapoint I/O Bus *cannot* support peripheral devices that lack their own independent power supplies. Auxiliary power supplies for peripherals are *required* for proper operation of the 1800.

6.2 Input/Output Physical Connections

The Input/Output System Bus connector on the 1800 is a 50-pin receptacle with female contacts, plus provisions for screw lock assemblies.

Each external device has two 50-pin Input/Output Bus connectors; one a female plug with male contacts labelled "I/O Bus In" and the other a male plug with female contacts labelled "I/O Bus Out." Both of these connectors have provisions for screw lock assemblies.

Datapoint Universal Input/Output cables have a male connector at one end and a female at the other.

Connection is made from the 1800 I/O connector to the "I/O Bus In" connector of an external device via a Universal I/O Cable. If more than one device is connected to the I/O Bus, connection is made from the "I/O Bus Out" connector of the first device to the "I/O Bus In" connector of the second device with a Universal I/O Cable. The process is repeated for other external devices; hence the term "daisy chain."

Every external device must connect each of the 50 pins (including spares) of its "I/O Bus In" connector to the corresponding pins of its "I/O Bus Out" connector in addition to connection to those lines required for the particular device. This is required for continuity of all signal, power and ground lines on the I/O Bus.

APPENDIX A
1800 DISKETTE DRIVE DESCRIPTION

CHAPTER 1. SOFTWARE INTERFACE

1.1 INTRODUCTION

The Datapoint 1800 communicates to the diskette through a 26 conductor flat cable from the rear of the 1800. The 1800 diskette drive is a two drive system capable of operating under both single and double density modes of operation for a capacity of 256K bytes (each drive) in single density mode or 512K bytes in double density mode. Qualified, double density diskette media must be used for optimum performance. All the commands described below are privileged and will cause an interrupt if executed in User Mode.

1.2 NORMAL DISKETTE INTERFACE LEVEL

The high level interface to the diskette presented in this section is the only one that should be used in normal operation (the lower level is useful for troubleshooting). It allows both double and single density diskettes to be written and read without having to deal directly with the conversion from single density data to double or vice versa. The only exceptions to this rule have to do with getting the hardware device status and performing head movement; these are described in following sections under the instructions UBIN (FDSTAT) and UBOUT (FCOMOD), respectively.

Master Reset (FDSCLR 0111, 0143)

The operation of the master clear instruction is identical to that of the System Status command below but it also performs a reset to internal control registers and resets the Busy status bit. This should be done only at power up or when the diskette system is first used in an operating system. It may also be necessary if during a diskette I/O operation (FDCMD) the diskette is 'popped out'. The operation is left hanging and the system will not allow FDCMD and FDDATA operations to occur until the FRBUSY flag is turned off (these are all described below).

System Status (FDSTAT 0143)

The system status command returns the present status of the high level diskette interface. This status is read by executing a 0143 command in the 1800 processor. The system status should be read to determine whether it is safe to issue a new command or if the present one has completed. The status is presented in the A register and the present track in the D register with the last sector read in the E register.

Use of the values given in the DE registers should take into consideration that some types of command errors will not give correct values in the DE pair. If an error occurred which prevented the reading of a header — as, for example, if the drive were "off-line" — the values in the DE registers would be invalid. Note that the error bit (2) is zero while busy is true. This bit only reflects the fact that the operation was not able to complete correctly after 4 index marks had been seen (the FFSYNC operation should always end this way). Reading the device status (FCINST) can aid in explaining and extending the System Status.

The status bits returned in the A register from FDSTAT are:

BIT ASSIGNMENTS	LABEL	DESCRIPTION
0	FRDRV	— 0 = right hand drive 1 = left hand drive
1	FRBUSY	— Busy (operation in progress)
2	FRINDX	— Operation did not complete before 4 index pulses were seen.
5 4 3	FFfcn	— FDCMD function bits of the last operation attempted.
6	FRIXCT	— Index pulse counter**
7	FRIXCX	— Index pulse counter**

**(To allow 1800 program to determine whether enough diskette rotations have occurred for operation to complete.)

Buffer Control (FDDATA 0141)

There are 8 subcommands in the buffer control group that allow operations to be performed on the diskette controller buffer, and a way of extending these to include deleted data operations. These higher level commands perform all buffer pointer setting, and status checking for the operations being performed. The buffer control commands are performed by executing a 0141 instruction with the value of the B register determining the subfunction. The contents of the A register are used to specify the device address (only 0 through 016 are valid). Registers C, D, and E are used; registers H and L must point to a memory location.

LABEL	"B" Reg	Subfunction	Timing
FDINS	000	Input data (single density)	1527.85
FDIND	010	Input data (double density)	1978.20
FDVRS	020	Check CRC of buffer data (single density)	1440.60
FDVRD	030	Check CRC of buffer data (double density)	1799.00
FDOTS	040	Load buffer (single density)	1419.25
FDOTD	050	Load buffer (double density)	1758.45
FDWPI	060	Load preamble (needed for writes only)	59.45
FDOUT	070	Output data without initialization	14.70
FDLTD	004	Deleted data modifier (for first 6)	+ (C●5.70)

Note: When starting any of these operations the least significant 2 bits of the B register must be 00. If the operation is completed successfully the lower two bits will be 11 when done.

Also note that FDLTD is added to any of the first six operations (making them 004 to 054) so that the operation will work on IBM-compatible Deleted Data Sectors.

INPUT DATA (FDINS, FDIND)

This command will read 128/256 bytes from the diskette buffer into memory, starting in the location specified by the H and L registers. The CRC is accumulated and checked as the data is read from the buffer.

Check CRC of Buffer Data (FDVRS, FDVRD)

The CRC is accumulated and checked on the buffer data without transferring it from the buffer to memory. This allows a sector to be written, read, and verified without destroying the copy in memory in case it needs to be rewritten. For this instruction, the H and L registers must point to a location in memory as if a FDINS/D operation was being executed, because a memory location will be set if there was an error due to bad buffer data or CRC error.

Load Buffer (FDOTS, FDOTD)

This command loads 128/256 bytes from the processor memory specified by the H and L registers into the diskette buffer. The SYNC bytes (either Normal or Deleted Data) are put in front of the data and the CRC as generated is put after the data. At the end of the above operations, D and E will contain the CRC that was entered into the buffer with the data. The two byte CRC is always entered and expected to be in a form similar to that used for single density data.

Load Preamble (FDWPI)

This loads the 14 bytes of preamble into the diskette buffer for write operations, an action only necessary when writes are to be performed. If this is not done, invalid data will be written on the diskette because the hardware allows any preamble format. For this instruction, the H and L registers must point to a location in memory as if a FDINS/D operation were being executed, because the memory locations will be read. It is up to the programmer to make sure FDWPI is done before write operations on all devices (addresses 0 to 016) after every time they are powered up. If not, results can be unpredictable.

Output Data (FDOUT)

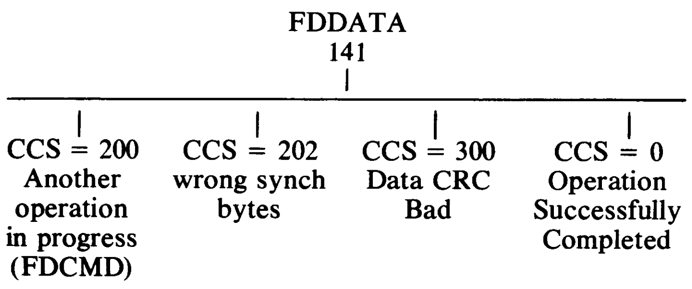
This extra operation allows outputting data to a device on the bus. It is not intended to be used on the diskette drive. The C register must be set to contain the number of bytes to be transferred. The operation FDIND can be used to input data from this non-disk device but care must be taken in the hardware/software design.

All Buffer Commands

When initiating any buffer command the lower 2 bits of the B register must be zero. For all these commands the C register is used as an internal counter and will be destroyed, except for FDOUT where C must be preset. The resulting CRC value will be in the D and E registers upon completion of the command, except for FDWPI and FDOUT.

For Input and Output buffer commands the H and L registers are used as pointers.

Upon completion of the FDDATA command, the status of the operation can be checked by executing a condition code save (CCS) instruction. This puts information about the flags (Zero, Carry, Sign and Parity) into the A register. The following status flowchart can be consulted to determine the status of the operation following an FDDATA command:


FDDATA
141
CCS = 200 Another operation in progress (FDCMD)
CCS = 202 wrong synch bytes
CCS = 300 Data CRC Bad
CCS = 0 Operation Successfully Completed

CCS / FLAG

	Z	C	S	P
0	1	0	0	0
200	1	1	0	0
202	0	1	0	1
300	0	1	1	1

If the B register low two bits are NOT zero initially, the operation will follow a well-defined pattern, but the results may seem unpredictable and possibly not what was intended.

These buffer commands are inhibited while Read and Write operations (FDCMD) are in progress, due to the use of common internal registers (notably the internal CRC registers). Attempts to violate the inhibition will generate a CCS value of 200 and an exit.

No control will prevent loading a buffer in one mode and then writing it to disk in another, i.e., loading a buffer in double density mode but writing it to disk in single density mode. The necessary consistency must be maintained by the executing program.

1.3 READ AND WRITE COMMANDS (FDCMD 0111, 0141)

In this group are eight subcommands used to read diskette data into the buffer, write the buffer data to the diskette, or convert to double density format. They are performed by executing an 0111, 0141 instruction in the 1800 processor. When these commands are executed the 1800 processor will interleave the diskette operations with executing program instructions to avoid "tying itself up."

If a Read or Write command is in progress (which could be noted by FDSTAT status bit FRBUSY), then the operation just specified is inhibited and the CCS value is 200. Normally, all flags will be false except the Zero flag, which will be true.

The following registers are used to define the read and write commands when the 0111, 0141 instructions are executed:

A Diskette controller address (0 to 016 are valid)
B Sub-command specification
C Drive Number (1 = right hand drive, 2 = left hand drive)
D Track Number (0 to 0114)
E Sector Number (0-special see description (1 to 032)

The sub-commands specified in the B register are:

LABEL	B Reg	Subcommand
FFREAD	000	Read specified sector
FFWRITE	040	Write buffer to diskette
FFSGL	000	Single Density mode operation
FFDBL	010	Double Density mode operation
FFRDLTD	020	Deleted Data Read operation
FFWDCG	060	Write DC gap
FFSYNC	070	Index mark sync-up

Read Sector (FFREAD)

This command will read the sector specified by the E register into the diskette buffer. If the track number read from the header on the diskette indicates that the head is not on the track specified by the D register the read will abort and a system status read can be performed (FDSTAT) to find the current track number. In any event, the system status must be read to determine when the operation is complete and whether or not it is successful.

A special case exists when the specified sector is zero. There is no sector zero on the diskette so specifying sector zero causes the diskette to read its next header and the data associated therewith (if on the current track) to determine where on the diskette the head is currently located — a useful procedure for optimizing diskette accesses.

This read operation can and must be extended to define which type of read is being performed (one of four). Specifying FFREAD + FFSGL generates a single density read; calling for FFREAD + FFDBL generates a specific check to ensure that the double density D.C. Gap was seen.

If FFRDLTD is also specified (FFREAD + FFSGL + FFRDLTD or FFREAD + FFDBL + FFRDLTD) the read operation is made for a Deleted Data sector. If the read operation is made for the wrong type of sector (Normal/Deleted Data) the sector will not be found and an error will occur. FFRDLTD need not and must not be specified with FFWRITE.

Write Buffer (FFWRITE)

With FFWRITE the present contents of the buffer can be written to the diskette using the drive, track, and sector specified in the C, D, and E registers. There are two versions of this command; one for double density and one for single density data (FFWRITE + FFDBL and FFWRITE + FFSGL).

Note that a D.C. gap must have been written on a sector to allow double density writes, whereas single density writes will happen even if there was a Gap on the sector; they will "over-write" the Gap, deleting it. If conversion back to single density is desired, each sector on the track should be written with zeroes before the conversion, otherwise it is possible to damage the diskette's control (header) information.

Write D.C. Gap (FFWDCG)

The D.C. gap differentiates between a double density diskette and a single density diskette. To convert a diskette to double density use, the Write D.C. gap command should be executed for *each* sector on each track of a diskette the first time it is used. Diskettes may not be a mix of single and double density sectors or tracks; the entire diskette must be single or double density. Before converting, the data on every sector of a track (be it single or double density) should be zero. Otherwise it is possible to damage control (header) information on the diskette.

Index Sync - up (FFSYNC)

This operation exists to aid in initializing a diskette to single or double density. This procedure should be as follows:

1) The buffer is initialized to single density (zeros is normal) or double density (0252 is normal) data, the preamble is also done (FDWPI), and the head is positioned to track zero.

2) The FFSYNC operation is performed and FDSTAT is tested for completion. It will be an error completion that will lock on the Index mark before sector one.

3) The sector is written in single density mode. FDSTAT must be checked not only for non-error completion but to ensure that the index counter bits are zero, signifying that no extra revolutions were necessary to find and write the sector.

4) Step three is repeated for each of the 26 sectors on a track. The sectors must be written in physically sequential order and, if there are any errors, retry must be from step two and not merely by repeating the operation for the sector in error.

5) If the diskette is to be Double Density, an equivalent to steps two, three and four must now be done to write the D.C. Gap on the diskette (FFWDCG).

6) After a track is initialized (single or double) the next track is stepped to and the procedure repeated from step two until all tracks have been written.

1.4 DISKETTE CONTROLLER COMMANDS

The diskette controller commands are a lower level of commands which deal directly with the diskette controller. Use of these commands requires considerable care and an understanding of the operation of the diskette because many restrictions and timing relationships must be met. These restrictions and relationships are presented here to provide further insight into the basic operation of the diskette system but do not need to be considered when using the higher level interface described above (except Input status and Step).

Note that indiscriminate use of these commands can cause irreparable damage to the data on a diskette. If use of these commands causes an unexpected interrupt, it will not only be ignored, but the device will be cleared to disable any further interrupts.

The actual instructions UBIN and UBOUT cannot of themselves differentiate between input and output subfunctions. Such a decision must be held and maintained by the program. For example, FCLEAR in A, data in B and UBIN can be done, but will generate unknown results. Similarly, FCINST and UBOUT can be done, but will cause two devices (controller and processor) to contend for the bus — this may cause hardware damage.

The instructions are:

Mnemonic	Op Code	Description	Timing
UBOUT	0145	Output to device using Strobe 1	3.95
UBIN	0111 0145	Input from device using Strobe 1	6.70
UBOUT2	0062 0145	Output to device using Strobe 2	6.15
UBIN2	0113 0145	Input from device using Strobe 2	6.80

Input Command (UBIN 0111, 0145)

The Input command is used to read status, data and pointers from the diskette controller. It is broken into subcommands by the contents of the processor A register which contains both the subcommand and the controller address (which allows for more than one controller). The device address is contained in the LSP of the A register and the subcommand is in the MSP part. The value read in on the input command will appear in the B register when the command completes.

Each of the following commands is executed by loading the value given in the command part of the A register.

Input Status (FCINST 000)

Reads the status of the selected drive into the B register.

Bit Assignments	Label	Description
0	FSONLN	Drive On Line (index pulses occurring)
1	FSTRIP	Transfer in progress
2	FSWDIS	Write disabled
3	FSPRO	File protected
4	FSGAP	Wide DC gap sensed
5	FSSTIP	Step in progress
6	FSTRKO	Track 0 flag
7	FSNSYN	Synch mark not found

Input Data (FCINDT 020)

Reads data from the buffer locations specified by the processor pointer. After each read the pointer is incremented automatically.

Input Disk Pointer (FCINDP 040)

Reads the value of the upper 8 bits of the disk pointer. The disk buffer is 512 bytes and the low order bit can not easily be ascertained.

Output Command (UBOUT 0145)

The output command transfers control or data information to the diskette controller. This command is used to load masks, set pointers, transfer data, etc. The A register will contain the address and subcommand just as in the Input Instruction. The B register will contain the additional control bits necessary for the command (such as values to be loaded into pointers). The following subcommands are used with the output command:

Clear Interrupt (FCLEAR 060)

The clear interrupt command resets interrupts or controls the lights on the front of the drives. If the corresponding bit in the B register is a one the interrupt is cleared or the function is performed.

Bit Assignments	Label	Description
0	FKINDX	Reset Index Interrupt
1	FKSTEP	Reset Step Complete Interrupt
2	FKTROK	Reset Transfer Complete Interrupt
3	FKTRER	Reset Transfer Incomplete Interrupt
4	FKPNTR	Reset Pointers Unequal Interrupt
5	FKRWMF	Reset Read/Write master flip-flop
6	FKLOFF	Indicator light off
7	FKLON	Indicator light on

Output Disk Pointer (FCOTDP 0100)

Loads the disk buffer pointer's most significant 8 bits (out of 9) with the contents of the B register (LSB is set to 0). The disk buffer pointer controls the buffer address used for the transfers to/from diskette to/from controller buffer.

Output Processor Pointer (FCOTUP 0120)

Loads the processor pointer's most significant 8 bits with the contents of the B register (LSB is set to 0). The processor buffer pointer controls the buffer address used for transfers to/from the processor to/from the controller buffer.

Output Sync Mark MSP (FCOSYM 0140)

Loads the contents of the B register into the MSB part of the sync register.

Output Sync Mark LSP (FCOSYL 0160)

Loads the contents of the B register into the LSB part of the sync register.

Write D.C. Gap (FCWDCG 0200)

This command will write a DC level four bytes long following the header on the diskette. This is used to convert the format of diskettes for double density operation. It must be issued within 288 microseconds of the end of a header read by FCRHDR.

Set Mode (FCOMOD 0220)

Sets operating modes that remain constant until the next Set Mode command is issued. The contents of the B register are used to set the following modes. Either Select Drive 0 or Select Drive 1 must always be specified (but not both). If this operation is used for low level control of a drive, then high level control must not be in progress on EITHER drive of a device. There is no interlock to ensure this other than by the executing program. This operation is necessary for head movement, and is the only way it can be performed under program control.

Bit Assignments	Label	Description
0	FODRO	Select Drive 0
1	FODRI	Select Drive 1
2	**FOLOAD	Load Head on selected drive
3	**FOUNLD	Unload Head
4	FOMVIN	Step IN one track
5	FOMVOT	Step OUT one track
6	**FO43LE	Select write current for tracks before 43
7	**FO43GT	Select write current for tracks after 43

**Do not use. Use the others only if no FCMD operations are in progress on the device.

Output Character (FCOUTC 0240)

Writes the contents of the B register into the buffer using the processor pointer. The processor buffer pointer is automatically incremented.

Output Interrupt Mask (FCOINT 0260)

Interrupts from the controller can be selectively enabled with this command. The contents of the B register will enable each interrupt with a corresponding one bit.

Bit Assignments	Label	Description
0	**FMINDX	Index interrupt
1	**FMSTEP	Step interrupt
2	**FMTROK	Transfer complete interrupt
3	**FMTRER	Transfer incomplete
4	**FMPNTR	Pointers unequal interrupt
5, 6, 7	**	Not used

**Do not use. Ignored if high level interface is not awaiting an interrupt.

Read Header (FCRHDR 0300)

When the next header passes under the read head (as noted by the sync code matching characters read), this command reads into the buffer using the disk pointer the number of characters specified by the B register times two. The sync characters are included in the read. Normally B will be 7 and 14 characters will be read. When the SYNC registers match the data on the disk it will be assumed that a header is under the read head and the time-out timers for controlling correct operation of the following three commands and the Write D.C. Gap command will start.

Read With Timer (FCRTIM 0320)

This will read double the number of characters specified by the B register into the buffer, using the disk buffer pointer. This command must be issued within 544 microseconds after the end of the header was found with a Read Header command. If the sync bytes (MSP & LSP) could not be found, status bit 7 (sync not found) will be set. Reading will start when the SYNC registers match data read.

Write Normal (FCWRTN 0340)

At the appropriate time after the header that was just read the controller will write double the number of bytes specified by the B register from the buffer using the disk pointer. The data to be written from the buffer must include all information to be written to the diskette (preamble, sync, data, CRC, and postamble). This will overwrite a DC gap if one exists, therefore this command should only be used to write single density data. If this command is issued too late after the header, status bit 2 (write disabled) is set. It must be issued within 288 microseconds of the end of the header.

Write Detecting Gap (FCWDGP 0360)

This command operates just like Write Normal except that it begins after the DC gap is detected. If a DC gap is not detected or the command is not issued soon enough FCINST status bit 4 (DC gap detected) will not be set. This command is used only for double density writes and must be issued within 352 microseconds of the end of the header.

APPENDIX B
MACRO ROM

CHAPTER 1. MACRO ROM DISKETTE ROUTINES

1.1 INTRODUCTION

The Macro ROM contains many useful routines for using the flexible diskette drives. However, the RAM values associated with them must be carefully assigned.

		Control Bits
FODR0	0001	Drive zero Select Bit
FODR1	0002	Drive one Select Bit
FOMVIN	0020	Move Head In (one step)
FOMVOT	0040	Move Head Out (one step)

(See discussion of Set Mode in Appendix A.)

		Data Area
$DISKBUF	0154000	DOS LF0-LF3 Diskette buffer area
$DOSEXT	0156000	DOS Extension area
$DATA$	0156600	Define Current Datapage
$LOGDRV	0156617	Current DOS Logical Drive Init -1)
$SECTOR	0156620	Current Sector
$TRACK	0156621	Current Track
$DEVADR	0156622	Diskette Module Address
$DRVNUM	0156623	Floppy Logical Drive (FODR0/FODR1)
$DRVTAB	0156624	Current Sector/Track for each DOS Logical Drive $SECTOR/ $TRACK

1.2 DISKETTE ROUTINES

$SEEK (0170162) Seek to the track in D. $TRACK contains current track (assumed to be accurate). On exit, $TRACK contains new track (assuming old value was accurate). If carry is true, the drive is off-line, or some other malfunction occurred. False carry implies that the seek was successful.

$STEP (0170165) Step in or out one track. On entry, A is set to operation (FOMVIN/FOMVOT), X to $DATA$ > 8, $DRVNUM to drive number, and $DEVICE to diskette module address. On exit, true carry indicates the drive is off-line. False carry with true zero indicates that track 0 was not detected, and false carry with false zero indicates that track 0 was detected.

$READ (0170170) Read a sector. On entry, H contains MSB of memory buffer, D the track to be read, E the sector to be read. On exit, $TRACK/$SECTOR are set to track/sector read. True carry indicates CRC or some other error; false carry that the operation successfully completed.

$WRITE (0170173) Write a sector. Entry and exit values are the same as for $READ, except that on entry A contains Write/Write Verify control — 007 for verification, 006 for no verification.

$ONLINE (0170176) Check drive on-line. On entry, $DRVNUM is set to Drive number, $DEVICE contains diskette module address. On exit, true carry means off-line and not ready. False carry with true zero indicates drive on-line and write enabled; false carry with false zero means drive on-line and write protected.

$RESTORE (0170201) Seek to track zero. On entry, $DRVNUM contains drive number, $DEVICE contains diskette module address. On exit, true carry signifies drive off-line or track 0 not found; false carry that $TRACK contains 0 and drive is at track zero.

$DOSDRIV (0170204) Convert DOS Drive to diskette module Drive. On entry, A contains the DOS logical drive (0 - 7). On exit, $LOGDRV = Selected logical drive (0 - 7), $TRACK contains Current track for that drive, $SECTOR the current sector for that drive, $DRVNUM the Drive select code (FODR0/FODR1), and $DEVADR the diskette module address (0 - 3).

Note that this routine does not do any actual diskette operations; it simply initializes all control information in memory.

Note also that the routines $DOIO and $STEP are called by all other routines to perform the necessary operations. These routines call the diskette "WAIT$" entry point (SVDISKW$ at location 0167502) while waiting for the specified operation to complete. In other words, while waiting for a step operation or a diskette read or write to complete, WAIT$ will be called at least once.

CHAPTER 2. ROM DEBUGGER FACILITY

2.1 INTRODUCTION

Upon entering the 1800 debugger the mode of the interrupted program remains established, the proper registers are used, and all condition codes and flags are saved to be restored upon exit. By use of the 'R' debug command the user may flip the processor mode from one register set to the other. In addition, the 1800 ROM debugger provides some extra commands not present in the 5500 debugger.

2.2. 1800 DEBUG COMMAND SUMMARY

A	— Address given or last I/O device
B	— Set Breakpoint at given or current address
C	— Call the given or current address (forces system mode)
D	— Decrement the current address
E	— Continue execution of program
F	— Fetch next data byte from current I/O device
G	— Go to Data mode in the current device
*H	— Hardware Floppy Diagnostic
I	— Increment current address
J	— Perform a jump to the given or current address
K	— *Not Used*
L	— Link to address pointed to by current address
M	— Modify the current address contents
N	— *Not Used*
O	— *Not Used*
P	— Display Base register or load W/value - 0100000
*Q	— Load the sector table
R	— Switch Alpha/Beta register mode
S	— Display specified Stack item
*T	— Start memory Test
U	— Sets user mode and does an "E" command
V	— EX COM4 to last I/O device
W	— EX WRITE to last I/O device
X	— EX COM1 to last I/O device
Y	— EX COM2 to last I/O device
Z	— EX COM3 to last I/O device
a	— A register display or set
b	— B register display or set
C	— C register display or set
d	— D register display or set
e	— E register display or set
f	— Condition Flags display
g	— *Not Used*
h	— H register display or set
i	— "E" command with EI/RET
j	— Display test
K	— *Not Used*
l	— L register display or set
m	— *Not Used*
n	— *Not Used*
o	— *Not Used*
p	— Display Base register or set with C
q	— *Not Used*
r	— *Not Used*
s	— *Not Used*
t	— *Not Used*
u	— *Not Used*
v	— *Not Used*
w	— *Not Used*
x	— Register display
y	— EX STATUS
z	— EX DATA
?	— Processor and Macro ROM type/version
.	— "M" command followed by "I" command
^	— ". " using last value
#	— Clear all Break points

*Must be preceded by '12345'

2.3 SPECIAL CONSIDERATIONS

It should be noted that the memory test — called a Moving Inversion ("MOVI") test — completes in different amounts of time depending upon the size of memory installed; the 1800 memory test is quite thorough, and performs its diagnostic check in 16K memory segments.

CHAPTER 3. CRT DISPLAY ROUTINES

3.1 INTRODUCTION

The RAM Hardware interrupt vectors beginning at location 0167400 are the same as those used on the 5500/6600 processors. Note that this is not a complete list of vectors, but rather includes only locations discussed in relation to CRT display routines.

3.1.1 Group I & RAM Vectors

0167400	SVMEMP	— Memory Parity Error
0167406	SVINP	— Input Parity Error
0167414	SVOUTP	— Output Parity Error
0167422	SVWVIOL	— Write Protection Violation
0167430	SVAVIOL	— Access Protection Violation
0167436	SVIVIOL	— Privileged Instruction Violation
0167444	SVONEMS	— One Millisecond
0167452	SVSCAL	— System Call
0167460	SVBKPNT	— Break Point
0167466	SVUAINS	— Unassigned Instruction
0167474	SVSTPAR	— Sector Table Parity Error
0167502	SVDISKW$	— "WAIT$" used by diskette routines
0167505	SVDISPW$	— "WAIT$" used by display routine

3.1.2 Group II — RAM Control Locations

The locations in this group are non-executable locations which contain strings, pointers, buffers, and other control information.

0167510	SVBEEP	— Standard Beep (Op Code 0151)
0167516	SVCLIK	— Standard Click (Op Code 0153)
0167537	SEACFLG	— Audio Channel flag control marker
0167540	SEACCNT	— Audio Channel control counters
0167640	SEDOPTS	— Options for the display routine
0167644	SECCHAR	— Character used for a cursor
0167645	SECPOS	— Cursor position
0150000	SECBUF	— Display buffer

3.1.3 Group III — ROM Vectors

This group includes externally usable vectors located in system ROM.

0170000	SRPOWER	— Power On Reset (POR)
0170003	SRRSTRT	— Restart operation
0170027	SRCLICKR	— Click from ROM (No RAM Reference)
0170055	$KEYCHAR	— DOSFNC 6,2 W/O Blink "DE"
0170060	$KBDSINI	— Init Keyboard and Display info
0170063	$CHARLOD	— Load Character set subroutine
0170066	$DSPINIT	— Initialize the Display
0170071	$DISPLAY	— 1800 Display routine
0170074	$DISPDOS	— DOS compatible display routine
0170077	D$CBL	— Compute a display buffer location
0170102	D$CURSES	— Suspend the cursor
0170105	D$BLINKDE	— Blink the cursor at DE
0170110	D$BLINK	— Blink the cursor at SECPOS
0170113	DO$FNC60	— DOS Function 6, Subfunction 0
0170116	DO$FNC61	— DOS Function 6, Subfunction 1
0170121	DO$FNC62	— DOS Function 6, Subfunction 2
0170124	DO$FNC63	— DOS Function 6, Subfunction 3
0170127	DO$FNC64	— DOS Function 6, Subfunction 4
0170132	DO$FNC65	— DOS Function 6, Subfunction 5
0170135	DO$FNC66	— DOS Function 6, Subfunction 6
0170140	DO$FNC67	— DOS Function 6, Subfunction 7
0170143	DO$FNC68	— DOS Function 6, Subfunction 8
0170146	DO$FNC69	— DOS Function 6, Subfunction 9
0170151	DO$FNC6A	— DOS Function 6, Subfunction 10
0170154	DO$FNC6B	— DOS Function 6, Subfunction 11
0170157	DO$FNC6C	— DOS Function 6, Subfunction 12 *NEW*

3.2 CRT Display Routines

The 1800 Macro ROM has a built in user callable CRT display routine which is similar to the standard DOS DSPLY$, with the addition of enhanced control characters. Moreover, the entire set of DOS Function 6 is included in ROM.

3.2.1 DOS DSPLY$ Compatible Routines

Entry:	B	=	Options as given for $O (Unless entered at $DISPDOS)
	D	=	Horizontal cursor position (0-79)
	E	=	Vertical Cursor position (-12-11)
	HL	=>	Starting Display Address
Exits:	A	=	Entry Value
	B	=	Last option value set
	C	=	Entry Value
	DE	=	Cursor position after last character displayed or Cursor position when last control executed
	HL	=>	Address of byte after string terminator

A maximum of six extra stack levels are is (exclusive of display WAIT$ overhead (if used)). If the display key inhibit is not set, the WAIT$ vector is called at least once and then continuously if the display key is being pressed. This happens as each control character is about to be interpreted ($F excepted). The WAIT$ vector is at memory location 0167505.

Entry points:

$DISPDOS 0170074

$DISPDOS is a DOS DSPLY$ compatible routine which causes blanks to be skipped, non-inverted video, and will not wait when the DISPLAY key is pressed.

$DISPLAY 0170071

The entry point does not assume any options, thus entry must be made with the B register set. Control characters:

$ES	003	End of string
$BP	007	Beep
$H	011	Horizontal position follows
$V	013	Vertical position follows
$EL	015	End of string w/ CR/LF
$EEOF	021	Erase to end of frame
$EEOL	022	Erase to end of line
$RU	023	Roll up one line
$RD	024	Roll down one line
$F	033	Force display of next character
$NS	0203	New string address follows (LSB,LSB)
$CK	0207	Click
$HA	0211	Horizontal adjustment follows
$VA	0213	Vertical adjustment follows
$NL	0215	Advance to the next line
$HU	0223	Home up (Upper Left-hand corner)
$HD	0224	Home down (Lower Left-hand corner)
$O	0233	New options follow

The byte following the $O control code contains the bits of the three following options which are to be set. Once an option is set, it will remain set until an end of string is encountered, or until new options are given.

$OI	0200	Inverted video option

If set, all characters displayed will be inverted (dark characters on a light background). Note particularly that the blank character is inverted. Rolling the screen, or even clearing it, will cause a light character to appear any place the screen would be blanked out. Once turned off, video is displayed normally.

$OS	0100	Skip blanks option

This option is normally set upon entry to $DISPDOS, thereby maintaining compatibility between DOS DSPLY$ and the 1800 display routine. When set, blanks in a string (040) will not be displayed. This was intended to speed up displaying of strings on the 2200 by bypassing the overhead of displaying blanks.

$OW	040	Inhibit wait on DISPLAY key option

This option is also set upon entry to $DISPDOS. If clear, each time any control code is executed, the display key is checked. (The key is checked before the control code is executed.) The routine will hold in a tight loop until the display key is released. If set, the display key will be ignored. During waiting, a continuous call to the display WAIT$ vector occurs (see above).

3.3 Dos Function 6 Routines

The 1800 Macro ROM includes all of the DOS function 6 routines as well as the DOS function 11 character set load routine. Note that unlike standard DOS functions, the contents of some registers are destroyed upon exit. These routines, therefore, should be avoided, and the user should use the standard DOS functions whenever possible. All functions save the contents of the X register and save or manipulate the contents of DE as needed; functions 0 and 1 save B; function 2 destroys the contents of all registers except X and DE; function 3 destroys the contents of register A, and the H and L registers contain the buffer address of the character written; functions 4 and 5 and 7 through 12 save the contents of all registers; function 6 saves the contents of X and DE.

DO$FNC60	0170113
DO$FNC61	0170116
DO$FNC62*	0170121
DO$FNC63	0170124
DO$FNC64	0170127
DO$FNC65	0170132
DO$FNC66*	0170135
DO$FNC67*	0170140
DO$FNC68*	0170143
DO$FNC69*	0170146
DO$FNC6A*	0170151
DO$FNC6B*	0170154
DO$FNC6C	0170157

*On entry to this subfunction, the cursor is permanently positioned to the screen position contained in DE.

Note: DO$FNC63 does not permanently position the cursor, thereby allowing characters to be written at any time to any screen position without moving the cursor to that position, thus not affecting the current cursor position.

Note: DO$FNC6A, DO$FNC6B, and DO$FNC6C refer to DOS function 6, subfunctions 10, 11, and 12. Subfunction 12 exists only on the 1800. Upon entry to this subfunction, no registers are used; upon exit, all registers are preserved except BC. The following bits reflect the following conditions for the B register:

0 — Display Key down
1 — Keyboard Key down
2 — Keyboard character ready
3 — Last key pressed is still down

The following bits indicate the following conditions for the C register:

0 — F1 Key is down
1 — F2 Key is down
2 — F3 Key is down
3 — F4 Key is down
4 — F5 Key is down
5 — Restart Key is down
6 — Attention Key is down
7 — Interrupt Key is down

A routine called $KEYCHAR at location 0170055 is essentially the same as DOS function 6, subfunction 2, except that it will not position the cursor to DE. The routine only modifies the H, L, and A registers. This allows complete separation of KEYIN and DISPLAY functions. Blinking is considered a display operation.

3.4 Cursor Manipulation

The following sections deal with ROM subroutines which may be used to control the flashing cursor. The cursor on the 1800 will not flash on its own. Routines are available to flash the cursor on or off (see below). In order to keep it flashing, these routines must be called continually by an interrupt driven routine or a subroutine call in your loop. For example, the DOS KEYIN$ routine contains a call to the blink routine in the loop which waits for a character.

3.4.1 Character Used for a Cursor

Location SECCHAR (0167644) contains the character which is to be used as a cursor. This is normally set to a binary zero, but may be changed by user-programs to any character desired. Note that modification of this location should only be done when either cursor blinking is off, or after D$CURSES has been called. Also, at location SECHIDE (0167643) is the character which is being hidden by the cursor at the current cursor position. This character is blinked alternately with the cursor character. This location should not be modified by user programs.

3.4.2. Turning the Cursor On and Off

The routine D$CURSES (0170102) is used to temporarily turn off the cursor so that the line in memory containing the cursor can be manipulated freely. D$BLINKDE (0170105) will cause the cursor to move to and blink at the cursor position contained in DE. D$BLINK (0170110) causes the cursor to blink at the current cursor position. If the user desires the cursor to continually flash, he must continually call one of these BLINK routines. Note that if the cursor is blinked at an illegal cursor position, the cursor will be blinked, but will not be visible on the screen.

3.5 Converting Cursor Positions to Memory Locations

The routine D$CBL (0170077) changes a standard cursor position in DE to the physical memory location of that position in HL. This routine destroys only the contents of the A register. Note: This routine only changes the address specified in DE to a physical memory location returned in HL, but does not actually position the cursor. If the cursor position in DE is off the screen, the CARRY flag will be returned true, otherwise, return will be made with CARRY false.

3.6 Keyboard/Display Initialization

The following routines allow user programs to initialize values used by the 1800 keyboard/display.

3.6.1 Initializing Values

$KBDSINI (0170060) causes the entire keyboard and display system to be reinitialized. All values will be set to their normal defaults, and the POR character set and Keyboard Translate Table will be reloaded. $DSPINIT (0170066) causes only the display pointers and values to be reinitialized. The keyboard translate table is located at 0156000 (SEKTRAN) and is 128 bytes long.

3.6.2 Loading the Character Set

A routine is available ($CHARLOD at 0170063) to load the 1800 character set. The contents of all registers except X are destroyed. On entry, HL points to a character set table consisting of some number of five or six byte entries. The first byte, if present, has its sign bit set. The other seven bits contain the number of the character whose pattern follows. The other five bytes contain the five scan columns of the character pattern from left to right. Bits 6-0 of each byte contain the 7 scan rows of the character pattern from top to bottom. If the first byte of an entry is absent, the character number in B is used. B is incremented after each character is loaded. The end of the list is indicated by a value of 0200. To load the pattern for the character value 0, the B register must have been initialized to 0, and the first byte of the list will not be present (since a 0200 terminates the list).

APPENDIX C
1800 SERIAL INTERFACE MODULE

CHAPTER 1. COMMUNICATIONS MODULE

1.1 INTRODUCTION

The serial interface, or communications, module provides the facility for half-duplex BISYNC operation or full-duplex communications under any of the following common protocols, or modes; SDLC, Generalized Synchronous and ASYNCHRONOUS. Circular buffers are provided for both the transmitter and receiver, as well as special instructions to initialize the module, move data in and out of the buffers, and handle modem and automatic call unit control and status signals. Since interface to the module is similar in all four modes, the following description consists of a section high-lighting the commonalities followed by sections describing each of the modes in detail.

1.2 FEATURES COMMON TO ALL MODES

1.2.1 Transmit Buffer

The transmit buffer is a 256 byte portion of memory extending from SYSCOM (0157000) to SYSCOM + 0377. Although special buffer handling istructions make detailed knowledge of the operation of the buffer unnecessary to the programmer, the following information is provided as an aid to understanding.

The buffer is organized as 128 characters interleaved with 128 status bytes. SYSCOM + 1 contains the status byte associated with the character at SYSCOM, SYSCOM + 3 contains the status byte associated with the character at SYSCOM + 2, etc.

Upon initialization (by the SISTART instruction) the transmitter's internal buffer pointer is positioned to the first status byte, at SYSCOM + 1. The transmitter waits for the status byte to become non-zero as an indication that a character is available for transmission. (The action taken by the transmitter if a character is not available varies with the mode. Also, in some modes additional information about special handling of the character is conveyed to the transmitter through the status byte.)

When a character is available the transmitter accepts it, sets the status byte to zero (indicating readiness for a new character/status pair in that buffer position), increments its internal buffer pointer to the next status location, and starts transmitting the character. When transmission of the character is complete the procedure described above is repeated.

This sequence continues as the transmitter works its way through the buffer until it finds an available character at SYSCOM + 0376 at which point, since the buffer is circular, it sets its internal buffer pointer to SYSCOM + 1 for another pass through the buffer.

1.3 TRANSMIT BUFFER INSTRUCTION

1.3.1 Serial Interface OUT - SIOUT (062 167)

Moves the contents of the A register into the transmit buffer. Sets TZ status if transmit buffer is full. Condition flags are all altered.

Upon initialization (by the SISTART instruction) an internal buffer pointer is positioned to the first status byte in the transmit buffer. When a SIOUT instruction is executed this status byte is tested. If the status is zero the contents of the A register are placed in the character location associated with the internal pointer, an 001 byte is placed in the status location following, and the internal pointer is positioned to the next status byte. If the status byte is non-zero TRUE ZERO processor status is set and no further action is taken.

1.3.2 Serial Interface Multiple OUT - SIMOUT (113 167)

Moves the number of characters specified in the C register from the field pointed to by HL into the transmit buffer. Sets TZ status if transmit buffer is full. Condition flags are all altered.

Entry:	HL = location of first character
	C = number of characters to move
Exit:	IF TZ;
	HL = location past last character moved
	C = entry value minus number of characters moved
	IF FZ;
	HL = location past last character moved
	C = zero

1.3.3 Serial Interface Control OUT - SICOUT (167)

This instruction is identical to SIOUT except that it places an 003 byte, rather than an 001 byte in the transmit buffer status location. SICOUT conveys to the transmitter that this character requires special handling, the nature of which varies from mode to mode.

1.3.4 Serial Interface Control Multiple OUT - SICMOUT (111 167)

This instruction bears the same relationship to SIMOUT that SICOUT bears to SIOUT.

1.4 RECEIVE BUFFER

The receive buffer is a 256 byte portion of memory extending from SYSCOM + 0400 to SYSCOM + 0777. Although special buffer handling instructions make detailed knowledge of the operation of the buffer unnecessary to the programmer, the following information is provided as an aid to understanding.

The buffer is organized as 128 characters interleaved with 128 status bytes. SYSCOM + 0401 contains the status byte associated with the character at SYSCOM + 400, SYSCOM + 0403 contains the status byte associated with the character at SYSCOM + 0402, etc.

Upon initialization (by the SISTART instruction) the receiver's internal buffer pointer is positioned to the first character, at SYSCOM + 0400. When a character is received the receiver places the character in the buffer location pointed to by its internal pointer, increments the pointer to the next location, places a non-zero status byte in that location, and increments the pointer to the next location. (In some modes the status byte contains not only a "ready" bit but additional information about any special significance of the associated character.)

This procedure is repeated for each character received until the receiver places the character/status pair in SYSCOM + 0776/SYSCOM + 0777, at which point, since the buffer is circular, it positions its internal pointer to SYSCOM + 0400 for another pass through the buffer.

Note that the receiver does not check for an over-run condition: that is, it places a character/status pair in the buffer without checking to see if the pair that was already there have been taken and the status byte set to zero. Thus *it is the responsibility of the programmer to see that he never "gets behind" the receiver by more than 128 characters.*

1.5 RECEIVE BUFFER INSTRUCTIONS

1.5.1 Serial Interface IN - SIIN (163)

Moves one character from the receive buffer to the A register. Sets TZ if buffer is empty. Sets TS and TP in some modes. Condition flags are all altered.

Upon initialization (by an SISTART instruction) an internal buffer pointer is positioned to the first status byte in the receive buffer. When a SIIN instruction is executed the status byte is tested and used to set the condition flags. If the byte is non-zero the associated character is placed in the A register. If the byte is zero, no action is taken.

1.5.2 Serial Interface Multiple IN - SIMIN (062 163)

Moves the number of characters specified in the C register from the receive buffer to the field pointed to by HL. Sets TZ if buffer is empty. Sets TS and TP in some modes. Condition flags are all altered.

Entry:	HL = location of first destination character
	C = number of character to move
Exit:	IF TZ OR TS
	HL = location past last character moved
	C = entry value minus number of characters moved
	IF FZ AND FS
	HL = location past last character moved
	C = zero

1.6 COMMUNICATIONS INITIALIZATION INSTRUCTIONS

1.6.1 Serial Interface START - SISTART (165)

Transmit and receive buffers must be zeroed before command executes. This instruction is used to select a mode (SDLC, BISYNC, GENERALIZED SYNCHRONOUS, ASYNCHRONOUS) or deselect all modes (disable the module). The selection is controlled by the contents of the A register. Some modes also require additional information to be supplied in the B register.

The internal pointers used by the transmitter, receiver, and buffer handling instructions are initialized, the transmitter output is set to MARKing, and the receiver is conditioned to look for whatever starting condition the mode requires (SYN character, START ELEMENT, etc.).

Note that the communications module may be completely disabled and the transmitter output held MARKing by executing an SISTART instruction with 000 in the A register.

1.6.2 Serial Interface SYNChronize - SISYNC (111 165)

This instruction conditions the receiver to look for whatever starting condition the mode requires (SYN character, START ELEMENT, etc.).

1.7 MODEM AND ACU INSTRUCTIONS

1.7.1 Serial Interface MODem IN - SIMODIN (161)

Returns the modem status signals in the A register:

BIT

0 = 0
1 = 0
2 = 0
3 = CLEAR TO SEND
4 = 0
5 = RECEIVED LINE SIGNAL DETECTOR
6 = RING INDICATOR
7 = 0

Sets True Zero if transmit buffer is empty. Sets True Carry if in SDLC mode activity has occurred on the receiver input since the last execution of an SIMODIN instruction. All condition flags are altered.

BIT 3 — CLEAR TO SEND: When this bit is a 1 the modem's CLEAR To Send status line is "ON". Clear To Send must be "ON" to enable the transmitter; when it is "OFF" the transmitter will not accept characters from the transmit buffer.

BIT 4 — DATA SET READY: When this bit is a 1 the modem's Data Set Ready status line is "ON" indicating that the modem is connected to a communication channel, the modem is not in test, talk, or dial mode, and any modem timing functions or answer tone transmissions are complete. The communications module takes no action based on this signal.

BIT 5 — RECEIVED LINE SIGNAL DETECTOR: When this bit is a 1 the modem's Received Line Signal Detector status line is "ON", indicating that the modem is receiving a signal which meets its suitability criteria. This signal must be "on" to enable the receiver. If it is off, the receiver will not accept data on the receive communications channel.

BIT 6 — RING INDICATOR: When this bit is a 1 the modem's Ring Indicator status line is "ON", indicating that a ringing signal is being received on the communications channel. This signal is typically "ON" for about 2 seconds and "OFF" for about 4 seconds during ringing. The communications module takes no action based on this signal.

NOTE: See EIA Standard RS-232-C and/or the manual for your modem for more detailed information on the above mentioned status signals.

1.7.2 Serial Interface MODem OUT - SIMODOUT (062 161)

Sets the modem control signals according to the A register:

BIT

0 = REQUEST TO SEND
1 = DATA TERMINAL READY
2 = 0
3 = NEW SYNC/RATE SELECT
4 = BREAK/SPARE
5 = 0
6 = 0
7 = 0

BIT 0 — REQUEST TO SEND: Setting this bit to a 1 turns the modem's Request To Send control signal "ON" and is used to condition the modem for data transmission and, on half duplex channels, to control direction of data transmission. Request To Send must be "ON" to enable the transmitter; when it is "OFF" the transmitter output is held MARKing.

BIT 1 — DATA TERMINAL READY: Setting this bit to a 1 turns the modem's Data Terminal Ready control signal "ON" and is used to control switching of the modem to the communications channel. Data Terminal Ready must be "ON" to enable the transmitter; when it is "OFF" the transmitter output is held MARKing.

NEW SYNC: Setting this bit to a 1 turns the modem's New Sync control signal "ON." This signal is used with some types of synchronous modems when communicating with several different remote modems on a common communications channel. It causes no action in the communications module.

RATE SELECT: Setting this bit to a 1 turns the modem's Data Signal Rate Selector control signal "ON." This signal is used to select between two data signaling rates in some dual rate synchronous modems and to select between two ranges of data signaling rates in some asynchronous modems. It causes no action in the communications module.

BIT 4 — BREAK: Setting this bit to a 1 forces the transmitter output SPACING to provide the BREAK function used in some asynchronous systems. Note that any characters present in the transmit buffer when this bit is turned on may be garbled or lost! It causes no action in the communication module.

NOTE: See EIA Standard RS-232-C and/or the manual for your modem for more detailed information on the above mentioned control signals.

1.7.3 Serial Interface ACU IN - SIACUIN (111 161)

Returns the ACU (Automatic Calling Unit) or supplementary modem status signals in the A register:

BIT	ACU USAGE/SUPPLEMENTARY MODEM USAGE
0 =	PRESENT NEXT DIGIT/SECONDARY SIGNAL DETECT
1 =	DATA LINE OCCUPIED/SECONDARY CLEAR TO SEND
2 =	CALL ORIGINATION STATUS/SECONDARY RECEIVED DATA
3 =	ABANDON CALL AND RETRY/SIGNAL QUALITY DETECTOR
4 =	POWER INDICATION
5 =	SPARE
6 =	0
7 =	0

1.7.3.1 ACU USAGE

BIT 1 — DATA LINE OCCUPIED: When this bit is a1the ACU's Present Next Digit status signal is "ON," indicating that the ACU is ready to accept the next digit. The communications module takes no action based on this signal.

BIT 1 — DATA LINE OCCUPIED:When this bit is a 1 the ACU's DataLineOccupiedstatus signal is "ON," indicating that the communications channel is in use. When Data Line Occupied is "OFF" a call may be originated provided that Power Indication is 'ON'. The communication module takes no action based on this signal.

BIT 2 — CALL ORIGINATION STATUS: When this bit is a 1 the ACU's Call Origination Status status signal is "ON," indicating that the ACU has completed its call origination functions and transferred control of the communications channel to the modem. The communications module takes no action based on this signal.

BIT 3 — ABANDON CALL AND RETRY: When this bit is a 1 the ACU's Abandon Call and Retry status signal is "ON," indicating that there is a high probability that the connection to the remote modem cannot be successfully established. The communications module takes no action based on this signal.

BIT 4 — POWER INDICATION: When this bit is a 1 the ACU's Power Indication status signal is "ON," indicating that power is available to the ACU. The communications module takes no action based on this signal.

NOTE: See EIA Standard RS-366 and/or the manual for your automatic calling equipment for more information on the above mentioned status signals.

1.7.3.2. SUPPLEMENTARY MODEM USAGE

BIT 0 — SECONDARY SIGNAL DETECT: This bit is equivalent to SIGNAL DETECT, SIMODIN bit 5, except that it indicates the proper reception of a secondary channel received line signal. The communications module takes no action based on this signal.

BIT 1 — SECONDARY CLEAR TO SEND: This bit is equivalent to CLEAR TO SEND, SIMODIN bit 3, except that it indicates the availability of the secondary channel instead of indicating the availability of the primary channel. The communications module takes no action based on this signal.

BIT 2 — SECONDARY RECEIVED DATA: Data received no secondary channel. The communications module takes no action based on this signal.

BIT 3 — SIGNAL QUALITY DETECTOR: When this bit is a 1 the modem is indicating that there is no reason to believe that an error has occurred. When it is a 0 the modem is indicating that there is a high probability of an error. The communications module takes no action based on this signal.

NOTE: See EIA Standard RS-232-C and/or the manual for your modem for more detailed information on the above mentioned control signals.

1.7.4 Serial Interface ACU OUT - SIACUOUT (113 161)

Sets the ACU or supplementary modem control signals according to the A register.

BIT	ACU USAGE/SUPPLEMENTARY MODEM USAGE
0 =	DIGIT 1
1 =	DIGIT 2
2 =	DIGIT 4
3 =	DIGIT 8
4 =	DIGIT PRESENT/SECONDARY TRANSMITTED DATA
5 =	CALL REQUEST/SECONDARY REQUEST TO SEND
6 =	0
7 =	0

1.7.4.1 ACU USAGE

BITS 0, 1, 2, & 3 & DIGIT 1, 2, 3, & 8: Setting any of these bits to a one turns the corresponding ACU Digit Signal Circuit control signal "ON." These signals are used to pass digits (in BCD) and control information (014 for EON, or End Of Number, and 015 for SEP, or SEParator) to the ACU. These signals cause no action in the communications module.

BIT 4 — DIGIT PRESENT: Setting this bit to a 1 turns the ACU's Digit Present control signal "ON," indicating to the ACU that it may read the code on the Digit Signal Circuits. This signal causes no action in the communications module.

BIT 5 — CALL REQUEST: Setting this bit to a 1 turns the ACU's Call Request control signal "ON." It is used to request the ACU to originate a call and to hold the connection until Call Origination Status comes on. It causes no action in the communications module.

NOTE: See EIA Standard RS-366 and/or the manual for your automatic calling equipment for more information on the above mentioned status signals.

1.7.4.2 SUPPLEMENTARY MODEM USAGE

BIT 4 — SECONDARY TRANSMITTED DATA: This bit is used to send data on the secondary channel. It causes no action in the communications module.

BIT 5 — SECONDARY REQUEST TO SEND: This bit is equivalent to REQUEST TO SEND, SIMODOUT BIT 0, except that it requests the establishment of the secondary channel instead of requesting the establishment of the primary channel. It causes no action in the communications module.

NOTE: See EIA Standard RS-232-C and/or the manual for your modem for more detailed information on the above mentioned control signals.

1.8 SDLC MODE

1.8.1 Initialization of SDLC Mode

Initialization of SDLC Mode is accomplished by executing the SISTART instruction with the A register containing either 003 ("ADDRESS DETECT" disabled) or 013 ("ADDRESS DETECT" enabled).

1.8.2 DSLC Receive

In SDLC mode the receiver deletes inserted zeros, detects FLAGs and ABORT sequences, and checks CRC.

Execution of a SISTART or a SISYNC Instruction conditions the receiver to look for a FLAG (01111110) followed by either a non-FLAG ("ADDRESS DETECT" disabled) or a character matching the address set in the address switches ('ADDRESS DETECT' enabled). (Note that an 0377 address byte, the SDLC "broadcast" address, is recognized as well as the address set in the address switches.) The first character placed in the receive buffer is the one following the FLAG. The status byte associated with this character, and all other characters until the end of the FRAME is detected, is 001. The end of the FRAME is recognized when another FLAG or an ABORT sequence (a zero followed by seven ones) is received.

If the FRAME is terminated with a FLAG the receiver checks the received CRC against the calculated CRC. If the two CRCs match, a 000 character and an 0301 byte are placed in the receive buffer. The 0301 status indicates an end of FRAME with good CRC and will cause a SIMIN instruction to terminate and either an SIIN or a SIMIN instruction to set TS (end of FRAME) and TP (good CRC). If the two CRC bytes do not match, a character consisting of a 111111xx (where x is unspecified) pattern and an 0201 status byte are placed in the receive buffer. The 0201 status indicates an end of FRAME with bad CRC and will cause a SIMIN instruction to terminate and either a SIIN or a SIMIN instruction to set TS (end of FRAME) and FP (bad CRC).

If the FRAME is terminated by an ABORT sequence a character consisting of a 1111111x (where x is unspecified) pattern and an 0201 status byte are placed in the receive buffer. The 0201 status indicates the end of a bad FRAME and will cause an SIMIN instruction to terminate and either a SIIN or a SIMIN instruction to set TS (end of FRAME) and FP (bad CRC).

1.8.3 SDLC Transmit

In SDLC mode the transmitter does zero-insertion as required to prevent data characters from looking like FLAGs or ABORT sequences and calculates the CRC.

After initialization (by a SISTART instruction) the transmitter sends MARKs until a character is available in the transmit buffer. The first character placed in the transmit buffer should be a FLAG (0176) with a 003 status byte (this may be accomplished by executing a SICOUT instruction with 0176 in the A register). The 003 status causes the transmitter to send the FLAG without zero-insertion and to initialize the CRC. After sending this FLAG, if there is no character available in the transmit buffer the transmitter will continue to send FLAGs (and keep the CRC initialized) until a character becomes available. Data characters should be placed in the transmit buffer with an 001 status byte (this may be accomplished using the SIOUT or SIMOUT instructions). The 001 status causes the transmitter to send the characters with zero-insertion and to consider a FRAME to be in progress.

Once a FRAME is in progress, a FLAG character with an 003 status will cause transmission of a FLAG preceded by the calculated CRC. The CRC is then initialized and the transmitter considers the FRAME terminated. If, while a FRAME is in progress, the transmitter finds an 003 status associated with any character other than a FLAG it sends eight ones (an ABORT pattern) and considers the FRAME terminated.

The action taken by the transmitter when no character is available in the transmit buffer depends on whether or not a FRAME is in progress. If a FRAME is in progress the transmitter sends eight ones and terminates the FRAME. If no FRAME is in progress the transmitter repeats the last eight bits it sent (either a FLAG or eight ones).

1.9 BISYNC MODE

1.9.1 Initialization of BISYNC Mode

Initialization of BISYNC mode is accomplished by executing the SISTART instruction with the A register containing either an 001 ("SYN STRIPPING" disabled)or an 011 ("SYN STRIPPING" enabled).

1.9.2 BISYNC Receive

In BISYNC mode the receiver performs character synchronization and checks CRC under control of the EBCDIC characters SYN, SOH, STX, ETX, ETB, ITB, ENQ, and DLE. Handling of transparency is also provided, including optionally stripping out DLE-SYN pairs.

Execution of a SISTART or SISYNC instruction conditions the receiver to look for two successive SYN characters to establish character synchronization. The first SYN received when establishing character synchronization is never placed in the receive buffer; the second SYN, and any SYNs following the second (including any SYNs imbedded in the block), are placed in the receive buffer only if "SYN STRIPPING" is disabled: i.e. only if the SISTART instruction used 001 in the A register. Once character synchronization is achieved, each received character is placed in the receive buffer with an 001 status byte.

CRC accumulation begins after receipt of an SOH, an STX, or a DLE-STX pair. Once the accumulation has begun it continues until receipt of an ITB, an ETB, an ETX, or an ENQ (if the CRC accumulation was started by a DLE-STX pair, indicating transparency, the aforementioned ending characters must be preceded by a DLE).

If the ending character is an ENQ the ENQ is placed in the receive buffer with an 0201 status byte and the receiver will revert to looking for two successive SYNs. The 0201 status indicates the end of a bad block and will cause a SIMIN instruction to terminate and a SIIN or SIMIN instruction to set TS (end of block) and FP (bad CTC).

If the ending character is an ETB or an ETX it is placed in the receive buffer with an 001 status byte, the next character (first CRC character) is received and placed in the receive buffer with an 001 status byte, and then the next character (second CRC character) is received and placed in the receive buffer with an 0301 or an 0201 status byte (depending on whether the received CRC was good or bad) and the receiver reverts to looking for two successive SYNs. The 0301 status indicates the end of a block with good CRC and will cause a SIMIN instruction to terminate and a SIIN or SIMIN instruction to set TS (end of block) and (good CRC).

If the ending character is an ITB it is treated the same as an ETB or ETX except that the receiver remains in character synchronization; i.e. it does not require two successive SYNs to continue.

Note that when transparency is in effect both characters of a DLE-SYN pair will be either placed in the buffer or ignored depending on whether or not "SYN STRIPPING" is enabled.

1.9.3 BISYNC Transmit

In BISYNC mode the transmitter calculates and sends the CRC under control of the EBCDIC characters SYN, SOH, STX, ETX, ETB, ITB, ENQ, and DLE. Handling of transparency is also provided, including conversion of DLEs to DLE-DLE pairs and fill with DLE-SYN pairs.

Use of the EBCDIC control characters is as described for BISYNC Receive. Except when using transparency all characters may be placed in the transmit buffer using the SIOUT or SIMOUT instructions. When using transparency an 020 (DLE) character sent as a transparent data character must be transmitted as a DLE-DLE pair. A DLE character with a 001 status byte in the transmit buffer will be transmitted as such a pair. In order to send a single DLE as a control character, preceding an ETB for instance, the associated status byte must be 003. This can be accomplished using the SICOUT instruction.

1.10 GENERALIZED SYNCHRONOUS MODE

1.10.1 Initializing GENSYNC Mode

Initialization of GENSYNC mode is accomplished by executing a SISTART instruction with the A register contents as follows:

BIT

0 = 0
1 = 0
2 = 0
3 = 0 for "SYN STRIPPING" disabled;
    1 for "SYN STRIPPING" enabled.
4 = 0
5, 6, 7 = L where L = 8 - number of bits per character

In addition, the B register must contain the SYN character to be used for character synchronization. If the characters are less than eight bits long this character must be right (least significant bit) justified with the unused high-order bits set to zero.

1.10.2 GENSYNC Receive

In GENSYNC mode the receiver performs only character synchronization.

Execution of a SISTART or SISYNC instruction conditions the receiver to look for two successive SYN characters to establish character synchronization. The first SYN received when establishing character synchronization is never placed in the receive buffer; the second SYN, and any SYNs following the second (including any SYNs imbedded in the block), are placed in the receive b only if "SYN STRIPPING"is disabled. Once character synchronization is achieved, each receive character is placed in the receive buffer with an 001 status byte. If the characters are less than eight bits long they are placed in the receive buffer right justified with the unused high order bits set to zero.

1.10.3 GENSYNC Transmit

In GENSYNC mode the transmitter provides only fill with SYN charactrs.

Any time the transmitter is ready for a new character and none is available in the transmit buffer it sends the SYN code specified by the SISTART instruction. There is no distinction in GENSYNC mode between SIOUT and SICOUT or between SIMOUT and SICMOUT.

1.11 ASYNCHRONOUS MODE

1.11.1 Initializing ASYNC Mode

Initialization of ASYNC mode is accomplished by executing a SISTART instruction with the contents of the A register as follows:

BIT

1 = 00 for 1 stop element per character;
    01 for 1.5; 10 for 2
2 = 1
3 = 0
4 = 0
5, 6, 7 = L where L = 8 - number of bits per character

In addition, the B register must contain a speed control value which will be used to divide the speed selected by the baud rate jumpers. This value may be anything from 1 to 127. For example, if the transmit and receive baud rate jumpers are set to 1200 baud, executing SISTART with 001 in the B register gives 1200 baud operation while executing SISTART with 004 in the B register gives 300 baud operation. Similarly, if the transmit baud rate jumper is set to 4800 baud and the receive baud rate jumper is set to 1200 baud, executing SISTART with 004 in the B register gives 1200 baud transmit and 300 baud receive operation.

RATE = (JUMPER RATE)/(VALUE IN B)

1.11.2 ASYNC Receive

In ASYNC mode the receiver provides start element and framing error (first stop element SPACEing) detection.

Executing a SISTART or a SISYNC instruction conditions the receiver to look for a start element. After finding a valid start element the receiver clocks in the number of bits specified in the SISTART instruction and then checks the first stop element for a MARK. The received character is placed (right justified with the unused high order bits set to zero) in the receive buffer with a status byte of 001 (if the first stop element is a MARK) or 0201 (if the first stop element is a SPACE). The 0201 status will cause a SIMIN instruction to terminate and a SIIN or SIMIN instruction to set TS (framing error).

Note that a continuously SPACEing line (a BREAK condition) will result in reception of continuous all zero characters with framing errors.

1.11.3 ASYNC Transmit

In ASYNC mode the transmitter provides for insertion of start and stop elements.

Any time the transmitter is ready for a new character and none is available in the transmit buffer it sends MARKs. There is no distinction in ASYNC mode between SIOUT and SICOUT or between SIMOUT and SICMOUT.

SERIAL INTERFACE CONNECTOR PIN ASSIGNMENTS

EIA	CKT	DESCRIPTION	SOFTWARE		PIN
AA		PROTECTIVE GROUND			1, 37
AB		SIGNAL GROUND			7,27,33,36
BA		TRANSMITTED DATA			2
BB		RECEIVED DATA			3
CA		REQUEST TO SEND	SIMODOUT	BIT 0	4
CB		CLEAR TO SEND	SIMODIN	BIT 3	5
CC		DATA SET READY	SIMODIN	BIT 4	6
CD		DATA TERMINAL READY	SIMODIN	BIT 1	20
CE		RING INDICATOR	SIMODIN	BIT 6	22
CF		RECEIVED LINE SIGNAL DETECTOR	SIMODIN	BIT 5	8
CG	**	SIGNAL QUALITY DETECTOR	SIACUIN	BIT 3	21
CH/CI		RATE SELECT/NEW SYNC	SIMODOUT	BIT 3	23
DA		TRANSMIT CLOCK FROM DTE			24
DB		TRANSMIT CLOCK FROM DCE			15
DD		RECEIVE CLOCK TO DTE			17
CRQ	*	CALL REQUEST	SIACUOUT	BIT 5	19
PWI		ACU POWER INDICATION	SIACUIN	BIT 4	28
DLO		DATA LINE OCCUPIED	SIACUIN	BIT 1	13
COS		CALL ORIGINATION STATUS	SINACUIN	BIT 2	16
ACR	*	ABANDON CALL AND RETRY	SIACUIN	BIT 3	21
NB1	*	DIGIT 1	SIACUOUT	BIT 0	29
NB2	*	DIGIT 2	SIACUOUT	BIT 1	30
NB4	*	DIGIT 4	SIACUOUT	BIT 2	31
NB8	*6	DIGIT 8	SIACUOUT	BIT 3	32
PND	*	PRESENT NEXT DIGIT	SIACUIN	BIT 0	12
DPR	*	DIGIT PRESENT	SIACUOUT	BIT 4	14
SBA	**	SECONDARY TRANSMITTED DATA	SIACUOUT	BIT 4	14
SBB	**	SECONDARY RECEIVED DATA	SIACUIN	BIT 2	16
SCA	**	SECONDARY REQUEST TO SEND	SIACUOUT	BIT 5	19
SCB	**	SECONDARY CLEAR TO SEND	SIACUIN	BIT 1	13
SCF	**	SECONDARY RECEIVED LINE SIGNAL	SIACUIN	BIt 0	12
		INTERNAL RECEIVE CLOCK GENERATOR			34
		INTERNAL TRANSMIT CLOCK GENERATOR			35
		AUDIO OUT GND			25
		AUDIO OUT (600 OHMS)			26
		RESERVED			9, 10
		SPARE INPUT	SIACUIN	BIT 5	11
		SPARE OUTPUT/BREAK	SIMODOUT	BIT 4	18

* ACU USAGE
**SUPPLEMENTARY MODEM USAGE

APPENDIX D
AUDIO CHANNEL USAGE

CHAPTER 1. PROGRAMMABLE BEEP AND CLICK

1.1 INTRODUCTION

The Datapoint 1800 contains a built in, externally programmable audio channel, composed of a digital-to-analog (D/A) converter in which digital information in memory is converted to analog voltages and fed to a built in speaker.

With this facility the user may program the Click and Beep as desired, including varying the pitch and loudness of these sounds.

1.2 AUDIO CHANNEL CONTROL

The audio channel is used by pointing a register pair at the string to be transmitted to the audio channel and executing a specified instruction. Note that all audio channel instructions must be executed from system mode; their descriptions follow:

PACO — op codes: < pair >, 0151

This executes the string pointed to by the register pair. If another string is currently being executed, PACO instruction will be ignored. < pair > is the register pair being used as pointers to a string location in memory; it may be 0176 for HL, 0174 for DE, 062 for BC, or 022 for XA.

PACOO — op codes: <pair>, 0153

This instruction is the same as PACO except that the string pointed to is forced into execution; if another audio command is in progress it will be terminated and the new string executed. Warning: if this instruction is executed too frequently, no sounds at all will be heard.

EX BEEP — op code: 0151

This is the standard BEEP as used on the 5500. It uses a string found at location 0167510 in system RAM, which implies that the user may change the sound of "BEEP" if desired.

EX CLICK — op code: 0153

The standard CLICK heard on the 5500. The string for this sound follows the string for a BEEP in RAM.

CLICKR — op codes: 0111, 0153

This is the same as EX CLICK but no RAM reference is used. CLICKR is used when a click is needed but possibly bad RAM locations must not be referenced.

1.3 AUDIO STRING CONSIDERATIONS

Once an audio instruction — PACO, PACOO, BEEP, CLICK, or CLICKR — is given, string execution begins. If the sign bit of the current byte is not set, the value of the string is simply sent to the D/A converter. If the sign bit is set, a control code is present. Note that bit 6 of all bytes is not used and should be set to zero; it follows that all string values lie in the range of 0 to 077 (octal).

Octal 040 is the zero voltage level. Any value less than 040 sends a negative voltage to the speaker and any value above 040 (but less than or equal to 077) sends a positive value to the speaker.

Every 100 microseconds a new value is sent to the speaker until a control code stops the audio channel. This 100 microsecond rate constitutes a 10 KHz sample rate for a 5 KHz maximum frequency. If the code to be executed following the control code is a simple value to be send to the D/A converter, both the control code and the string value will be executed in the 100 us time window.

In addition, there are 8 registers which may be used as desired. They are referenced through user programs or the audio channel string, or both. The string may execute functions to set a register or decrement it.

These registers are located at SEACCNT (0167540); each is one byte long. A special flag, located at SEACFLG (0167537), is set to zero upon execution of the terminator byte of the string (0200). This byte may be set to a nonzero value before execution of the string, and the user program may then test it and when it becomes zero, string execution is complete.

1.4 AUDIO CONTROL CODES

0200	Stop execution of the string (terminator).
0210	Jump to the location specified by the next two bytes as LSB and MSB. Continue execution from this new location.
022n	Decrement counter <u>n</u> and if not zero skip forward by the value specified in the next byte.
023n	Decrement counter <u>n</u> by 1 and if result is not zero skip backward by the value specified in the next byte.
024n	Load counter <u>n</u> with the following byte value. Note that this code, along with 022n and 023n, allows looping.
025n	Load attenuation (0 - 3). This selects the volume, where <u>n</u> is 0 to 3. Note that loudnesses 0 and 1 sound almost the same; the range of volumes is therefore: 0 or 1 = loud; 2 = louder; 3 = loudest.
0260	Set SEACFLG to the following byte value.

DATAPOINT CORPORATION

The leader in dispersed data processing ™