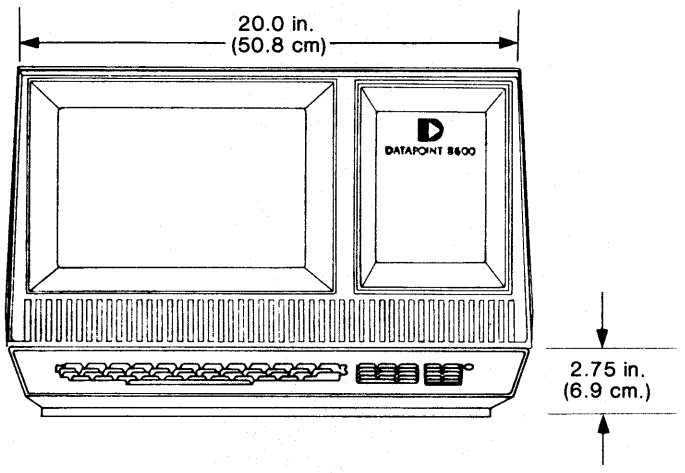
Product specification and
hardware reference manual

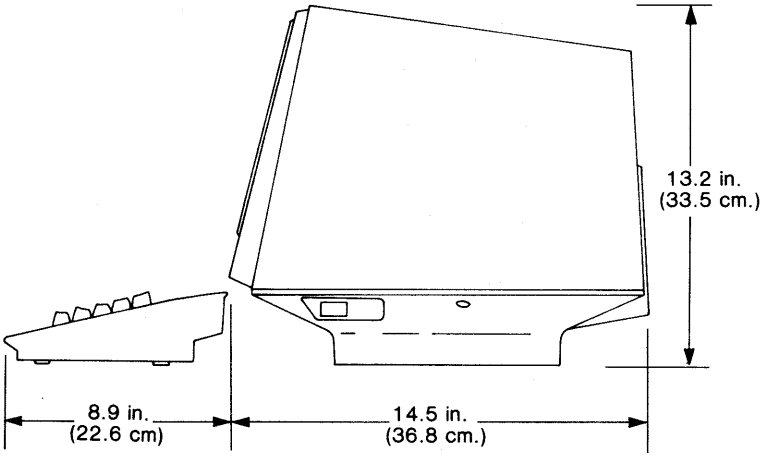
Datapoint 8600

8600 PROCESSOR

FRONT



SIDE



PREFACE

The computer-oriented user will find this manual useful in evaluating the capabilities of the Datapoint 8600 processor. However, only the hardware considerations are covered in this manual. The full utility of the Datapoint 8600 cannot be appreciated until the available software support for the machine is reviewed.

A complete family of software packages available for the 8600 processor includes items such as high-level languages, operating systems, source code and text editors, communications programs, and utility programs. Please refer to the latest issue of the Datapoint Software Catalog for the most complete information.

This page intentionally left blank.

TABLE OF CONTENTS

PART 1 GENERAL FEATURES

1.1 Introduction 1
1.2 KDS Module 2
1.3 Bus Architecture 2
1.4 Memory 2
1.5 Processor 3
1.6 Disk Interface 3
1.7 Multiport Communications Adapter (MPCA) 3
1.8 Multifunction Communications Adapter (MFCA) 3
1.9 Ancillary Equipment 3
1.10 General Specifications 3
1.11 Peripherals 3
1.12 Model Codes 3

PART 2 KEYBOARD AND DISPLAY SUBSYSTEM (KDS)

2.1 General 5
2.2 Keyboard 5
2.2.1 Keyboard Control 8
2.2.2 Special Key Sequence Controls 8
2.3 Display 8
2.3.1 Display Character Format 9
2.3.2 Video Attributes 9
2.3.3 Character Font Load 9
2.4 Serial I/O Port 10
2.5 KDS Programming Considerations 10
2.5.1 Keyboard and Screen 10
2.5.1.1 Commands 10
2.5.2 Speaker Commands 11
2.5.3 Video Attributes 11
2.5.4 Serial I/O Port 12

PART 3 BUS ARCHITECTURE

3.1 General 13
3.2 Common Bus 13
3.2.1 Interrupts 13
3.2.2 Direct Memory Access 13
3.2.3 Common Bus Signals 13
3.2.3.1 Multiplex Signals 13
3.2.3.2 Cycle Controls 13
3.2.3.3 Interrupt Control Signals 13
3.2.3.4 DMA Control Signals 14
3.2.3.5 Miscellaneous Signals 14
3.2.4 Read/Write Cycles 14
3.2.5 Interrupt Cycles 14
3.2.6 Priority Transfer Cycle 14
3.3 Peripheral Bus 14
3.3.1 Control Characters 14
3.3.2 Addressing 14
3.4 Microbus 14
3.4.1 Microbus Signals 15
3.4.2 Microbus Timing 15
3.4.3 Microbus Input/Output Cycles 15
3.4.4 Microbus Interrupt Cycle 15

PART 4 MEMORY

4.1 General 17
4.2 Memory Cycles 17
4.2.1 Word Read 18
4.2.2 Byte Read 18
4.2.3 Word Write 18
4.2.4 Byte Write 18
4.3 Refresh Cycles 18
4.4 Cycle Arbitration 18
4.5 Error Detection 18

PART 5 PROCESSORS

5.1 General 19
5.2 Registers 19
5.2.1 User Register 19
5.2.2 Condition Code Flags 20
5.2.3 System Status 20
5.2.4 System Control Register 21
5.2.5 Base Register 21
5.3 Sector Tables 21
5.4 Address Generation 21
5.5 Stack 21
5.6 Input/Output 22
5.7 Interrupts 22
5.7.1 Millisecond Interrupt 22
5.7.2 Restart Interrupt 23
5.7.3 Error Interrupt 23
5.7.4 Vector Interrupt 23
5.8 RIM 23
5.9 Direct Memory Access 24
5.10 System/Auxiliary ROM 24
5.11 Instruction Set 24
5.11.1 Presentation Format 24
5.11.2 Category 1 — Load Group 25
5.11.3 Category 2 — Stack Control 26
5.11.4 Category 3 — Byte Arithmetic, A Register 27
5.11.5 Category 4 — Word Arithmetic 29
5.11.6 Category 5 — Jumps, Calls, Returns 32
5.11.7 Category 6 — I/O Group 33
5.11.8 Category 7 — System Instructions 35
5.11.9 Category 8 — String Operations 36

PART 6 DISK INTERFACE

6.1 General 41
6.2 Peripheral Input/Output Module 41
6.2.1 Transmitter Logic 42
6.2.2 Receiver Logic 42
6.3 Microbus Interface Module 42
6.3.1 Common Bus Interface 43
6.3.2 Microbus Interface 43
6.3.3 Data Transfer 43
6.3.4 Polling Function 43

PART 7 MULTIPORT COMMUNICATIONS ADAPTER (MPCA)

7.1 General 45
7.2 Microprocessor 45
7.3 Memory 46
7.4 USART 46
7.5 Baud Rate Generators 46
7.6 CPU Interface 46
7.7 Interrupt Structure 46
7.8 Firmware 46
7.9 Diagnostics 46

PART 8 MULTIFUNCTION COMMUNICATIONS ADAPTER (MFCA)
8.1 General 47
8.2 Microprocessor 48
8.3 Serial Interface 48
8.4 Counter/Timer Circuit 48
8.5 Memory 48
8.6 Interrupt Structure 48
8.7 Firmware 48
8.8 Diagnostics 48

PART 9 SYSTEM FIRMWARE
9.1 Introduction 49
9.2 Initialization 49
9.3 Diagnostics 49
9.4 System RAM Vectors 49
9.5 IPL Block Loader 50
9.6 Keyboard/Display Routines 50
9.6.1 $86KEYIN 50
9.6.2 $86KDSII 50
9.6.3 $86CHRLD 50
9.6.4 $86DSPII 50
9.6.5 $86DSPLY 51
9.6.6 $86CRSLD 51
9.6.7 $86CLOC 51
9.6.8 $86RSTRT 52
9.6.9 $86DOSKY 52
9.7 Debug 52
9.7.1 Entry to Debug 52
9.7.2 Saving the Machine State 52
9.7.3 Display Format 52
9.7.4 Command Syntax 52
9.7.5 Input Command List 53

APPENDIX A. ANCILLARY EQUIPMENT
A.1 General 55
A.2 Power Supply 55
A.2.1 Protection Circuits 56
A.2.2 Power Fail Alarm 56
A.3 Motherboard 56
A.3.1 Signal Configuration 56
A.3.2 Open Collector Signals 56
A.3.3 Ground Plane 56
A.3.4 Power and I/O Interconnect 56
A.3.5 DMA Priority Daisy Chain 56

APPENDIX B. INSTRUCTION TIMINGS 57

APPENDIX C. 8600 COMMON BUS I/O ADDRESSES 61

LIST OF FIGURES AND TABLES

Figure 1-1: 8601 Processor 1
Figure 1-2: 8602 Processor 2
Figure 2-1: KDS Module 5
Figure 2-2: General Purpose Keyboard 6
Table 2-1: General Purpose Keyboard Coding 7
Figure 2-3: Character Row 9
Figure 2-4: Status Register 12
Figure 2-5: Command Register 12
Figure 2-6: Mode Register 1 12
Figure 2-7: Mode Register 2 12
Figure 4-1: 128K Memory 17
Figure 5-1: Central Processor/Arithmetic Logic Unit and Central Processor/Control Boards 19
Figure 5-2: Central Processor/RIM Board 20
Figure 5-3: Address Generation Diagram 22
Figure 5-4: Stack Information After Interrupt 22
Figure 5-5: Vector Interrupt Table Format 23
Figure 5-6: Interrupt Mask Byte Format 23
Figure 6-1: Peripheral Input/Output Module 41
Figure 6-2: Microbus Interface Board 42
Figure 7-1: Multiport Communications Adapter Board 45
Figure 8-1: Multifunction Communications Adapter Board 47
Figure 9-1: System RAM Vectors 49
Figure A-1: Power Supply 55

PART 1
GENERAL FEATURES

1.1 Introduction

The Datapoint® 8600 is a versatile, high-performance processor featuring a large-screen amber display and an ergonomic housing. The 8600 supports both of Datapoint's operating systems (Disk Operating System and Resource Management System™) in a variety of configurations. Basic memory size is 128K, expandable to 256K. The 8600 is configurable for standalone, ARC™ (Attached Resource Computer®), or DATASHARE® environments.

All 8600 logical subassemblies reside on printed circuit boards that are contained in an internal ten-slot card cage. The basic 8600 processor includes a three-card central processor, 128K of memory, a Keyboard/Display Subsystem (KDS) module, with an internal Resource Interface Module (RIM). Additional boards can be added for memory expansion, disk interface, terminal support, and data communications. A serial port is included. The 8600 also includes an internal power supply that provides DC power for the keyboard, CRT, and logic boards. These components are linked internally by a common bus.

The 8601 processor is designed to perform as an ARC applications processor in environments requiring full processor power in a single workstation. The only configurable option in an 8601 is memory size (maximum of 256K).

The 8602 processor supports all feature options including additional memory, Multiple Port Communications Adaptor (MPCA), Multiple Function Communications Adaptor (MFCA), and disk interface. The 8602 is also available in system configurations with either the Datapoint 9301 or 9310 disk drives for standalone or RMS ARC operation. An 8601 can be converted to an 8602 with a field upgrade kit. Block diagrams of the 8601 and 8602 processors are given in Figures 1-1 and 1-2. The following sections in this portion introduce the basic elements of the 8600 processor family.

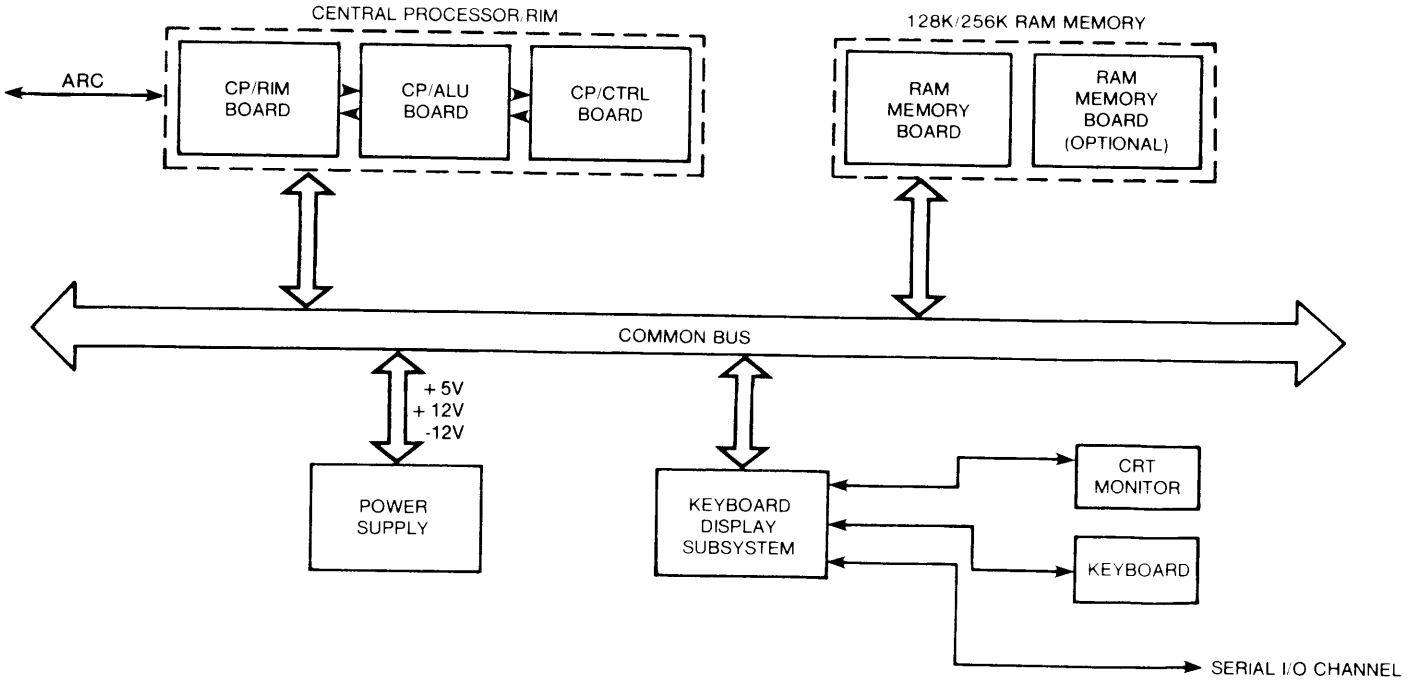
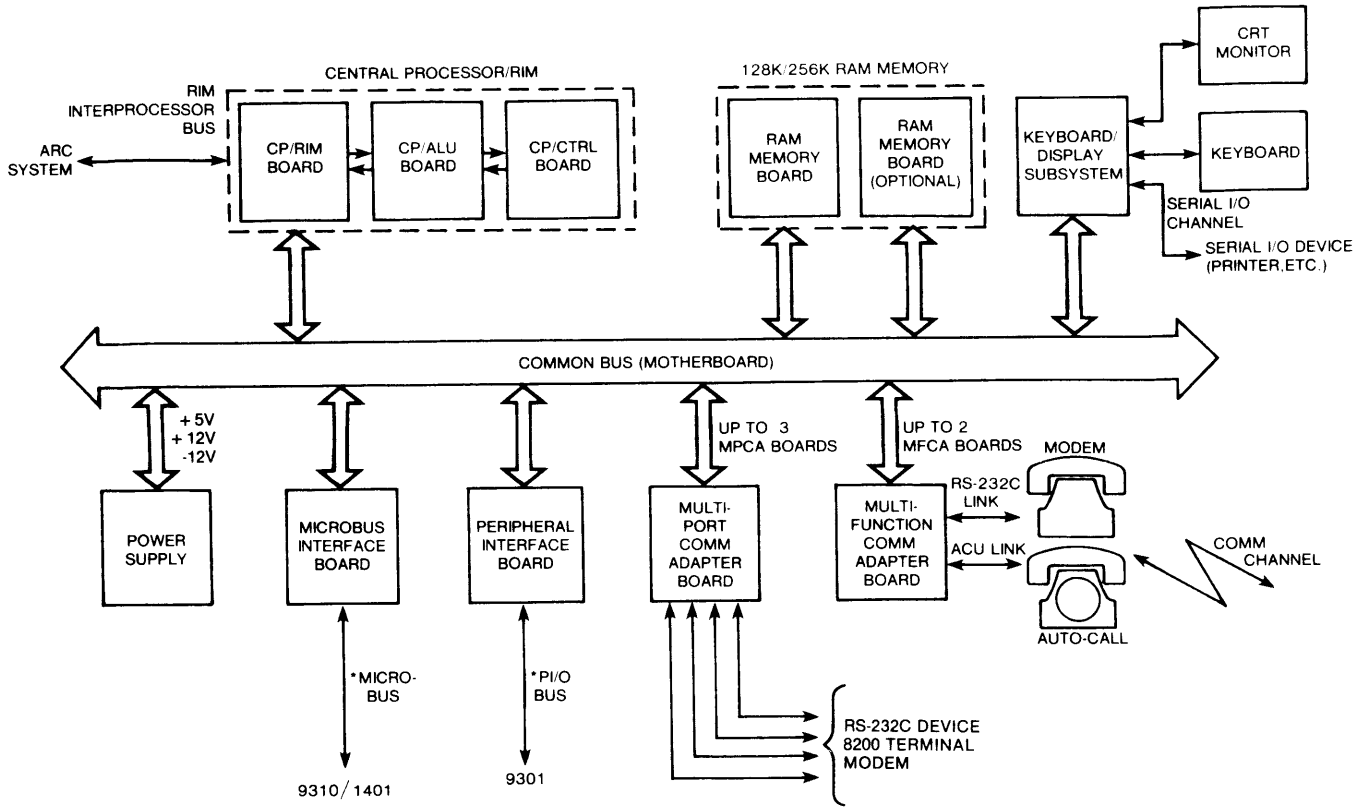


Figure 1-1: 8601 Processor



*A processor may have either a Microbus interface or a Peripheral interface

Figure 1-2: 8602 Processor

1.2 KDS Module

The KDS module provides interface between the keyboard and the processor. The KDS also controls the CRT display and provides a serial I/O port for interface to a Datapoint serial printer or terminal.

The 8600 is equipped with the Datapoint General Purpose Keyboard, which contains a 55-key typewriter pad, an 11-key numeric pad, and a 10-key function pad. The keyboard provides a multi-key roll-over characteristic for ease of typing. The keyboard is detached and can be placed up to one meter from the processor.

The display is a magnetically deflected raster scan with an amber screen CRT. It provides a display of 1920 characters, organized as 24 rows of 80 characters each. Character attributes such as inverse video, underlining, two-level video and blink on a character by character basis are available under software control. The display uses a standard 128-character set or a user-definable font. The viewing area is 5.4 X 8.9 inches (13.7 X 22.6 cm), and standard characters are produced using an 8 X 12 dot matrix in a 9 X 12 field. The display/refresh rate is 60/50 frames/second (line synchronized). Display brightness can be set to one of 16 levels from the keyboard. Beep and click are available under software control.

1.3 Bus Architecture

The 8600 uses an internal bus to provide the interface between the processor, memory, and peripherals. This bus is implemented on the CP/RIM module through the motherboard and is referred to as the common bus.

To communicate with disk storage systems, the 8600 uses one of two external buses: either the peripheral bus or the microbus. An external bus interfaces directly to one of two types of internal logic boards (either a PIO module or a MIFM module, as explained in Section 1.6), which in turn interfaces to the processor and main memory.

The peripheral bus is a high performance 12-bit asynchronous NRZ format serial bus that provides character or block transfers, and a single parity bit for error detection. The peripheral bus is compatible with the Datapoint 9301 series compact disk drives.

The microbus is a parallel asynchronous bus that is used to connect the processor to the Datapoint 9310 disk or 1401 diskette.

1.4 Memory

Basic 8600 memory is 128K expandable to 256K. Each memory module contains 128K bytes of data plus parity and interfaces to the common bus. The memory can be software configured for word or byte accesses.

1.5 Processor

The central processor is implemented on three logic boards:

CP/CTRL (Control) — contains the instruction decode, control store, and sequencer.

CP/ALU (Arithmetic Logic Unit) — encompasses most of the processor registers and handles data flow for the machine.

CP/RIM (Resource Interface Module) — contains the system/auxiliary ROM, sector tables, and an integral RIM for interface to a Datapoint ARC system.

The processor implements the Datapoint 6600 user mode instruction set. The processor is driven by vectored interrupts, which means that it responds directly to interrupting devices. Access protect and/or write protect can be performed on 4K blocks of memory. Four sector tables are organized into system and user areas. The processor uses a 32-word stack located in main memory and can use a stack exchange for multiple stacks. System/auxiliary ROM contains system functions such as power-up, keyboard/display drivers, interrupt, debug, and diagnostic routines.

1.6 Disk Interface

Interface to a disk system is accomplished through an internal printed circuit board. The processor supports two types of disk interface boards: the Peripheral Input/Output module (PIO) and the Microbus Interface module (MIFM). The PIO module provides interface to the Datapoint 9301 series compact disk drives through the peripheral bus. The MIFM module provides interface to the Datapoint 9310 disk drive and the 1401 diskette drive through the microbus. The 8600 processor is available with either type of interface.

1.7 Multiport Communications Adapter (MPCA)

The MPCA logic board houses four serial asynchronous RS-232C compatible communications ports. Each port has full duplex transmit and receive capability. Data transfer rates are software programmable from 50 to 19200 baud. Character lengths and the number of stop bits are individually programmable. The MPCA provides parity bit generation and detection. The MPCA also includes a Z80-A microprocessor, local memory, four USARTs (Universal Synchronous Asynchronous Receiver/Transmitter), and baud rate generators. An 8600 can be configured with a maximum of three MPCAs.

1.8 Multifunction Communications Adapter (MFCA)

The MFCA logic board provides a means of synchronous or asynchronous by-directonal information transfer between the processor and an RS-232C compatibe communications channel with reverse channel. The MFCA may be connected to an external modem and/or an RS-366 compatible Automatic Calling Unit (ACU) and can communicate over switched or leased lines using the following protocols: BISYNC, SDLC, HDLC, ADCCP, or GENSYNC. Baud rates are programmable from 110 to 56K. The MFCA includes a 280-A microprocessor, a serial input/output channel, a counter/timer circuit, and 16K of local memory. And 8600 can be configured with a maximum of two MFCAs.

1.9 Ancillary Equipment

In addition to the components discussed above, the 8600 includes an internal power supply and motherboard. The power supply is housed in the internal card cage and provides DC power for the keyboard, CRT, and logic boards. The motherboard is a nine-card backplane that implements the 8600 common bus.

1.10 General Specifications

Power Requirements:
120 or 240 VAC (+/- 10%)
50 or 60 Hz (+/- 1 Hz)
230 watts (785 Btu/hr)

Equipment Dimensions:
Keyboard:
Width 20.0 inches (50.8 cm)
Height 2.75 inches (6.9 cm)
Depth 8.9 inches (22.6 cm)

Terminal:
Width 20.0 inches (50.8 cm)
Height 13.2 inches (33.5 cm) without base
Depth 14.5 inches (36.8 cm)

Total Weight 60 pounds (22.4 kg)

Operating Environment:
50 to 100 degrees F
10 to 38 degrees C
20 to 90 percent relative humidity, non-condensing

1.11 Peripherals

The 8600 will accommodate a wide variety of peripherals including the Datapoint 9301 series compact disk drives, 9310 disk drive, 1401 diskette drive, printers, and communications equipment. Refer to the Datapoint Equipment Catalog (Model Code 60001) for a complete description of peripherals.

1.12 Model Codes

8601	Applications Processor, 128K-256K memory
8602	Standalone/Data Resource Processor, 128K-256K memory

This page intentionally left blank.

PART 2
KEYBOARD AND DISPLAY SUBSYSTEM (KDS)

2.1 General

The Keyboard and Display Subsystem (KDS) is a logic board that provides interface between the keyboard and the central processor. The KDS also controls the CRT display and provides a serial I/O port for interface to a Datapoint serial printer or terminal. The KDS includes local RAM for screen memory, row pointers, the cursor pointer, and character font storage. A block diagram of the KDS module is shown in Figure 2-1.

The KDS is accessed through I/O address space. Common bus reads or writes to the KDS are allowed at any time during the video cycle.

2.2 Keyboard

The 8600 processor is equipped with the Datapoint General Purpose Keyboard, which contains a 55-key typewriter pad, an 11-key numeric pad, and a 10-key function pad. The keyboard is used for data entry and control of the 8600. The keyboard is detached and may be placed up to one meter from the processor. Figure 2-2 illustrates the general purpose keyboard. Keyboard coding is given in Table 2-1.

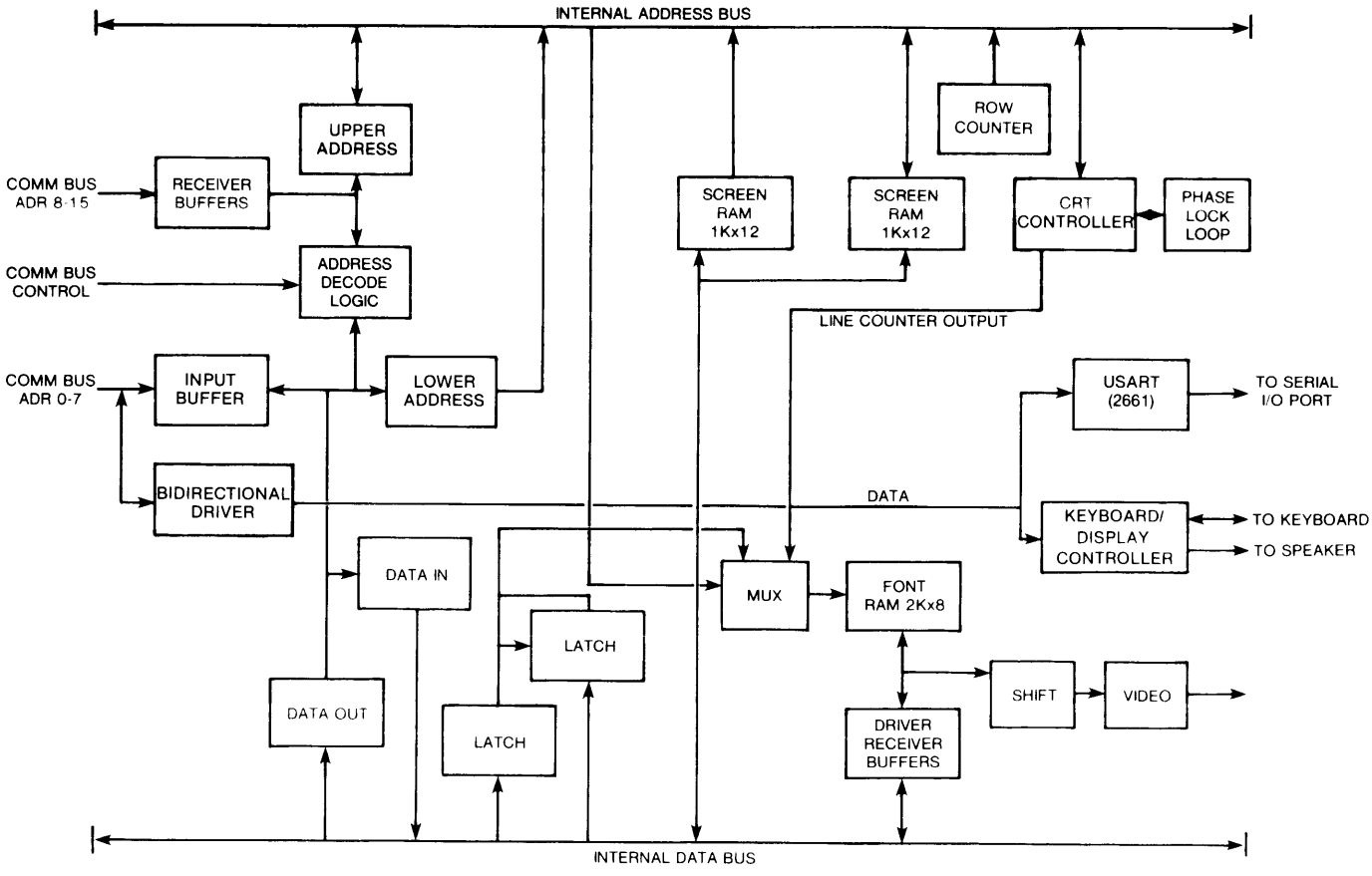


Figure 2-1: KDS Module

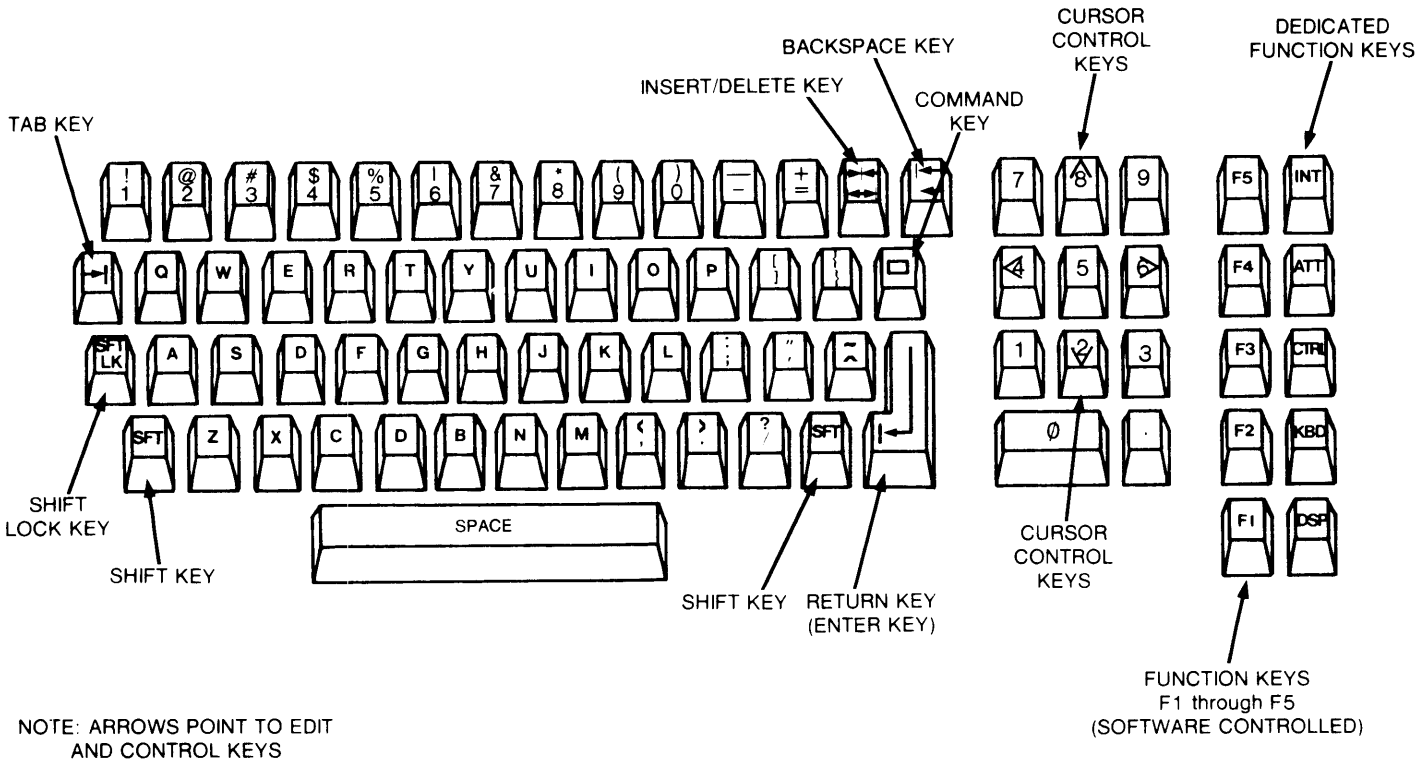


Figure 2-2: General Purpose Keyboard

Table 2-1 General Purpose Keyboard Coding

This table represents the octal equivalents of all characters on the keyboard. The values to the left of the character are the translated codes; those to the right are the untranslated codes.

T		U	T		U	T		U	T		U
101	A	101	141	a	141	060	0	060	072	:	053
102	B	102	142	b	142	061	1	061	073	;	073
103	C	103	143	c	143	062	2	062	074	<	074
104	D	104	144	d	144	063	3	063	075	=	137
105	E	105	145	e	145	064	4	064	076	>	076
106	F	106	146	f	146	065	5	065	077	?	077
107	G	107	147	g	147	066	6	066	100	@	042
110	H	110	150	h	150	067	7	067	133	[	100
111	I	111	151	i	151	070	8	070	135	]	140
112	J	112	152	j	152	071	9	071	136	^	135
113	K	113	153	k	153	040	Space	040	137	_	075
114	L	114	154	l	154	041	!	041	173	{	174
115	M	115	155	m	155	042	"	052	174	|	046
116	N	116	156	n	156	043	#	043	175	}	134
117	O	117	157	o	157	044	$	044	176	~	175
120	P	120	160	p	160	045	%	045	033	Tab	033*
121	Q	121	161	q	161	046	&	047	036	Tab	233
122	R	122	162	r	162	047	'	072	015	Return	015*
123	S	123	163	s	163	050	(	051	035	Return	215
124	T	124	164	t	164	051	)	000	010	Backspace	100*
125	U	125	165	u	165	052	*	050	020	Backspace	210
126	V	126	166	v	166	053	+	177	037	Insert	133
127	W	127	167	w	167	054	,	054	177	Delete	173
130	X	130	170	x	170	055	–	055	134	Command	136*
131	Y	131	171	y	171	056	.	056	136	Command	176
132	Z	132	172	z	172	057	/	057			

Numeric Pad

Symbol	Unshifted		Shifted	
	T	U	T	U
.	056	256	056	256
0	060	260	060	360
1	061	261	061	361
2	002	262	062	362
3	063	263	063	363
4	004	264	064	364
5	065	265	065	365
6	006	266	066	366
7	067	267	067	367
8	005	270	070	370
9	071	271	071	371

Function Keys

Symbol	Unshifted		Shifted	
	D**	R**	D**	R**
F5	300	320	340	320
F4	302	322	342	322
F3	304	324	344	324
F2	306	326	346	326
F1	310	330	350	330
RESTART	301	321	341	321
ATT	303	323	343	323
INT	305	325	345	325
KBD	307	327	347	327
DSP	311	331	351	331

* Unshifted representation

** D is the code for depression; R is the code for release.

2.2.1 Keyboard Control

The keyboard contains a number of control and function keys. These keys are defined below.

RETURN (ENTER) — The Return key is under software control. It is normally programmed to do a carriage return, accept data, control entry, and/or start execution.

SHIFT — The Shift key causes the keyboard to produce uppercase character code. When this key is not pressed, the keyboard will produce lowercase character code.

SHIFT LOCK — The Shift Lock key locks the Shift key in the uppercase position. When Shift Lock is activated, the LED on this key cap will be illuminated.

TAB — The Tab key is used in word processing to advance the cursor to pre-defined or user-defined tabular fields within text. When not used for word processing, this key acts as a Cancel key to move the cursor to the start of a line.

INSERT/DELETE — The Insert/Delete key is used in word processing to open text for an insert when unshifted or to delete text when shifted. When not used for word processing, this key advances the cursor one position to the right, leaving a space.

COMMAND — The Command key is used in word processing to enter selected commands. When not used for word processing, the key produces an accent (′) when unshifted and a backslash (\ ) when shifted.

BACKSPACE — This key backspaces the cursor one position, erasing the preceding character.

CURSOR CONTROL — These keys are used in word processing to move the cursor up, down, left, and right when unshifted. When shifted, these keys produce the indicated numerals. When not used for word processing, these keys produce only numerals.

INT (Interrupt) — The INT key, when pressed with the CTRL key, causes the processor to halt and execute the restart routine contained in system ROM. This key is also used in conjunction with the CTRL and DSP keys to cause entry to debug.

CTRL (Control) — The CTRL key works in conjunction with other function keys to initiate special action.

ATT (Attention) and KBD (Keyboard) — These keys are used in conjunction with the CTRL key to increase or decrease display brightness.

DSP (Display) — The DSP key causes entry into debug when used with the CTRL and INT keys.

F1, F2, F3, F4, F5 — Each of the function keys is under software control.

The INT, CTRL, ATT, KBD and DSP keys can be detected under software control.

2.2.2 Special Key Sequence Controls

The 8600 processor has six functions that can be initiated through the use of special keyboard keys, held down together to perform unique functions. These functions are listed below, followed by a group of keys that must be held down together to control the functions.

Restart	CTRL, INT
Restart (RIM boot)	KBD, DSP, CTRL, INT (release CTRL and INT)
Brightness Increase	CTRL, ATT (release ATT)
Brightness Decrease	CTRL, KBD (release KBD)
Keyboard Lockout	Depress the CTRL key while entering the TAB key, zero to four character lock code keys, and then the ENTER key. Release the CTRL key.
Keyboard Unlock	same as Keyboard Lockout.
Restart Lockout	Same as Keyboard Lockout, except shifted ENTER key is pressed. Check for presence of lockout key depressing CTRL and DSP; release DSP. Processor will beep if lockout is set.*
Restart Unlock	Same as Restart Lockout.*
Debug	DSP, CTRL, INT (release CTRL or INT)

* These features available after July, 1982.

The brightness level of the display can be set from the keyboard to one of 16 levels. To increase brightness, press the CTRL key followed by the ATT key. Every release of ATT, while CTRL is down, will increase the brightness by one level. After a brightness level of zero has been reached, this command will be ignored.

To decrease brightness, press the CTRL key followed by the KDB key. Every release of KBD, while CTRL is down, will decrease the brightness by one level. After a brightness level of 15 has been reached, this command will be ignored.

2.3 Display

The display uses a magnetic deflection technique with an amber screen CRT. It provides a display of 1920 characters, organized as 24 rows of 80 characters each. The character font is generated through the use of RAM memory. A standard font is loaded into RAM from ROM after power-up. Other character fonts may be loaded as desired under software control. Up to 128 different individual 8 X 12 dot matrix characters may be produced. (Standard dot matrix is 7 X 9 for upper case letters. Lower case are 7 X 11 for descenders.) The display/refresh rate is 60/50 frames/second (line synchronized).

The screen refresh memory is organized as 2K bytes in I/O memory space located at address 0140000 octal. It contains the cursor pointer, row pointer, and video information required for screen refresh.

The lower 64 bytes of the screen memory are dedicated for the cursor and row pointers. The cursor pointer is stored in locations 0000 and 0001. Any of the 1920 character locations can be loaded into these locations to display the cursor at any of the character locations.

The address of the first character of each of the 24 rows is stored in locations 0002-0061, with the first character address pointer for the first row at 0002 and 0003, and the first character address pointer for the last row at 0060 and 0061. The display can be manipulated by storing different character addresses in the row pointers. The cursor and row pointers must always be loaded as an even byte followed by an odd byte. All addresses are relative to a base address of KDS (0140000).

2.3.1 Display Character Format

Every display character is stored in the screen memory as a byte using its ASCII value. The lower seven bits represent the ASCII value of the display character; the eighth bit is a video attribute. The seven ASCII bits are used to obtain the start address of the character font in the display font RAM. The video attribute bit is used for inverse video on a character by character basis.

2.3.2 Video Attributes

The character by character attributes are inverse video, underline, two-level video, and blink. Inverse video is the most significant bit of the screen memory data location. When this bit is set (1), the character is displayed in inverse video. The character is displayed in regular video when the bit is reset (0).

Underline, two-level video, and blink are discussed in Section 2.5.3.

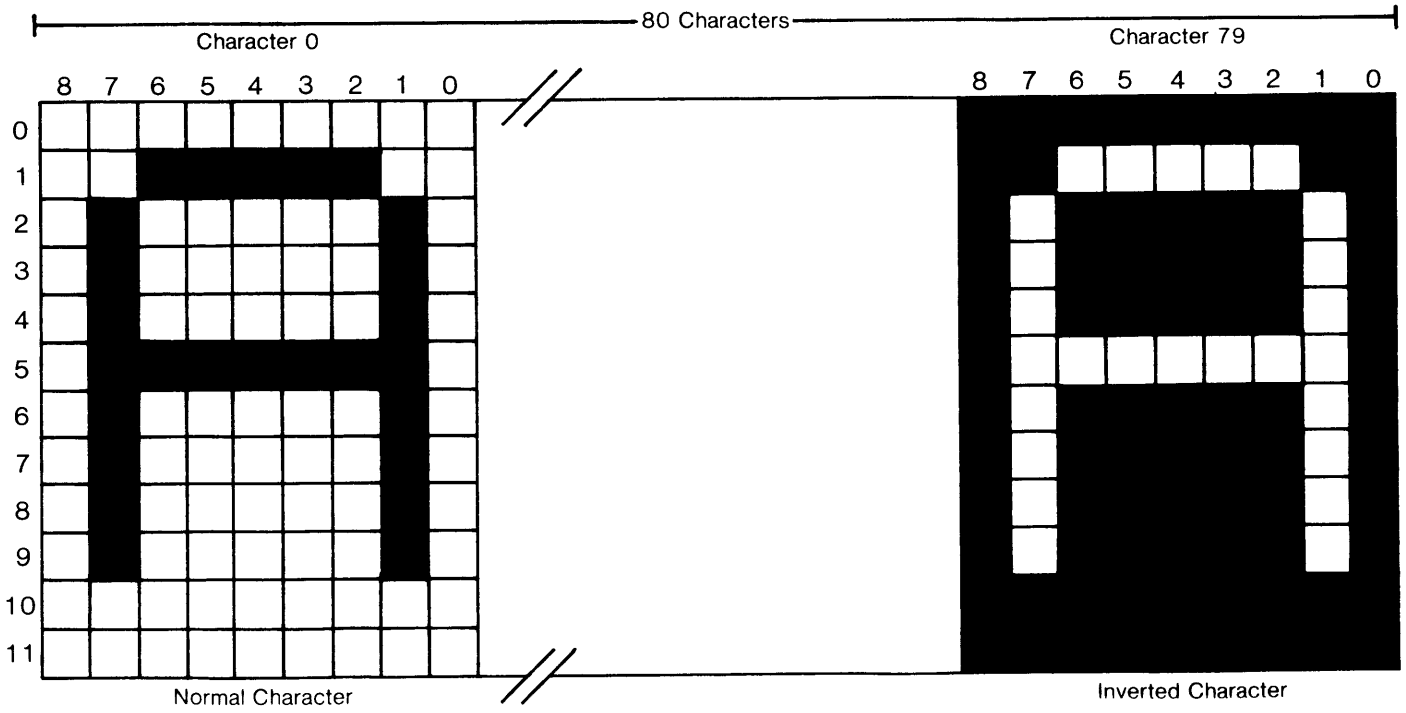
2.3.3 Character Font Load

Displayed characters are generated from a loadable RAM memory, providing for intenational character sets and other user-selected character set variations. The RAM is down-line loaded through optional ROM software by the main program. The character generation RAM is a 2048 by 8 memory organized as 128 characters of twelve bytes each.

The display font is addressed by using the ASCII value of the character. The ASCII character value is shifted left four bits, and the result is added to the base address of 0140000 to obtain the start address of the character font. The RAM address for the display font may be separated into fields as follows:

Base Address					ASCII (7 bit)							Video Line			
A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

The top line of the display character is defined by the eight bits of data at the start address. The next lines of the character define the next nine memory bytes. The last line of the character is contained in the eleventh memory address. The first and twelfth data locations are filled with zeros so that a border can be formed for the inverse video. Twelve lines, total, can be loaded.



Scan Line No.	
0	Always Zero (Firmware)
1-10	Video Lines
11	Always Zero (Firmware)

Figure 2-3: Character Row

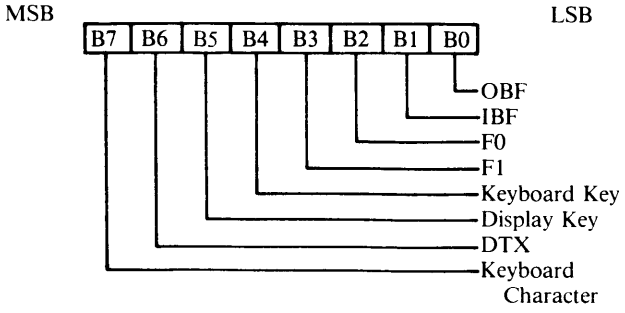
2.4 Serial I/O Port

The KDS is equipped with one standard RS-232C serial interface for connection to a Datapoint printer or workstation. The baud rate is software controlled from 50 to 19200 baud. All control signal detection and character formatting is handled by an on-board USART (Universal Synchronous Asynchronous Receiver/Transmitter). When running a terminal off the KDS I/O port, the transmit and receive baud rates must be the same.

2.5 KDS Programming Considerations

2.5.1 Keyboard and Screen

Status and commands for the keyboard and display are accessed by reading from base page I/O address 031 for status or writing it for commands. The status word format follows. Once the appropriate command has been issued and executed by the KDS, any requested status will be placed in the Data Bus Buffer (DBB) at base page I/O address 030.



The following descriptions explain the bits.

Bit 0 — Output Buffer Full (OBF). This is set by hardware whenever the KDS outputs a byte to the master in the DBB. It is cleared when the master reads a byte.

Bit 1 — Input Buffer Full (IBF). This is set by hardware when the master writes a byte to the KDS in the DBB. It is cleared when the KDS reads it.

Bit 2 — The F0 busy flag is set by the KDS firmware to indicate that it is busy servicing a command. It is cleared when the KDS is ready to service a new command.

Bit 3 — The F1 flag is set by the hardware to the state of address line 0 and is used by firmware to distinguish between commands and data.

Bit 4 — If this bit is set, it indicates that the keyboard key is depressed.

Bit 5 — If this bit is set, it indicates that the display key is depressed.

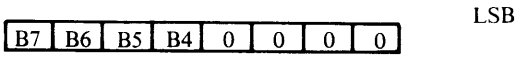
Bit 6 — Data Transmit Ready (DTX). If this bit is set, the KDS is ready to transmit a character to the keyboard.

Bit 7 — If this bit is set, the KDS has a keyboard character available for the master upon request.

2.5.1.1 Commands

There are three commands associated with the keyboard and display control: read request commands, screen commands, and restart clear commands.

READ REQUEST COMMANDS



The bits are defined as follows:

Bit 0-Bit 3 — 0000 informs the KDS that it is a read request command.

Bit 4-Bit 7 — 0000 commands the KDS to reset.

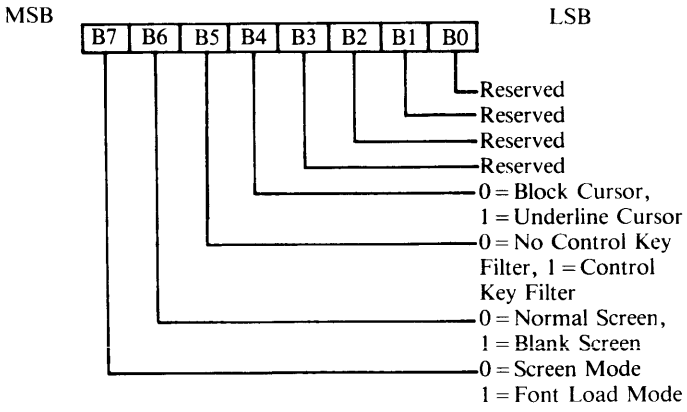
0001 commands the KDS to present the keydown status in the DBB.

0010 commands the KDS to present the screen/status byte in the DBB.

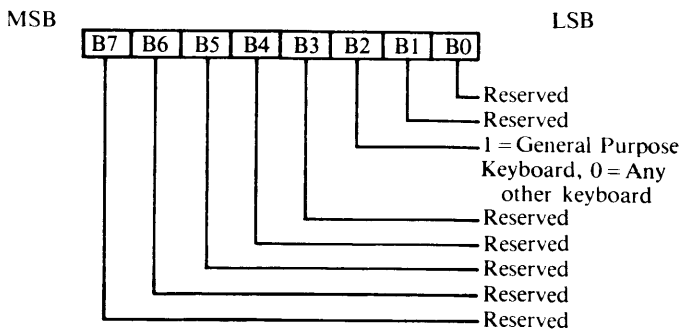
0100 commands the KDS to present the keyboard character into the DBB.

1000 commands the KDS to present the keyboard type in the DBB.

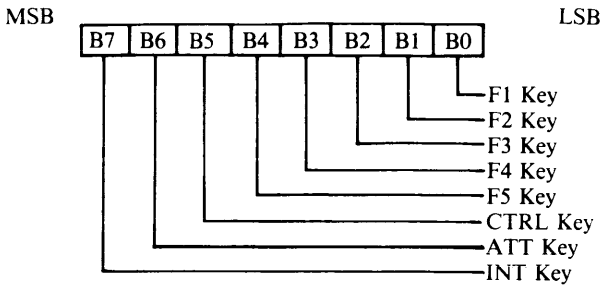
The format for the individual status bytes is shown below:



The Keyboard type byte is shown below:

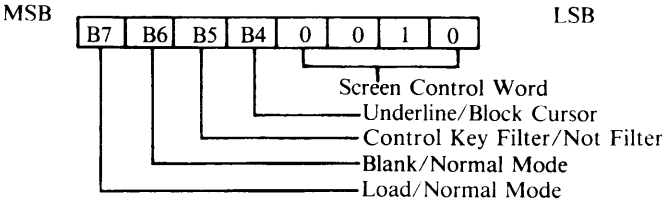


The keydown status byte is shown below. Each status bit will be a one when the corresponding key is depressed and will remain a one until the key is released.



SCREEN COMMANDS

The reception of this control word informs the KDS to perform certain screen manipulations



Bits 0-3 — 0010 informs the KDS that it is a screen control word.

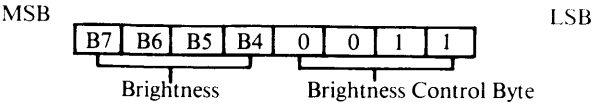
Bit 4 — When this bit is set to 1, the KDS will set the underline cursor mode. When it is a 0, the KDS will set the block cursor mode. On power up, the cursor is set to the block mode.

Bit 5 — When this bit is set to 1, the KDS will not send control key codes to the master. When this bit is reset to 0, all key codes are sent to the master, except for the special key sequences and the CTRL key code (see Section 2.2.2).

Bit 6 — When this bit is set to 1, the KDS blanks the screen. When it is set to 0, the screen operates in normal mode.

Bit 7 — When this bit is set to 1, the KDS overlays the 2K character font RAM on the screen buffer. When this is cleared, the screen enters normal mode. On power up, normal mode is set.

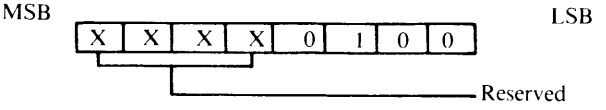
Brightness Control



Bits 0-3 — 0011 informs the KDS that it is a brightness control byte.

Bits 4-7 — These bits control the brightness of the screen. The brightness can be set to any one of the sixteen levels, 0 being the highest brightness and 15 being the lowest.

Restart Clear Command

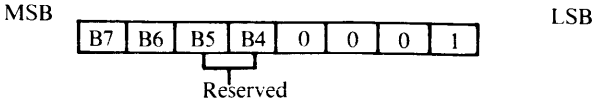


Bits 0-3 — 0100 informs the KDS that it is a restart acknowledge byte. Upon receipt of this command, the KDS clears the restart pulse.

Bits 4-7 — Reserved

2.5.2 Speaker Commands

Speaker control is initiated by writing to base page I/O address 031. The byte format is as follows:



Upon receipt of this command, the KDS initiates the beep or click operation. The beep is a 1200 Hz tone generated for a duration of 250 milliseconds. The click is an 800 microsecond pulse with a minimum of 800 microseconds guaranteed between clicks.

Bits 0-3 — 0001 informs the KDS that it is a speaker control word.

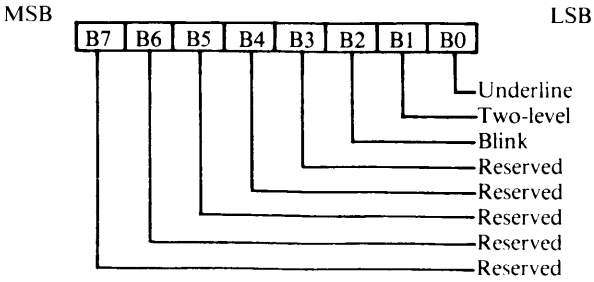
Bits 4-5 — Reserved.

Bit 6 — When this bit is set, it commands the KDS to initiate a beep operation.

Bit 7 — When set, this bit commands the KDS to initiate a click operation.

2.5.3 Video Attributes

Underline, two-level video, and blink are programmed by writing to the attribute RAM. Screen character locations run from 0140100 to 0143777, and attribute RAM extends from 0144000 to 0147777. Attribute data for a particular screen byte will reside at a 04000 offset from the screen RAM location. Therefore, screen data at location 0140100 will have its attributes at extended I/O address 0144100 and so on. The bit positions for attributes are shown below:



As mentioned in Sections 2.3.1 and 2.3.2, the inverse video attribute bit is controlled by manipulation of the most significant bit of the screen RAM byte for a particular character.

2.5.4 Serial I/O Port

The KDS serial I/O port is accessed through base page I/O address 034 to 037.

The RS-232C signals supported are Transmit Data, Receive Data, Secondary Receive Line Signal Detect (printer busy), Data Set Ready, and Data Carrier Detect. Control of all signals is handled by a USART.

Connections to the USART and the function of each signal are as follows. Transmit and receive data are self-explanatory. Emptying of the transmit holding register or fill-

3.2.3.4 DMA Control Signals

These signals provide for the transferring of bus control between devices.

CBDMAREQ/	DMA Request - Negative True
CBBUSY/	Busy - Negative True
CBPRII	DMA Priority Input - Positive True
CBPRIO	DMA Priority Out - Positive True

3.2.3.5 Miscellaneous Signals

CBRESET/	System Reset - Negative True
CBALARM/	System Power Alarm - Negative True
CBRESTART/	Restart - Negative True
CBLFCLK	Line Frequency Clock

3.2.4 Read/Write Cycles

Four types of read/write cycles are defined for the common bus:

- Memory read/write word
- Memory read/write byte
- I/O read/write word
- I/O read/write byte

Memory cycle bus timings are identical. For I/O operations, a single wait state is automatically inserted.

3.2.5 Interrupt Cycles

The interrupt sequence provides an entry address (interrupt vector) to the processor. The bus sequence is identical for each of the three cycles that make up the interrupt acknowledge sequence.

3.2.6 Priority Transfer Cycle

A priority cycle is one that is initiated by a request from a DMA device. The requesting device becomes the bus master when the processor relinquishes control of the bus.

3.3 Peripheral Bus

The external peripheral bus is a high-performance, general purpose serial bus. Commands and data are transferred between the processor and multiple attached peripherals over two sets of individually shielded twisted pair wires at distances of up to 100 feet. The wires are driven and received by RS-422 differential devices. The bus also includes additional wires for an interrupt request line, a power-on indication, and logic ground. All wires are enclosed in an overall jacket to complete the cable assembly.

Peripherals interface to the bus in a multi-drop fashion, and physical connection is via an "IN" and "OUT" connector on each peripheral in a daisy-chain fashion. The last peripheral in the chain has a terminator attached to its "OUT" connector. One twisted pair wire set is used for communication from the processor to the peripherals, while the second wire pair is used for transfer from the peripherals to the processor. Data transfer over the cable is half-duplex.

Characters are transferred over the bus in a 12-bit asynchronous format at 13.055 Mbits/second, which is compatible with the transfer rate of the Datapoint 9301 disk. This corresponds to a maximum continuous transfer rate of 1.088 Mbytes/second over the bus. A single parity bit is included in the 12-bit character format for error control. Data is encoded in NRZ format, and character synchronization is achieved by using a precision edge-triggered oscillator tuned to the proper frequency. Data transfers may be on a character or block basis.

The twisted pair wire set that is not used for data transfer during a communication is used as a "handshake line," which allows matching of the transfer rate of the processor to any peripheral on a character-by-character basis.

3.3.1 Control Characters

Control characters are used to address devices, issue commands, read status, and for bus control functions. Control characters are uniquely identified by the control/data bit being a logical 1 in each control byte. This allows any 8-bit binary value to be sent as a control character.

3.3.2 Addressing

All peripherals on the bus will power-up de-addressed and will become de-addressed if issued a bus reset or a de-address control byte. When a peripheral receives an address command, it will compare the address field of the command against its own address. If the addresses match, the peripheral will become addressed. If the addresses do not match, the peripheral will become or remain de-addressed. Once addressed, the peripheral will automatically go to transmit mode and send the interrupt status byte before returning to receive mode.

When a peripheral is not addressed, its corresponding interface module will accept only an address control byte or a bus reset.

Four bits of addressing are provided to access one of several peripherals that can be attached to the peripheral bus. No permanent address assignments exist. Each peripheral, regardless of its type, is simply assigned the next sequential address not currently in use. The only restriction when configuring peripherals is that each device have a unique address before the system is powered up. The processor will determine the peripherals that are present by polling all addresses and reading the interface type status byte during power-up initialization.

3.4 Microbus

The microbus is an external bus that provides interface between the processor's MIFM module and as many as four peripheral devices. The microbus handles block data transfer rates of 500K bytes per second. Maximum block length is 256 bytes.

The microbus is composed of an eight-bit command and address bus, an eight-bit bi-directional data bus, two command strobes, an interrupt acknowledge strobe, an interrupt request line, and a +5 volt power indication.

3.4.1 Microbus Signals

The microbus provides four bits of peripheral device address (A0-A3). Four command lines provide unique commands for each of two transfer strobes (USTB 1 & 2). A third strobe (IACK) is used to acknowledge an interrupt request. The content of the command and address lines is not defined during the IACK strobe. Decoding of the peripheral device address lines is determined by jumpers in each device. No two devices may have the same address on the same microbus.

The eight bi-directional lines are used to transfer data to and from the peripheral device under control of the processor and the MIFM. The drivers are open collector type. The data drivers in the MIFM are enabled only during an MIFM write cycle. The data drivers in the peripheral device are active only when commanded to be by the MIFM.

3.4.2 Microbus Timing

The address, command, and data lines are stable for at least 100 ns before the leading edge of the transfer strobe, and remain so for at least 100 ns after the trailing edge. The transfer strobes are a minimum of 400 ns wide. During a microbus input cycle, the peripheral device will put stable data on the data lines at least 100 ns prior to the end of the transfer strobe and hold it for 100 ns after the end of the transfer strobe.

IACK strobe timing differs from that of the transfer strobes since it is propagated through each peripheral in a daisy-chain fashion. The pulse width of the IACK strobe is jumper selectable to provide a maximum of 1400 ns, which is required when four peripheral devices that use the microbus interrupt line are present on the bus.

3.4.3 Microbus Input/Output Cycles

During an output cycle the MIFM loads the address, command register, and data lines with the appropriate information and enables the bus drivers. A transfer strobe is then generated. The peripheral device, by decoding the address lines, determines if the command and data information is to be latched, and completes the transfer on the trailing edge of the transfer strobe.

During an input cycle, the MIFM loads the command and address register with the appropriate information and enables the bus drivers. A transfer strobe is then generated. The peripheral device, by decoding the address and command information, determines what information to place on the data bus. The peripheral device disables its bus drivers after detecting the trailing edge of the transfer strobe.

The peripheral device completes the specified command and is ready to accept additional commands at the rate of one every two microseconds. Commands requiring longer than this are associated with a busy status bit.

3.4.4 Microbus Interrupt Cycle

An interrupt cycle is initiated on the MIFM whenever a peripheral device pulls the microbus IREQ line low, or when the polling sequence detects an expected condition and generates an interrupt. The processor may respond at any time to the IREQ by initiating the microbus IACK strobe. The IREQ line is independent of all other activity on the microbus.

The peripheral device that initiated the IREQ responds to the IACK strobe by placing its device address and other information (as determined by the peripheral device) on its data lines. The peripheral device releases the data lines on the trailing edge of the IACK strobe.

4.2.1 Word Read

The word read cycle begins when the memory detects that it has been selected for access by decoding the proper address and memory cycle request bits off of the common bus during the time that Address Strobe is present. This information is latched on the trailing edge of Address Strobe thus generating the Row Address Strobe (RAS). The Address Mux Signal (MUX) and Column Address Strobe (CAS) are generated from delayed versions of RAS. The RAS signal input to the memory array is further decoded to select the proper bank of the array. All of these signals are terminated at the same time when the CAS signal has met the minimum pulse width specification. This allows the cycle to run free of any bus timing restraints and only requires Address Strobe to initiate a cycle. Data is enabled onto the bus when Transfer Strobe has been detected during a read cycle and the board has been selected. If during the course of the cycle the RAM INHIBIT signal is detected, the output drivers are not enabled although the cycle runs to completion.

4.2.2 Byte Read

The byte read cycle is identical to the word read cycle except for the case of an odd byte access. In this case the odd byte is physically located in the upper byte of the memory word and, upon access, is driven to the lower byte where the bus master is expecting the data. These bits are also driven on the most significant part of the bus, although no device will receive them. During even byte reads, the corresponding odd byte is driven to the MSB of the bus, although it is assumed no device will receive it.

4.2.3 Word Write

The word write cycle is similar to the word read cycle except that when the Write Strobe and Transfer Strobe are present on the bus and the board has been selected, a write pulse is generated and sent to the memory array. Even parity is generated over each byte of the data to be written and is subsequently written into the array with the data.

4.2.4 Byte Write

The byte write cycle is similar to the word write cycle except that the write pulse is sent only to the selected byte instead of the entire word. A read-modify-write cycle is not required, since parity is generated over the byte.

4.3 Refresh Cycles

The refresh controller is synchronized to the system bus clock that runs at a nominal frequency of 4MHz. A counter is implemented such that a refresh request is generated every 15 microseconds.

The address selection jumper on the board is used to offset the refresh counter so that systems with two 128K modules will not refresh their respective arrays simultaneously.

A refresh is performed by selecting one of 128 rows in the memory and strobing the chip with RAS. A counter on the board is used to select the row to be refreshed. If no cycle is in progress and a refresh request is detected (by the 15-microsecond timeout), a refresh cycle is initiated. At this time the counter outputs are gated to the memory array, and RAS is generated from the bus clock. RAS is one cycle wide. Its completion increments the row counter and resets the refresh request condition.

4.4 Cycle Arbitration

Bus access cycles and refresh cycles cannot have simultaneous memory access. Bus master requests to memory are deferred during a refresh cycle; refresh requests are deferred during a bus access cycle. Cycles will proceed upon demand during those times that arbitration is not required. The refresh logic provides 128 refresh cycles every two milliseconds to ensure the validity of data within the memory.

4.5 Error Detection

The memory carries parity over each eight-bit byte. When a write to memory occurs, a ninth bit (for each byte) is generated via a parity generator and written to the memory. The parity is checked when the data is read from memory. If the parity checking logic does not compute the correct parity, an error condition signal is placed on the common bus.

PART 5
PROCESSOR

5.1 General

The processor in the 8600 is implemented on three printed circuit boards that reside in the internal card cage:

CP/CTRL (Control) — contains the instruction decode, control store, and sequencer.

CP/ALU (Arithmetic Logic Unit) — encompasses most of the processor registers and handles data flow for the machine.

CP/RIM (Resource Interface Module) — contains the system/auxiliary ROM, interrupt controller, sector tables, and a RIM for interface to a Datapoint ARC system.

Block diagrams of the CPU board set are shown in Figures 5-1 and 5-2. Figure 5-1 illustrates the CP/CTRL and CP/ALU boards. Figure 5-2 illustrates the CP/RIM board.

5.2 Registers

5.2.1 User Registers

The 16 user-accessible registers are implemented as two banks of eight, referred to as alpha and beta mode registers. Each group of registers has a corresponding group of condition flags. The registers are referred to as A, B, C, D, E, H, L, and X. The registers are normally assigned the following functions:

A	Accumulator
B,C,D,E	General Purpose
H,L	Memory Pointers

The X register is a working page register and is used to form the upper eight bits of the address for paged address mode.

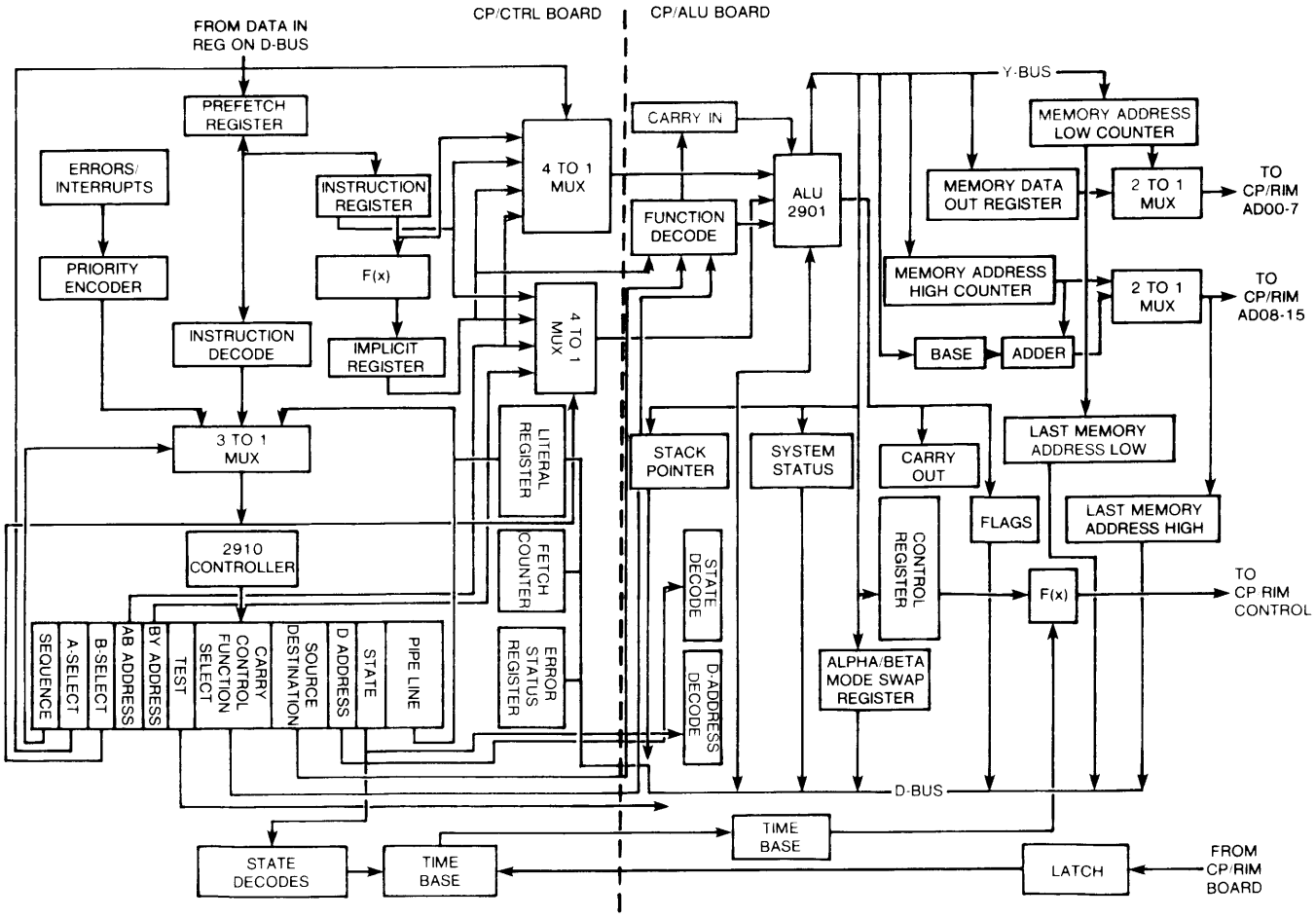


Figure 5-1: Central Processor/Arithmetic Logic Unit and Central Processor/Control Boards

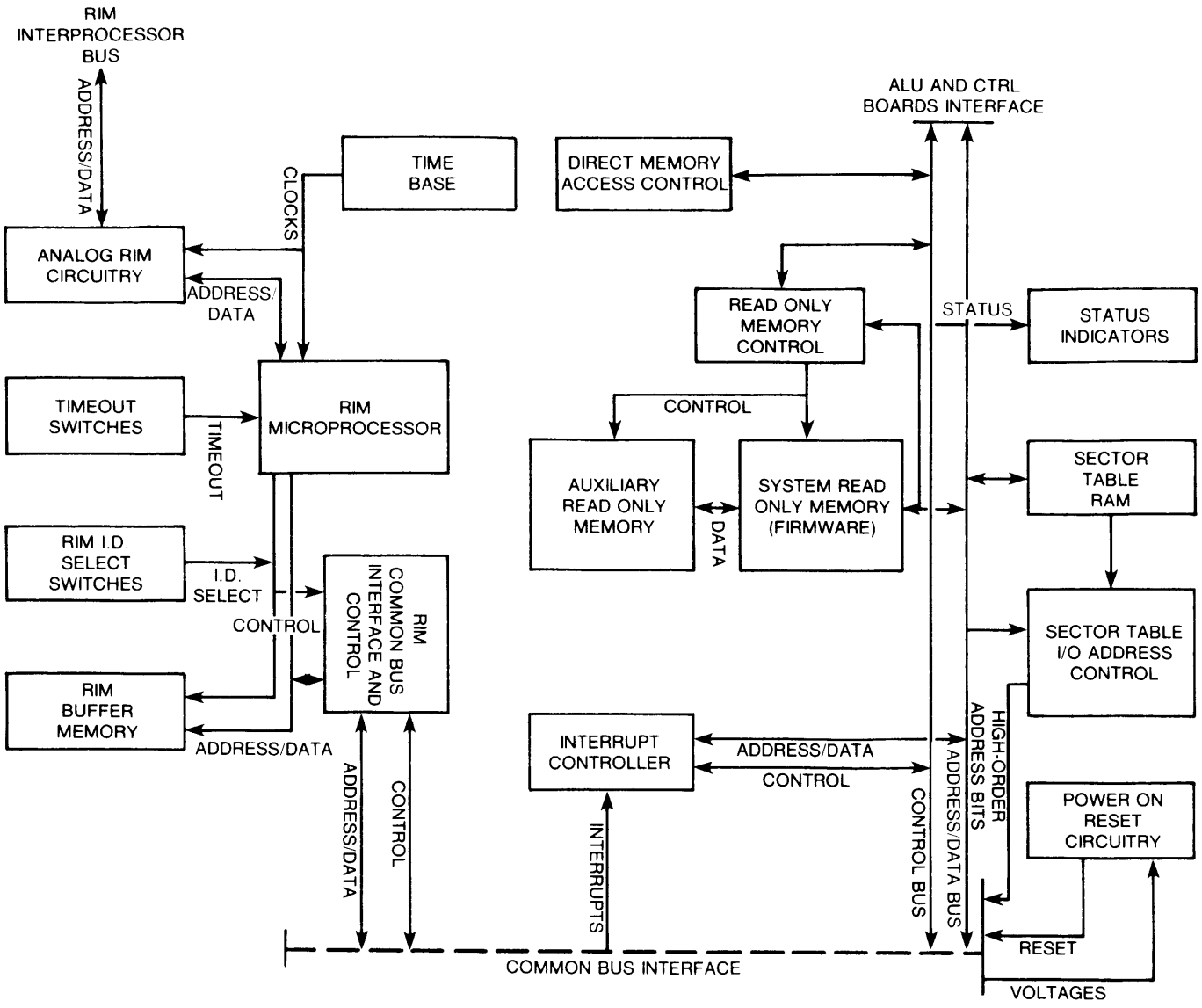


Figure 5-2: Central Processor/RIM Board

5.2.2 Condition Code Flags

There are four condition code flags for each of the two user register banks. The Carry, Sign, Zero, and Parity flags reflect the results of the data following certain instructions:

C Carry - set if there is a carry out of or a borrow into the most significant byte (MSB) of the result of an arithmetic operation. It is cleared on logical operations.

S Sign - reflects the MSB of the result of an arithmetic or logical operation.

Z Zero - set if all bits of an arithmetic or logical operation are zero. Otherwise, it is cleared.

P Parity - set if there is an odd number of ones in the result of an arithmetic or logical operation.

All logic within the processor is 2's complement.

5.2.3 System Status

The system status is an 8-bit, read-only register that reflects the state of the interrupt enable, alpha/beta, and user mode flip-flop, as follows:

Bit 7: System Interrupt Enable. When equal to 1, vector interrupt requests will cause an interrupt trap.

Bit 6: Millisecond Interrupt Enable. When equal to 1, a millisecond timeout will cause the processor to take the millisecond interrupt trap.

Bit 5: User Mode. When equal to 1, execution of a priviledged instruction code will cause an error interrupt and causes the selection of the user sector tables if they are enabled.

Bit 4: Beta Mode. When equal to 1, beta user register set and flags are selected; otherwise, alpha registers and flags are selected.

Bits 3-0: These bits are reserved and are always read back as a zero.

All bits of the system status register are set to zero after reset or restart. Also, system status bits 6 and 7 will always be set to the same value, so that Millisecond and Vector Interrupts are always enabled and disabled together.

5.2.4 System Control Register

The system control register is an 8-bit read/write register that controls basing and sector table selection. Bits 7-3 are loaded by an LKA instruction only if bit 2 is equal to 1.

Bit 7: Enable System Data Segment. When equal to 1, it selects system data sector table on data read or write memory cycles.

Bit 6: Enable User Segments. When equal to 1, it selects user sector tables when in user mode.

Bit 5: Enable User Data Segment. When equal to 1, it selects the user data sector table when in user mode and when control bit 6 is equal to 1.

Bit 4: Enable Instruction Segment Base. This bit causes logical address to be based when set to 33 if the memory cycle is an instruction fetch and when the address is in the range from 0100000 to 0137777 when set.

Bit 3: Enable Data Segment Base. This bit causes logical address to be based when it is set to 1 if the memory cycle is a data cycle with an address in the range of 0100000 to 0137777 when set.

Bit 2: Control Load Enable. This bit is not saved in the control register but rather, when equal to 1 in the data being loaded, allows control register bits 7-3 to be loaded. If it is equal to 0 in the load data, only bits 0 and 1 are loaded. Bit 2 is always read back as a 1.

Bits 1-0: Sector Table Load Select. These bits determine which of the four sector tables is selected for a sector table load or read operation as follows:

00 System Instruction Sector Table
01 System Data Sector Table
10 User Instruction Sector Table
11 User Data Sector Table

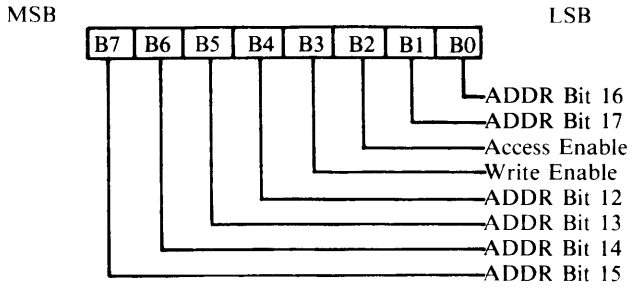
All control register bits are cleared to zero after the system is reset.

5.2.5 Base Register

The Base Register is an 8-bit, read/write register used in physical address generation. This register is added to the most significant 8-bits of the logical memory address if this feature is selected by the control register and if logical address is in the range from 0100000 to 0137777.

5.3 Sector Tables

The 8600 processor contains four sector tables: System Instruction, System Data, User Instruction, and User Data. Physical memory address is generated by one of the four segments depending on the machine state. The tables allow for complete separation of user and system programs and their respective data areas. In addition, the sector tables allow 4K-byte blocks of memory to be access protected (in user mode only) and/or write protected.



5.4 Address Generation

The processor transforms a 16-bit logical address into an 18-bit physical address through the use of the sector tables. Multiple sector tables allow 128K bytes accessibility in the system mode, and 128K bytes accessibility in the user mode. Each 128K byte area is further sub-divided into a 64K byte instruction area and a 64K byte data area. The translation mechanism is illustrated in Figure 5-3.

Addresses are generated based on the type of cycle (Input/Output or Memory) that the processor is to perform. Input/Output cycles use the 16 address/data lines directly to form the I/O address. Memory cycles use the S1 and S2 outputs to select one of the four sector tables and map address lines A12-A15 through a high speed RAM to produce a physical sector select of six bits.

The sector tables are formed via high speed 64 X 9 memory, which allows the based logical address to be transformed into a physical address. Parity is generated with the 8 bit transfer, and is compared to parity calculated on the 9 bit word that is present during memory accesses. Errors in the comparison produce a sector table parity error that provides a system level interrrupt to the processor.

5.5 Stack

The processor has a 32-entry stack that is located in main memory and is accessed through the System Data sector table. An entry always consists of two bytes. A special instruction (STKMV) allows multiple stacks to be located throughout memory. The stack pointer is decremented by one before each byte write and incremented by one following each byte read.

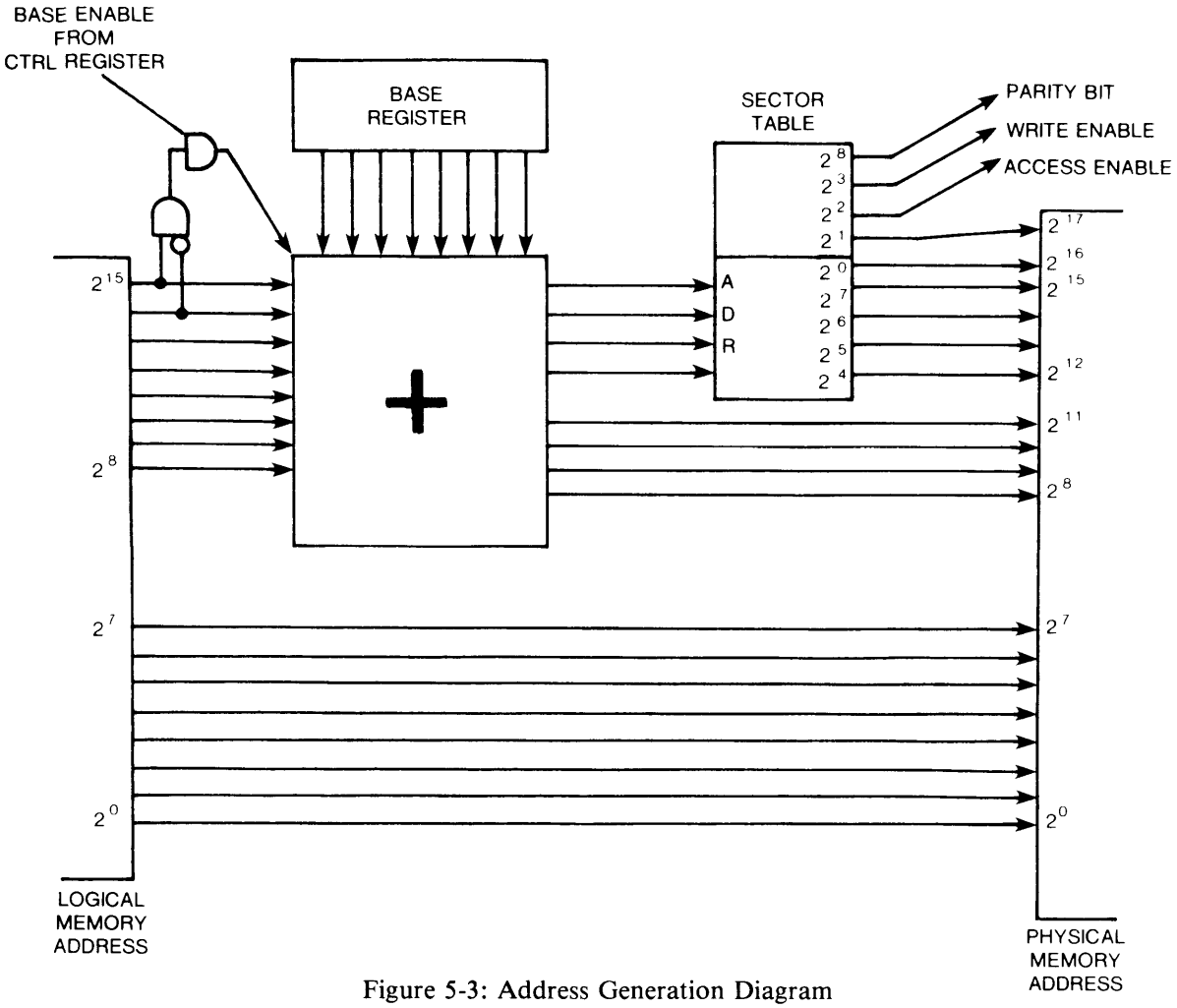


Figure 5-3: Address Generation Diagram

5.6 Input/Output

The processor has a maximum I/O space accessibility of 64K bytes. Base page I/O is provided to allow the more common peripherals to reside in the first 256 locations of I/O space. Peripherals in this area are accessed by means of the "short I/O" (CBSOUT and CBSIN) instructions that require less execution time at the system level than the I/O instructions that provide full 16 bit addresses.

5.7 Interrupts

The 8600 processor is capable of supporting four types of interrupts: millisecond, restart, system error, and vectored. If any of these interrupts is present and unmasked, an interrupt trap occurs. This means that normal instruction execution is interrupted, current machine state information is pushed on the stack, and the System Interrupt Enable, Millisecond Interrupt Enable, and User Mode bits are cleared; instruction execution resumes at a pre-defined address in system ROM. The format of the information pushed on the stack is shown in Figure 5-4.

The condition code save value shown has the same format as the output of the condition code save instruction.

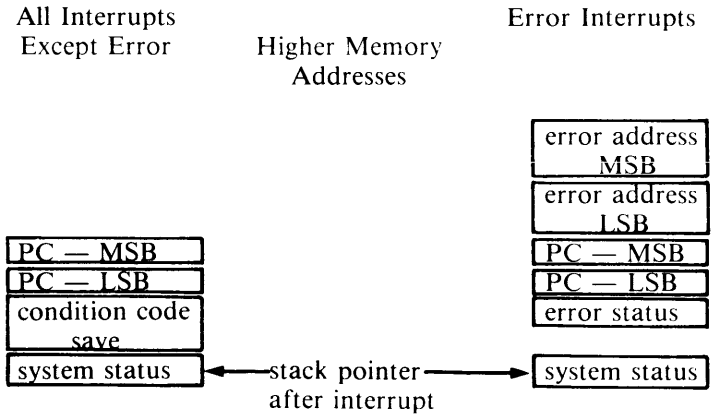


Figure 5-4: Stack Information After Interrupt

5.7.1 Millisecond Interrupt

The 8600 maintains a timer that runs continuously and times out every 1000 microseconds. When a time-out occurs, a millisecond interrupt request is made to the processor which is held until the interrupt is taken by the processor. The interrupt will only be taken if the Millisecond Interrupt Enable bit

in the system status is a 1 and the processor finishes executing an instruction. After the interrupt trap occurs, instruction execution resumes at 0170000 in system ROM.

5.7.2 Restart Interrupt

The Restart Interrupt allows the operator to regain control of the processor without having to perform a power-down/power-up procedure. Restart is not maskable by software, but it is masked after reset or restart until the execution of an IRET instruction unmasks it again. On the 8600, restart is initiated by a boot load or debug keyboard key sequence or by a thermal failure detected from the power supply or power fail alarm.

5.7.3 Error Interrupt

An Error Interrupt may occur during program execution due to a variety of fault conditions described below. The error interrupt trap causes an error status code and sometimes an error address to be pushed on the stack as shown in Figure 5-4. After the error interrupt trap occurs, instruction execution resumes at 0170004 in system ROM. System ROM uses the error status code on the stack to decode the error vector in system RAM to jump to; it also substitutes the condition code save value for the error status value on the stack before jumping to the error vector. See Part 9, System ROM, for more details.

The error address on the stack is the based-logical address of the memory cycle that actually gives rise to the error condition while the PC is the logical address of the instruction that was executing at the time. For priviledged op violation and unassigned instruction violation, the error address is not pushed on the stack since this will be the same as the PC value.

5.7.4 Vector Interrupt

Vector Interrupts allow the processor to respond directly to an interrupting common bus attachment by causing the processor to jump to a vector interrupt table in memory after the interrupt trap is taken. The format of this table is shown in Figure 5-5. The vector interrupt table base address must be on a 32-byte boundary in logical system instruction address space. Normally, a jump instruction pointing to the interrupt service routine is placed in the table for each device that asserts a common bus interrupt. (See Section 3.2.1 for these interrupt assignments.)

Vector Interrupt Table Base Address — Lower Memory Addresses

+ 000 common bus interrupt 0
+ 004 common bus interrupt 1
+ 010 common bus interrupt 2
+ 014 common bus interrupt 3
+ 020 common bus interrupt 4
+ 024 common bus interrupt 5
+ 030 common bus interrupt 6
+ 034 common bus interrupt 7

Higher Memory Addresses

Figure 5-5: Vector Interrupt Table Format

Vector Interrupts are implemented by a Programmed Interrupt Controller (PIC) that must be initialized before it can be used. On an 8600, the following initialization sequence should be used:

[VTABLE (lsb).AND.0340].OR.PICIW1 ⟶ CBPIC0
VTABLE (msb) ⟶ CBPIC1 I/O port adress
PICIW4 ⟶ CBPIC1 I/O port adress

where VTABLE is the 16-bit address of the 8-entry vector interrupt table with 4 bytes of executable code per entry and where:

CBPIC0	EQU	0000	I/O Port Address for PIC, Register 0
CBPIC1	EQU	0001	I/O Port Address for PIC, Register 1
PICEOI	EQU	0040	End of Interrupt Code
PICIW1	EQU	0037	PIC INIT WORD #1
PICIW4	EQU	0014	PIC INIT WORD #4

PIC interrupt is accomplished by writing a mask byte to address CBPIC1 after the interrupt controller has been initialized. The format of this mask byte is shown in Figure 5-6. Each bit in the mask byte inhibits the corresponding interrupt level when set to a 1 and enables the interrupt when set to 0.

I7	I6	I5	I4	I3	I2	I1	I0

Ix = Mask bit for Interrupt level x

Figure 5-6: Interrupt Mask Byte Format

When an interrupt trap occurs for an unmasked interrupt level, control is normally transferred to a software service routine. Before exiting this service routing (normally by an IRET instruction), the PIC must be cleared by sending the End-of-Interrupt as follows:

PICEOI ⟶ CBPIC0 (I/O port address)

5.8 RIM

The RIM provides standard interface to a Datapoint ARC system. Though the RIM physically resides on the CP/RIM board, it interfaces directly to the common bus and functions independently of other CP logic. The RIM appears to the processor as a peripheral device located within the 64K I/O space. Communication between the processor and RIM occurs through programmed I/O instructions.

The RIM is accessed via the Status/Mask Register located at I/O address 042 and the Command Register located at I/O address 043. The RIM buffer memory is located at I/O address 074000-077777. Two bytes, at I/O address 040 and 041, are used to enable or disable the various RIM buffers that can exist in the system. Address 040 is used to enable the CP/RIM module RIM buffer. Any data written to this address will enable the CP/RIM module RIM buffer. A write to

address 041 is required to disable it. Writing a 0 to 041 will disable all RIM buffers. I/O address 041 is used to enable RIM buffers that do not reside on the CP/RIM board. Five additional RIMs may exist within an 8600 so that a three-bit code will be used to select the proper buffer for the RIM. To enable one of the five additional RIMs, a value (001-005) is written to I/O address 041. This selects the RIM buffer indicated by the value and disables the CP/RIM buffer. Only one buffer is enabled at any time in the system. I/O addresses 044-055 are status and command for RIMs 1-5.

5.9 Direct Memory Access

The processor has the ability to release control of the system bus to allow other bus master devices to gain access to various bus slave devices. The bus may be relinquished following any memory cycle, at which time the processor ceases its execution. Execution is stopped, since the processor is a memory bound device. The processor assumes control of the bus when it has determined that no device requires the system bus.

5.10 System/Auxiliary ROM

System ROM contains power-up, interrupt, debug, and other routines that are required by the processing system. The 4K bytes of system ROM are addressed only in system mode at octal addresses 0170000-0177777. See Part 9 of this manual for a complete description of system ROM.

Auxiliary ROM is 4K bytes, starting at 0120000, 8K bytes starting at 0110000, or 12K bytes, starting at 0100000. It is used for the storage of diagnostic routines. Auxiliary ROM is mapped into system memory space through an I/O command only during the power-on-reset firmware sequence and when firmware diagnostics are run. It is also mapped in during firmware disk and tape loads. During normal system operation, this memory does not overlap other memory in the system.

5.11 Instruction Set

The 8600 processor supports the 6600 user mode instruction set. The instructions have been grouped into eight categories for convenience of presentation:

Category 1	Load Group
Category 2	Stack Control
Category 3	Byte Arithmetic
Category 4	Word Arithmetic
Category 5	Jumps, Calls, Returns
Category 6	I/O Group
Category 7	System Instructions
Category 8	String Operations

5.11.1 Presentation Format

A description of each 8600 instruction is given in the following sections. The symbols and abbreviations listed below are used in the presentation of the instructions.

Operation:	Symbolic representation of instruction description.
Op Code:	Operation Code, expressed in octal.
Entry:	Conditions necessary before execution.
Exit:	Conditions existing after execution.
()	The contents of.
⟶	Is transferred to.
:	Is compared with.
V	Logical "Or" operation.
-V—	Logical "Exclusive Or" operation.
—/|—	Logical "AND" operation.
|	Concatenation operator.
A, B, C, D, E, H, L, X	8-bit processor registers
M	Contents of memory location designated by the specified register pair. HL is used if no pair is designated.
P	Program counter.
P+X	Location relative to first byte of instruction.
Stack	The pushdown stack.
(OP)	One of eight ALU 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; however, A=022 may be used. Otherwise, the following values are used: B=0111, C=062, D=0113, E=0174, H=0115, L=0176, X=117.
rp	A register pair select op code. No byte is necessary for the selection of HL; however, HL=0176 may be used. Otherwise, the following values are used: BC=062, DE=0174, XA=022.
rp+1	BC=0113, DE=0115, HL=0117, XA=0111.
(vvv)	An 8-bit immediate value used in an instruction.
(adr)	A 16-bit immediate value used in an instruction with the LSB first, followed by the MSB.
(cf)	Condition flags (CZSP) (c=0 to 3).
(exp)	External command.
data	An expression reducing to an 8-bit immediate value.
loc	An expression reducing to a 16-bit address (LSB, MSB).
PPP	An 8-bit immediate value specifying a port number for I/O operations.

5.11.2 Category 1—Load Group

LOAD IMMEDIATE L(r)

Op Code: 0r6 (vvv)
Operation: (vvv)⟶(r)

This instruction transfers the value of the given operand to the register specified by bits 3-5 (d) in the Op Code. The condition flags are unaffected.

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

For L(rd)M, L(rd)M(rp)
Op Code: 3d7, rp 3d7 d<6
Operation: (M)⟶(rd)
For L(rd)(rs)
Op Code: 3ds
Operation: (rs)⟶(rd) s<6, d<6
For LM(rs), LM(rs)(rp)
Op Code: 37s, rp 37s s<6
Operation: (rs)⟶(M)

This instruction transfers the operand from the source specified by bits 0-2 of the Op Code to the destination specified by bits 3-5 (d) of the Op Code. The source is unaffected. If a memory access is required, the HL pair or the specified register pair will contain the memory address. The condition flags are unaffected.

PAGED LOAD PL

Mnemonic	Op Code
PL A, (loc)	105 LSB
PL B, (loc)	114 LSB
PL C, (loc)	124 LSB
PL D, (loc)	134 LSB
PL E, (loc)	144 LSB
PL H, (loc)	154 LSB
PL L, (loc)	164 LSB

Operation: (X | LSB)⟶r

These instructions load the register specified by bits 3-5 of the op code from the memory location specified by the data given in the instruction (LSB) and the X register (MSB). The flags are unaffected.

PAGED STORE PS

Mnemonic	Op Code
PS A, (loc)	107 LSB
PS B, (loc)	116 LSB
PS C, (loc)	126 LSB
PS D, (loc)	136 LSB
PS E, (loc)	146 LSB
PS H, (loc)	156 LSB
PS L, (loc)	166 LSB

Operation: r⟶(X | LSB)

These instructions store the register specified by bits 3-5 of the op code in the memory location specified by the data given in the instruction (LSB) and the X register (MSB). The flags are unaffected.

DOUBLE LOAD DL

Mnemonic	Op Code
DL DE,HL	047
DL BC,HL	111 047
DL BC,BC	062 047
DL BC,DE	113 047
DL DE,BC	174 047
DL DE,DE	115 047
DL HL,BC	176 047
DL HL,DE	117 047
DL HL,HL	057

Operation: (M, M+1)⟶rp

These instructions load the register pair specified by the first operand from the memory locations pointed to by the register pair specified by the second operand. The LSB register (C, E, or L) is loaded from the specified memory location, and the MSB register (B, D, or H) is loaded from the next higher memory location. The flags are unaffected.

DOUBLE STORE DS

Mnemonic	Op Code
DS DE,HL	027
DS BC,HL	111 027
DS BC,DE	113 027
DS DE,BC	174 027
DS HL,BC	176 027
DS HL,DE	117 027

Operation: rp⟶(M, M+1)

These instructions store the register pair specified by the first operand in the memory locations pointed to by the register pair specified by the second operand. The LSB register (C, E, or L) is stored in the specific memory location, and the MSB register(B, D, or H) is stored in the next higher location. The flags are unaffected.

DOUBLE PAGED LOAD DPL

Mnemonic	Op Code
DPL BC, (loc)	111 124 LSB
DPL DE, (loc)	113 144 LSB
DPL HL, (loc)	115 164 LSB

Operation: (X | LSB, X | LSB+1)⟶rp

These instructions load the specified register pair from the memory locations specified by the data given in the instruction (LSB) and the X register (MSB). The C, E, or L register is loaded from the specified memory location, and the B, D, or H registeris loaded from the next higher location. The flags are unaffected.

DOUBLE PAGED STORE DPS

Mnemonic	Op Code
DPS BC, (loc)	111 126 LSB
DPS DE, (loc)	113 146 LSB
DPS HL, (loc)	115 166 LSB

Operation: rp⟶(X | LSB, X | LSB+1)

These instructions store the specified register pair in the locations specified by the data given in the instruction (LSB) and the X register (MSB). 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. The flags are unaffected.

DOUBLE PAGED LOAD REVERSED DPLR(rp),loc

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

Operation: (X | LSB+1, X | LSB)⟶rp

These instructions load the specified register pair from the memory locations specified by the data given in the instruction (LSB) and the X register (MSB). 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. This is similar to the DPL instruction except that the order in which the registers are loaded is reversed. The flags are unaffected.

DOUBLE PAGED STORE REVERSED DPSR(rp),loc

Mnemonic	Op Code
DPSR BC, loc	062 116 LSB
DPSR DE, loc	174 136 LSB
DPSR HL, loc	176 156 LSB

Operation: rp⟶(X | LSB+1, X | LSB)

These instructions store the specified register pair into the locations specified by the data given in the instruction (LSB) and the X register (MSB). The B, D, or H register is stored into the specified memory location, and the C, E, or L register is stored in the next higher location. This is similar to the DPS instruction except that the order in which the registers are stored is reversed. The flags are unaffected.

REGISTER STORE REGS

Op Code: 055

This instruction stores all of the registers for the currently selected mode in the field pointed to by the top entry of the stack. The registers are stored in reverse, decrementing the top entry in the stack after each register is stored. When the instruction terminates, the top entry of the stack will be the address of the last byte minus one.

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. The flags and the contents of the registers are not affected.

REGISTER LOAD REGL

Op Code: 111 055

This instruction loads all of the registers for the currently selected mode from the field pointed to by HL. The registers are loaded in reverse order, decrementing the address after each byte is transferred. In this manner, the registers can be reloaded from values stored by the REGS instruction. In the example given for the REGS instruction, if the REGL instruction were entered with HL=02007, the registers shown would be loaded from the locations shown. The condition flags are not affected by this instruction.

5.11.3 Category 2—Stack Control

POP POP(rp)

Op Code: 060 or rp 060
Operation: (Stack)⟶rp

This instruction pops the most recent stack entry into the specified register pair. If no pair is specified, the destination defaults to the HL register pair. The flags are unaffected.

PUSH PUSH(rp)

Op Code: 070 or rp 070
Operation: rp⟶Stack

This instruction pushes the contents of the specified register pair onto the stack. If no pair is specified, the source defaults to the HL register pair. The flags are unaffected.

PUSH IMMEDIATE PUSH loc

Op Code: 051 (adr)
Operation: (adr)⟶Stack

This instruction pushes the value of the operand onto the stack. The flags are unaffected.

STACK STORE STKS

Op Code: 065

This instruction POPs a specified number of stack entries and stores them (LSB followed by MSB) in the field pointed to by HL. HL is incremented after each byte is transferred. C is decremented after each stack entry is stored. This instruction is interruptible after each stack entry is stored.

Entry: HL = first location in the storage area.
C = the number of entries to be POPped and stored (1 through 16; 0 or 16 implies 16).
Exit: HL = address of last byte stored plus one.
C = modulo 16 zero.

Condition flags are not affected.

STACK LOAD STKL

Op Code: 111 065

This instruction pushes onto the stack the specified number of entries from the field pointed to by HL. The entries are loaded in reverse order to allow restoring the stack from locations stored using the STKS instruction. HL is decremented after each transfer; C is decremented after each stack entry. This instruction is interruptible after each byte is stored on the stack.

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 = address of the last byte pushed minus one.
C = modulo 16 zero.

Condition flags remain unchanged.

5.11.4 Category 3—Byte Arithmetic, A Register

ADD IMMEDIATE Ad (rd) data

Op Code: 004 (vvv), r 004 (vvv)
Operation: (r) + vvv⟶r

This instruction adds the value of the operand to the specified register. If no register is specified, the A register is used. All flags are set based on the result of the ADD operation.

ADD AD(rs), AD(rs)(rd)
ADM, ADM (rd),

For AD(rs), r AD(rs)
Op Code: 20s, r 20s
Operation: (r) + (rs)⟶r
For ADM, r ADM
Op Code: 207, r 207
Operation: (r) + (M)⟶r

This instruction adds the value of the source (rs or M) to the specified register. If no register is specified, the A register is used. All flags are set based on the result of the ADD operation.

ADD WITH CARRY IMMEDIATE AC (rd) data

Op Code: 014 (vvv), r 014 (vvv)
Operation: (r) + vvv + Carry⟶r

Adds the Carry bit and value of the operand to the specified register. If no register is specified, the A register is used. All flags are set based on the result of the operation.

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

For AC(rs), r AC(rs)
Op Code: 21s, r 21s
Operation: (r) + Carry + (rs)⟶r
For ACM, r ACM
Op Code: 217, r 217
Operation: (r) + Carry + (M)⟶r

Adds the Carry bit and the source (rs or M) to the specified register. If no register is specified, the A register is used. All flags are set based on the result of the operation.

SUBTRACT IMMEDIATE SU (rd) data

Op Code: 024 (vvv), r 024 (vvv)
Operation: (r) - vvv⟶r

Subtracts the value of the operand from the contents of the specified register. If no register is specified, the A register is used. All flags are set based on the result of the operation. A borrow out sets the carry flag.

SUBTRACT SU(rs), SU(rs)(rd)
SUM, SUM(rd),

For SU(rs), r SU(rs)
Op Code: 22s, r 22s
Operation: (r)-(rs)⟶r
For SUM, r SUM
Op Code: 227, r 227
Operation: (r)-(M)⟶r

Subtracts the source (rs or M) from the specified register. If no register is specified, the A register is used. All flags are set based on the result of the operation. A borrow out sets the carry flag.

SUBTRACT WITH BORROW IMMEDIATE SB (rd) data

Op Code: 034 (vvv), r 034 (vvv)
Operation: (r)-vvv-Carry⟶r

Subtracts the value of the operand and the Carry bit from the specified register. If no register is specified, the A register is used. All flags are set based on the result of the operation. If a borrow out occurs, the Carry flag is set.

SUBTRACT WITH BORROW — SB(rs), r SB(rs)
SBM, r SBM

For SB(rs), r SB(rs)
Op Code: 23s, r 23s
Operation: (r)-(rs)-Carry⟶r
For SBM, r SBM
Op Code: 237, r 237
Operation: (r)-(M)-Carry⟶r

Subtracts the source (rs or M) and the Carry bit from the specified register. If no register is specified, the A register is used. All flags are set based on the result of the operation. If a borrow out occurs, the Carry flag is set.

AND IMMEDIATE — ND data, r ND data

Op Code: 044 (vvv), r 044 (vvv)
Operation: (r) —/|— vvv⟶r

Forms the logical product of the specified register with the value of the operand. If no register is specified, the A register is used. The Carry flag is reset upon completion. All other flags are set based on the result of the operation.

AND — ND(rs), r ND(rs)
NDM, r NDM

For ND(rs), r ND(rs)
Op Code: 24s
Operation: (r) —/|— (rs)⟶r
For NDM, r NDM
Op Code: 247, r 247
Operation: (r) —/|— (M)⟶r

Forms the logical product of the specified register with the source (rs or M). If no register is specified, the A register is used. The Carry flag is reset upon completion. All other flags are set based on the result of the operation.

OR IMMEDIATE — OR data, r OR data

Op Code: 064 (vvv), r 064 (vvv)
Operation: (r) V vvv⟶r

Forms the logical sum of the specified register and the value of the operand. If no register is specified, the A register is used. The Carry flag is reset upon completion. All other flags are set based on the result of the operation.

OR — OR(rs), r OR(rs)
ORM, r ORM

For OR(rs), r OR(rs)
Op Code: 26s, r 26s
Operation: (r) V (rs)⟶r
For ORM, r ORM
Op Code: 267, r 267
Operation: (r) V (M)⟶r

Forms the logical sum of the specified register with the source (rs or M). If no register is specified, the A register is used. The Carry flag is reset upon completion. All other flags are set based on the result of the operation.

EXCLUSIVE OR IMMEDIATE — XR data, r XR data

Op Code: 054 (vvv), r 054 (vvv)
Operation: (r) -V— vvv⟶r

Forms the logical difference of the specified register and the value of the operand. If no register is specified, the A register is used. The Carry flag is reset upon completion. All other flags are set based on the result of the operation.

EXCLUSIVE OR — XR(rs), r XR(rs)
XRM, r XRM

For XR(rs), r XR(rs)
Op Code: 25s, r 25s
Operation: (r) -V— (rs)⟶r
For XRM, r XRM
Op Code: 257, r 257
Operation: (r) -V— (M)⟶r

Forms the logical difference of the specified register from the source (rs or M). If no register is specified, the A register is used. The Carry flag is reset upon completion. All other flags are set based on the result of the operation.

COMPARE IMMEDIATE — CP data, r CP data

Op Code: 074 (vvv), r 074 (vvv)
Operation: (r) : vvv

The value of the operand is subtracted from the specified register but the results are not restored. If no register is specified, the A register is used. All flags are set as if the instruction had been a SUBTRACT IMMEDIATE.

COMPARE — CP(rs), r CP(rs)
CPM, r CPM

For CP(rs), r CP(rs)
Op Code: 27s, r 27s
Operation: (r):(rs)

For CPM, r CPM
Op Code: 277, r 277
Operation: (r):(M)

The source (rs or M) is subtracted from the specified register but the results are not restored. If no register is specified, the A register is used. All flags are set as if the instruction had been a SUBTRACT.

SHIFT LEFT CIRCULAR — SLC, r SLC

Op Code: 002, r 002
Operation: r(N-1)⟶r(N), r7⟶r0, r7⟶Carry

Shifts the contents of the specified register left one bit in a circular fashion. The most significant bit is moved to the least significant bit, as well as to the Carry bit. If no register is specified, the A register is used. The Carry bit is the same as bit 0 upon completion. Zero, Sign, and Parity flags are not affected.

SHIFT RIGHT CIRCULAR SRC, r SRC

Op Code: 012, r 012
Operation: r(N)→r(N-1), r0→r7, r0→Carry

Shifts the contents of the specified register right one bit in a circular fashion. The least significant bit is moved into the most significant bit, as well as to the Carry bit. If no register is specified, the A register is used. The Carry bit is the same as bit 7 upon completion. The Zero, Parity, and Sign flags are not affected by this instruction.

SHIFT RIGHT EXTENDED SRE, r SRE

Op Code: 032, r 032
Operation: r(N)→r(N-1), Carry→r7, r0→Carry

The specified register is shifted right one bit with the most significant bit being replaced by the Carry bit and the least significant bit replacing the Carry bit. If no register is specified, the A register is used. The carry bit is set as described above. The Zero, Parity, and Sign flags are not affected by this instruction.

SINGLE PAGED
TO REGISTER OPERATIONS P(op)(r),loc

Mnemonic		Op Code
PAD	(r),loc	|r| 106 LOCLSB
PAC	(r),loc	|r| 112 LOCLSB
PSU	(r),loc	|r| 122 LOCLSB
PSB	(r),loc	|r| 132 LOCLSB
PND	(r),loc	|r| 142 LOCLSB
PXR	(r),loc	|r| 152 LOCLSB
POR	(r),loc	|r| 162 LOCLSB
PCP	(r),loc	|r| 172 LOCLSB

These instructions perform the indicated operation between the 8-bit value in the memory location specified by the last byte in the instruction (LSB) and the X register (MSB) and the 8-bit value in the specified register with all, except the comparison operation, depositing the result in the specified register. The COMPARE instruction performs the subtract operation but does not store the result. All flags are set based on the result of the specified operation. The logical operations cause the Carry flag to be reset.

5.11.5 Category 4—Word Arithmetic

INCREMENT REGISTER PAIR INCP

Mnemonic	Op Code
INCP HL	015
INCP HL,2	117 015
INCP HL,A	017
INCP BC	062 015
INCP BC,2	113 015
INCP BC,A	062 017
INCP DE	174 015
INCP DE,2	115 015
INCP DE,A	174 017
INCP XA	022 015
INCP XA,2	111 015
INCP XA,A	022 017

These instructions increment the indicated register pair by one, two, or the contents of the A register. The increment value is added to the LSB register and then the carry is added to the MSB register. The Carry flag reflects the carry out of the MSB. All other flags are indeterminate.

DECREMENT REGISTER PAIR DECP

Mnemonic	Op Code
DECP HL	035
DECP HL,2	117 035
DECP HL,A	037
DECP BC	062 035
DECP BC,2	113 035
DECP BC,A	062 037
DECP DE	174 035
DECP DE,2	115 035
DECP DE,A	174 037
DECP XA	022 035
DECP XA,2	111 035
DECP XA,A	022 037

These instructions decrement the indicated register pair by one, two, or the contents of the A register. The decrement value is subtracted from the LSB register and then the borrow is subtracted from the MSB register. The Carry flag reflects the borrow out of the MSB. All other flags are indeterminate.

DOUBLE MEMORY
TO REGISTER OPERATIONS D(op)M(rp)

Mnemonic		Op Code
DADM	(rp)	| rp | 013
DACM	(rp)	| rp | 310
DSUM	(rp)	| rp | 033
DSBM	(rp)	| rp | 330
DNDM	(rp)	| rp | 043
DXRM	(rp)	| rp | 053
DORM	(rp)	| rp | 063
DCPM	(rp)	| rp | 073

These instructions perform the indicated operation between the 16-bit value at the memory location pointed to by the HL register pair (LSB at the location pointed to and MSB at the next higher location) and the 16-bit value in the specified register pair (BC, DE, HL, or XA). In subtraction and comparison, the value in the register pari. In all operations except comparison, the result is deposited in the register pair. The Carry, Sign, and Zero condition flags reflect the entire 16-bit result. The logical operations cause the Carry flag to be reset.

DOUBLE PAGED
TO REGISTER OPERATIONS D(op)P(rp),loc

Mnemonic		Op Code
DADP	(rp),loc	| rp+1 | 013 LOCLSB
DACP	(rp),loc	| rp+1 | 310 LOCLSB
DSUP	(rp),loc	| rp+1 | 033 LOCLSB
DSBP	(rp),loc	| rp+1 | 330 LOCLSB
DNDP	(rp),loc	| rp+1 | 043 LOCLSB
DXRP	(rp),loc	| rp+1 | 053 LOCLSB
DORP	(rp),loc	| rp+1 | 063 LOCLSB
DCPP	(rp),loc	| rp+1 | 073 LOCLSB

These instructions perform the indicated operation between the 16-bit value at the memory pointed to by X (MSB) and LOCLSB (LSB), and the 16-bit value in the specified register pair. In subtraction and comparison, the value in memory is substracted from the value in the register pair. In all operations except comparison, the result is deposited in the register pair. Carry, Sign, and Zero reflect the entire 16-bit result. The logical operations cause the Carry flag to be reset.

DOUBLE IMMEDIATE
TO REGISTER OPERATIONS D(op)I(rp),data

Mnemonic		Op Code
DADI	(rp),data	| rp | 110 LSB MSB
DACI	(rp),data	| rp | 311 LSB MSB
DSUI	(rp),data	| rp | 130 LSB MSB
DSBI	(rp),data	| rp | 331 LSB MSB
DNDI	(rp),data	| rp | 140 LSB MSB
DXRI	(rp),data	| rp | 150 LSB MSB
DORI	(rp),data	| rp | 160 LSB MSB
DCPI	(rp),data	| rp | 170 LSB MSB

These instructions perform the indicated operation between the 16-bit operand (LSB, MSB) following the Op Code and the 16-bit value in the specified register pair. In subtraction and comparison, the value of the operand is subtracted from the value in the register pair. In all operations except comparison, the result is deposited in the register pair. Carry, Sign, and Zero reflect the entire 16-bit result. The logical operations cause the Carry flag to be reset.

DOUBLE REGISTER
TO MEMORY OPERATIONS DM(op)(rp)

Mnemonic	Op Code
DMAD(rp)	| rp+1 | 110
DMAC(rp)	| rp+1 | 311
DMSU (rp)	| rp+1 | 130
DMSB (rp)	| rp+1 | 331
DMND(rp)	| rp+1 | 140
DMXR(rp)	| rp+1 | 150
DMOR(rp)	| rp+1 | 160
DMCP (rp)	| rp+1 | 170

These instructions perform the indicated operation between the specified registe pair and 16-bit value pointed to by the HL register pair. The LSB is at the indicated location, and the MSB is at the next higher location. In subtraction and comparison, the value in the register pair is subtracted from the value at the memory location. In all operations except comparison, the result is deposited in the memory location pointed to by the HL pair (LSB, MSB). Carry, Sign, and Zero reflect the entire 16-bit result. The logical operations cause the Carry flag to be reset.

DOUBLY LINKED LIST DELETE LLDEL

Op Code: 111 051

A doubly linked list construct appears as follows:

ITEM1	DA	ITEM2	forward pointer
	DA	ITEM3	backward pointer
ITEM2	DA	ITEM3	forward pointer
	DA	ITEM1	backward pointer
ITEM3	DA	ITEM1	forward pointer
	DA	ITEM2	backward pointer
ITEM4	DA	00000	item to be inserted
	DA	00000	

When the linked list delete instruction is performed with HL pointing to ITEM2, the instruction deletes ITEM2 from the list by moving its forward pointer to the forward pointer of ITEM1 and its backward pointer to the backward pointer of ITEM3. When the instruction completes, the entry value of HL has not been changed, while the DE register is left pointing to ITEM1 and BC is left pointing to ITEM3. The flags are not affected.

DOUBLY LINKED
LIST INSERT LLINS

Op Code: 062 051

This instruction inserts a list item into a linked list construct. Using the example shown for the LLDEL instruction, if the insert instruction is performed with DE pointing to ITEM2 and HL pointing to ITEM4, the instruction exits with ITEM2's forward pointer pointing to ITEM4, ITEM4's forward pointer pointing to ITEM3, ITEM3's backward pointer pointing to ITEM4, and ITEM4's backward pointer pointing to ITEM2. Finally, the entry values of the DE and HL register are unchanged and the BC register is left pointing to ITEM3. The flags are not affected.

INTEGER MULTIPLY: IMULT
HLDE = HL * BC

Op Code: 111 011

This instruction multiplies the unsigned values in HL and BC, putting an unsigned result in the HLDE register quadruple (MSB in H and LSB in E). The A, B, C, and X registers are not changed by this instruction. The Zero flag reflects the result in the HL register pair. The Carry flag is set if the sign bit in the D register is a one. Sign and Parity flags are undefined.

DOUBLE INTEGER DIVIDE: DIDIV
HLDE/BC => Q(DE),R(HL)

Op Code: 111 031

This instruction produces an error indication with the Carry condition flag set if the BC register pair is less than or equal to the HL register pair (in unsigned arithmetic). Otherwise, it divides the unsigned HLDE register quadruple by the BC register pair, placing the quotient in the DE register pair and the remainder in the HL register pair. The A, B, C, and X registers are unchanged by this instruction. The Carry flag is cleared to indicate a successful division. The Zero flag reflects the remainder in the HL register pair. The Sign and Parity flags are undefined.

INTEGER DIVIDE: IDIV
DE/BC => Q(DE)R(HL)

Op Code: 062 031

This instruction loads the HL register pair with zeros, then it divides the unsigned quadruple HLDE by the BC register pair, placing the quotient in the DE register pair and the remainder in the HL register pair. The A, B, C, and X registers are unchanged by this instruction. The Carry flag is reset, while the Zero flag reflects the remainder in the HL pair. The Sign and Parity flags are undefined.

TWOs COMPLEMENT
A REGISTER PAIR COMP(rp)

Mnemonic	Op Code
COMP BC	062 011
COMP DE	174 011
COMP HL	176 011

This instruction first clears bits 0-6 of the A register. If the sign bit of the A register is set, this instruction performs a 2's complement upon the specified register pair. The flags are undefined.

TWOs COMPLEMENT
A REGISTER PAIR COMPS(rp)

Mnemonic	Op Code
COMPS BC	113 011
COMPS DE	115 011
COMPS HL	117 011

This instruction first clears bits 0-5 of the A register and duplicates bit 7 in bit 6. If the sign bit (bit 7) is set, this instruction performs a 2's complement upon the specified register pair. The flags are undefined.

INCREMENT AND DECREMENT INDEX INCI,DECI

Mnemonic	Op Code
INCI (disp),(index)	005 LSB(i)
DECI (disp),(index)	025 LSB(i)
INCI* (disp),(index)	111 005 LSB MSB(i)
DECI* (disp),(index)	111 025 LSB MSB(i)

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 LSB of the index value, the MSB being in the next higher memory location. The LSB of the index address is specified by (i), while the MSB of index address is specified by the X register. The instruction also contains a displacement, shown as (disp) above, that is either one or two bytes in length, depending upon the op code. These instructions either increment or decrement the value of the index by the value of the displacement. The Carry flag is set based on the 16-bit result. The other flags are indeterminate.

LOAD FROM INDEX
INCREMENTED OR DECREMENTED LFII,LFID

Mnemonic		Op Code
LFII BC,	(disp),(index)	062 005 LSB(i)
LFID BC,	(disp),(index)	062 025 LSB(i)
LFII BC,*	(disp),(index)	113 005 LSB MSB(i)
LFID BC,*	(disp),(index)	113 025 LSB MSB(i)
LFII DE,	(disp),(index)	174 005 LSB(i)
LFID DE,	(disp),(index)	174 025 LSB(i)
LFII DE,*	(disp),(index)	115 005 LSB MSB(i)
LFID DE,*	(disp),(index)	115 025 LSB MSB(i)
LFII HL,	(disp),(index)	176 005 LSB(i)
LFID HL,	(disp),(index)	176 025 LSB(i)
LFII HL,*	(disp),(index)	117 005 LSB MSB(i)
LFID HL,*	(disp),(index)	117 025 LSB MSB(i)

The processor has a construct called an index, which is a 16-bit value kept in memory. The concept is similar to that of index registers, except all the values are kept in a page of memory pointed to by the X register. The index is specified by a single byte in the instruction, shown as (i), which points to the memory location containing the LSB of the index value, the MSB being the next higher memory location. The letter (i) specifies the LSB of the index address, while the X register specifies the MSB of the index address. The instruction also contains a displacement, shown as (disp) above, that is either one or two bytes in length, depending on the opcode.

These instructions are similar to the INCI and DECI instructions, except that they increment or decrement the value of the index and deposit it in the specified register. The Carry flag is set based on the 16-bit result in the specified register pair. The other flags are indeterminate.

5.11.6 Category 5—Jumps, Calls, Returns

UNCONDITIONAL JUMP — JMP loc

Op Code: 104 (adr)
Operation: (adr)→P

This instruction represents an unconditional transfer of control. The second byte of the instruction represents the LSB of the jump address, while the third byte of the instruction represents the MSB. The flags are not affected.

JUMP IF CONDITION TRUE — JT(cf) loc

Op Code: 1(c + 4) 0 (adr)
Operation: If condition true, (adr)→P

This instruction examines the flag designated by c. If the flag is set, control is transferred to (adr). If it is reset, the next sequentially available instruction is executed. The flags are not affected.

JUMP IF CONDITION FALSE — JF(cf) loc

Op Code: 1c 0 (adr)
Operation: If condition false, (adr)→P

This instruction examines the flag designated by c. If the flag is reset, control is transferred to (adr). If it is set, the next sequentially available instruction is executed. The flags are not affected.

NOP JUMP — NOJ loc

Op Code: 045 (adr)
Operation: P + 3→P

This instruction increments the P-counter twice. 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 occur while the jump is being overstored. The flags are not affected.

SUBROUTINE CALL — CALL loc

Op Code: 106 (adr)
Operation: P + 3→Stack, (adr)→P

This instruction transfers the address of the next sequentially available instruction to the pushdown stack, and transfers control to the address specified by the contents of the two memory locations immediately following the Op Code. The first byte following the Op Code is the LSB of the address and the next byte is the MSB. The flags are not affected.

SUBROUTINE CALL
IF CONDITION TRUE — CT(cf) loc

Op Code: 1(c + 4) 2 (adr)
Operation: If condition true, P + 3→Stack, (adr)→P

Examines the flag designated by c. If the flag is set, the instruction transfers the address of the next sequentially available instruction to the pushdown stack and transfers control to (adr). If it is reset, the next sequentially available instruction is executed. The flags are not affected.

SUBROUTINE CALL
IF CONDITION FALSE — CF(cf) loc

Op Code: 1c2 (adr)
Operation: If condition false, P + 3→Stack, (adr)→P

This instruction examines the flag designated by c. If the flag is reset, the instruction transfers the address of the next sequentially available instruction to the pushdown stack and transfers control to (adr). If it is set, the next sequentially available instruction is executed. The flags are not affected.

SUBROUTINE RETURN — RET

Op Code: 007
Operation: (Stack)→P

This instruction transfers control to the address specified by the most recent entry in the pushdown stack and deletes that entry from the stack. The flags are not affected.

SUBROUTINE RETURN
IF CONDITION TRUE — RT(cf)

Op Code: 0 (c + 4)3
Operation: If condition true, (Stack)→P

This instruction examines the flag designated by c. If the flag is set, control is transferred to the address specified by the most recent entry in the pushdown stack and that entry is deleted. If the flag is reset, the next sequentially available instruction is executed. The flags are not affected.

SUBROUTINE RETURN
IF CONDITION FALSE — RF (cf)

Op Code: 0c 3
Operation: If condition false, (Stack)→P

This instruction examines the flag designated by c. If the flag is reset, control is transferred to the address specified by the most recent entry in the pushdown stack and that entry is deleted. If the flag is set, the next sequentially available instruction is executed. The flags are not affected.

SYSTEM CALL SC

Op Code: 067

This instruction clears the bits in the system status register, the User Mode System Interrupt Enable, and Millisecond Interrupt Enable flags. It pushes the current program counter, status, and flags onto the stack and performs a jump to 170020 in system ROM. This is the mechanism by which the user would communicate with the operating system incorporating the User mode. The flags are not affected.

BREAKPOINT BP

Op Code: 052

This instruction clears the User Mode, System Interrupt Enable, and Millisecond Interrupt Enable flags. It pushes the current program counter, status, and flags onto the stack and performs a jump to 170024 in system ROM. The flags are not affected.

USER RETURN UR

Op Code: 111 102

This instruction sets the User Mode flag, transfers control to the most recent entry on the stack, and deletes that entry from the stack. This is a privileged instruction. The flags are not affected.

ENABLE INTERRUPTS AND JUMP EJMP(loc)

Op Code: 111 050 (adr)

This instruction sets the System Interrupt Enable flag and Millisecond Interrupt Enable flag, enabling interrupts after the next instruction. It then transfers control to (adr). This is a privileged instruction. The flags are not affected.

ENABLE INTERRUPTS AND RETURN EUR

Op Code: 062 050

This instruction sets the System Interrupt Enable flag, Millisecond Interrupt Enable flag, and the User Mode flag. It then transfers control to the most recent entry on the stack and deletes that entry. This is a privileged instruction. The flags are not affected.

INTERRUPT RETURN IRET

Op Code: 161

This instruction POPs the top entry of the stack into the system status register and flags. It then executes an unconditional return to the address specified by the second entry on top of the stack, enabling interrupts after the next instruction. This is a privileged instruction. The flags are not affected.

5.11.7 Category 6—I/O Group

EXECUTE BEEP EX BEEP

Op Code: 151

This instruction activates a tone producing mechanism. It causes the system status to be saved and control to be transferred to system ROM. The ROM then activates a tone producing mechanism. CAUTION: Three stack entries are used during execution.

EXECUTE CLICK EX CLICK

Op Code: 153

This instruction activates an audible click producing mechanism. It causes the system status to be saved and control to be transferred to system ROM. The ROM then activates a click producing mechanism. CAUTION: Three stack entries are used during execution.

COMMON BUS OUTPUT CBOUT

Op Code: 145, r 145

This instruction places the contents of the specified register on the data bus while placing the DE pair on A15-A0 as a port number. If no register is specified, the A register is used. This instruction is used to operate on peripheral attachments used on the Datapoint common bus (CB). This is a privileged instruction. The flags are not affected.

COMMON BUS INPUT CBIN

Op Code: 141, r 141

This instruction reads the contents of the data bus into the specified register after placing DE on A15-A0 as a port number. If no register is specified, the A register is used. This instruction is used to operate on peripheral attachments used on the CB. This is a privileged instruction. The flags are not affected.

COMMON BUS SHORT OUTPUT CBSOUT

Op Code: r 147 PPP

This instruction places the specified register on the data bus and uses PPP as a port number (forces zero on upper eight bits). If no register is specified, the A register is used. This instruction can only deal with the first 256 ports. This is a privileged instruction; the flags are not affected.

COMMON BUS SHORT INPUT CBSIN

Op Code: r 143 PPP

This instruction sends PPP (forces zero on upper eight bits) out as a port number and reads the data bus into the specified register. If no register is specified, the A register is used. This instruction can only deal with the first 256 ports. This is a privileged instruction; the flags are not affected.

COMMON BUS BLOCK INPUT — BLKIN

Op Code: 171

This instruction copies a block of bytes from the I/O address specified by the DE register pair to the memory address specified by the HL register pair. The block length is contained in the C register. Transfer will stop if a character is stored which is equal to the 2's complement of the B register, if B is not equal to zero. The HL and DE register pairs are incremented, and C is decremented after each byte is transferred. This instruction is interruptible after each byte is transferred. It is a privileged instruction.

Entry: HL = memory address of first destination byte.
DE = I/O address of first source byte.
C = number of bytes to move (C = 1 to 255; C = 0 implies 256).
B = 2's complement of terminating character.
B = 0 implies no terminating character.

Exit: HL = destination address of last byte stored plus one.
DE = source address of last byte stored plus one.
C = zero or count before termination character found.
B, A, X = entry value.

The flags are not affected.

COMMON BUS BLOCK OUTPUT — BLKOUT

Op Code: 173

This instruction copies a block of bytes from the memory address specified by the HL register pair to the I/O address specified by the DE register pair. The block length is contained in the C register. Transfer will stop if a character is copied which is equal to the 2's complement of the B register, if B is not equal to zero. The HL and DE register pairs are incremented, and the C register is decremented after each transfer. This instruction is interruptible after each byte is transferred. It is a privileged instruction.

Entry: HL = memory address of first source byte.
DE = I/O address of first destination byte.
C = number of bytes to be transferred (C = 1 to 255, C = 0 implies 256).
B = 2's complement of terminating character.
(B = 0 implies no terminating character.)

Exit: HL = source address of last byte moved plus one.
DE = destination address of last byte moved plus one.
C = zero or count before terminating character found.
B, A, X = entry value.

The flags are not affected.

COMMON BUS MODIFIED BLOCK INPUT — MBLKIN

Op Code: 111 171

This instruction copies a block of bytes from the I/O address specified by the DE register pair to the memory address specified by the HL register pair. The block length is contained in the C register. Transfer will stop after a character is stored which is equal to the 2's complement of the B register, if B is not equal to zero. The HL register pair is incremented, and the C register is decremented after each byte is transferred. This instruction is privileged and is interruptible after each byte is transferred.

Entry: HL = memory address of first destination byte.
DE = I/O address of the source.
C = number of bytes to be moved (C = 1 to 255; C = 0 implies 256).
B = 2's complement of the terminating character.
(B = 0 implies no terminating character).

Exit: HL = destination address of last byte stored plus one.
DE = entry value.
C = zero or count before terminating character is found.
B, A, X = entry value.

The flags are not affected. This instruction is designed to assist in programming the MIFM common bus attachment.

COMMON BUS MODIFIED BLOCK OUTPUT — MBLKOUT

Op Code: 111 173

This instruction copies a block of bytes from the memory address specified by the HL register pair to the I/O address specified by the DE register pair. The block length is contained in the C register. Transfer will stop after a character is copied which is equal to the 2's complement of the B register, if B is not equal to zero. The HL register pair is incremented and the C register is decremented after each byte is transferred. This instruction is privileged and is interruptible after each byte is transferred.

Entry: HL = memory address of the first source byte.
DE = I/O address of the destination.
C = number of bytes to be transferred (C = 1 to 255, C = 0 implies 256).
B = 2's complement of the terminating character.
(B = 0 implies no terminating character).

Exit: HL = memory address of last byte transferred plus one.
DE = entry value.
C = zero or count before terminating character is found.
B, A, X = entry value.

The flags are not affected. This instruction is designed to assist in programming the MIFM common bus attachment.

5.11.8 Category 7—System Instructions

SYSTEM INFORMATION — INFO

Op Code: 111 010

This instruction is used to differentiate the 8600 from other Datapoint processors. In the 5500, this instruction performs no operation. In the 6600, this instruction loads a 1 into the A register and the revision number of the microcode ROM into the B register. In the 8600, INFO loads the A register with a 5 and leaves the B register unchanged. None of the other registers are affected by this instruction. The flags are not affected.

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

The functions of this instruction are reserved for future use.

HALT — HALT

Op Code: 000, 001 or 377
Operation: The processor halts

These instructions PUSH the current system status, flags, and program counter (address following the HALT) onto the stack. They clear the USER, SIE, and MIE status and jump to 170030 in system ROM. These are privileged instructions, and the flags are not affected.

ENABLE INTERRUPTS — EI

Op Code: 050

This instruction sets the System Interrupt Enable and Millisecond Interrupt Enable bits in the status register. Interrupts cannot occur until one instruction after EI is executed. This is a privileged instruction, and the flags are not affected.

DISABLE INTERRUPTS — DI

Op Code: 040

This instruction clears the System Interrupt Enable and Millisecond Interrupt Enable bits in the status register, disabling interrupts immediately. This is a privileged instruction, and the flags are not affected.

CONDITION CODE SAVE — CCS,CCS(r)

Op Code: 042, r 042

This instruction loads the register (r) with a value such that if the value is added to itself using the AD(r) 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 = 0
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.

BASE REGISTER LOAD — BRL,BRL(r)

Op Code: 072, r 072

This instruction loads the base register from the specified register. If no register is specified, the A register is used. This is a privileged instruction, and the flags are not affected.

BASE REGISTER SAVE — BRS

Op Code: 062 163

This instruction loads the A register from the base register. This is a privileged instruction.

LOAD SYSTEM CONTROL — LKA

Op Code: 163

This instruction copies the contents of the A register into the system control register. Bits 3 through 7 are loaded only if bit 2 of the A register is equal to 1; otherwise, only bits 0 and 1 are loaded. This is a privileged instruction; the flags are not affected.

LOAD A FROM SYSTEM CONTROL — LAK

Op Code: 111 163

This instruction loads the A register from the system control register. Bit 2, the control register load enable bit, will always be set to a one when read back. This is a privileged instruction. The flags are not affected.

SAVE SYSTEM STATUS — LAS

Op Code: 165

This instruction copies the contents of the system status register into the specified register. This is a privileged instruction. The flags are not affected.

STACK MOVE — STKMV

Register Pair	Op Code
BC	062 065
DE	174 065
HL	176 065

This instruction swaps the current value of the stack pointer and offset with the contents of the specified register pair. This allows the location of the old stack area to be read and a new stack area substituted. This is a privileged instruction. The flags are not affected.

SECTOR TABLE LOAD STL

Op Code: 077

This instruction loads up to 16 entries of the sector table from a byte string pointed to by HL. The entries are loaded from entry 0 to entry 15. The memory address is incremented after each byte is transferred. The sector entry is output on A15-A12, and the entry location is incremented with each transfer. This is a privileged instruction.

Entry: HL = location of first byte.
C = number of entries to be loaded (0 to 16; 0 implies no loads to be performed).

Exit: No registers are changed.

The flags are not affected.

SECTOR TABLE
LOAD STARTING AT OFFSET STLO(r)

Mnemonic	Op Code
STLO A	022 077
STLO B	111 077
STLO C	062 077
STLO D	113 077
STLO E	174 077

This instruction loads up to 16 entries of the sector table from a byte string pointed to by the HL. The offset is contained in the most significant four bits of (r). It is output on A15-A12 and incremented after each byte is transferred. The memory address is also incremented after each transfer. This is a privileged instruction.

Entry: HL = location of first byte in a table of up to 16 sector table entries to be loaded.
C = number of entries to be loaded (0 though 16 0 implies no load takes place).
(r) = starting sector table entry (upper four bits 0 through 15; the lower four bits of (r) can be any value).

Exit: Sector table loaded.

No registers are changed. The flags are not affected.

NOTE: For the special case of STLOC where the C register contains both the offset and number of entries to load, a load count of 16 is not possible.

SECTOR TABLE READ STR

Mnemonic	Op Code
STR A	022 055
STR C	062 055
STR D	113 055
STR E	174 055
STR H	115 055
STR L	176 055
STR X	117 055

This instruction loads the specified register from the sector table entry contained in the high order four bits of the B register. The sector table selection is made from control register bits 1 and 0 (S2, S1 respectively). The flags are not affected.

SELECT ALPHA MODE ALPHA

Op Code: 030

This instruction selects the alpha mode registers and control flip-flops. This is a privileged instruction. Flags are switched to alpha mode.

SELECT BETA MODE BETA

Op Code: 020

This instruction selects the beta mode registers and control flip-flops. This is a privileged instruction. The flags are switched to beta mode.

5.11.9 Category 8—String Operations

BLOCK TRANSFER
OR BLOCK TRANSFER REVERSE BT,BTR

Op Codes: 021 for BT, 111 021 for BTR

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 if a character that is equal to the 2's complement of the B register is stored in the destination field (stops after the matching character is moved). These instructions are interruptible upon completion of each transfer.

Entry: HL = location of first source 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.

Exit: HL = location past last source byte.
DE = location past last destination byte.
A = entry value.
B = entry value.
C = zero or count before termination character found.

The flags are indeterminate.

BLOCK CONVERT BCV

Op Code: 062 021

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. DE is incremented after each transfer. The translation is performed by adding the source byte to the value of the L register (the addition is performed in the A register) and using the resulting value (H | L + s) as a memory address to the translate string. The translated value is compared to the B register (if B does not equal zero), and early termination occurs if the byte matches in 2's complement form. This instruction is interruptible upon completion of each transfer.

Entry: HL = location of the translate table (must not cross a page boundary).
DE = location of the first byte to be translated.
C = number of bytes to move.
B = 2's complement of terminating character if not 0.
A = no entry value used.

Exit: HL = undefined.
DE = location of last source byte translated plus one.
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. Set A to the sum 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. (Add - 1.)*
9. Go to step 1 if the result is non-zero.

*A decrement operation is actually an add of -1.

The flags are indeterminate.

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

Op Code: 011 for BFAC, 031 for BFSB

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 to by DE. The fields may be 1 through 16 bytes in length as determined by the lower four bits of the C register. Each string is stored in memory with the MSB at the smallest address to the LSB at the largest address. HL and DE are decremented after each transfer. This instruction is interruptible after the addition or subtraction of each byte takes place.

Entry: HL = location of LSB of the operand field.
DE = location of LSB of the accumulator field.
C = number of bytes in the field (1 through 16; 0 implies 16).

Exit: HL = address of operand MSB minus one.
DE = address of accumulator MSB minus one.
C = modulo 16 zero.

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

Carry is carry or borrow from the last operation. Other flags are indeterminate.

DECIMAL FIELD ADD WITH CARRY DFAC

Op Code: 111 041

A zoned BCD number is one that contains two distinct sections within a single byte. The lower four bits are the actual BCD number, while the upper four bits are a code unique to that given system of numbers. Thus, compatible zoned BCD numbers are those with matching zone sections (bits 4-7). This instruction takes a field of zoned BCD digits pointed to by the HL register pair, adds it to the field of zoned BCD digits pointed to by the DE register pair, and stores it in the field pointed to by the DE pair. HL and DE are decremented after each transfer. The zone bits are set to those contained in the B register. Each string is stored in memory with the MSB at the smallest address and the LSB at the largest address. The fields may be 1 to 16 bytes in length, as determined by the C register. This instruction is interruptible after each byte addition takes place.

Entry: HL = location of LSB of the operand field.
DE = location of LSB of accumulator field.
B = zone information (bits 0-3 must be 0, bits 4-7 must be other than zero).
C = number of bytes in the field (1-16, 0 implies 16).

Exit: HL = address of operand MSB minus one.
DE = address of accumulator MSB minus one.
B = entry value.
C = modulo 16 zero.

Algorithm: 1. Load the implicit register from C.
2. Get the byte pointed to by HL.
3. Add it with carry to the byte pointed to by DE.
4. Strip away the zone bits.
5. Clear the Carry and go to step 7 if the result is less than 10.
6. Subtract 10 from the result and set the Carry.
7. Set the zoning bits.
8. Store the result where DE points.
9. Decrement HL and DE by one.
10. Decrement the implicit register by one.
11. Go to step 2 if the implicit register is not zero.

Carry is carry from the last operation. Other flags are indeterminate.

DECIMAL FIELD SUBTRACT WITH BORROW DFSB

Op Code: 062 041

This instruction takes a field of zoned BCD numbers pointed to by the HL register pair, subtracts it from a field of zoned BCD numbers pointed to by the DE register pair, and stores the results in the field pointed to by the DE register pair. HL and DE are decremented after each transfer. The zone bits of the two fields must be identical. The zone bits of the result are set to those contained in the B register. The fields may be from 1 to 16 bytes in length, as determined by the C register. Each string is stored in memory with the most significant byte at the smallest address and the least significant byte at the largest address. This instruction is interruptible after each subtraction takes place.

Entry: HL = location of LSB of operand field.
DE = location of LSB of accumulator field.
B = zone information (bits 0-3 must be 0, bits 4-7 must be other than zero).
C = number of bytes in the field (1 to 16, 0 implies 16).

Exit: HL = address of operand MSB minus one.
DE = address of accumulator MSB minus one.
B = entry value.
C = modulo 16 zero.

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

Carry is borrow from the last operation. All other flags are indeterminate.

BLOCK COMPARE BCP

Op Code: 041

This instruction matches two strings of bytes from the MSB to the LSB until either a mismatch is found or the specified maximum number of bytes has been scanned. HL and DE are incremented after each byte. This instruction is interruptible after each byte is compared.

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

Exit: IF A MISMATCH WAS FOUND:
HL = location of the mismatch plus one in the subtracting field.
DE = location of the mismatch plus one 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. Zero is clear.
IF ALL BYTES MATCHED:
HL = location of the last byte plus one in the subtracting field.
DE = location of the last byte plus one in the field subtracted from.
C = zero.

Algorithm: 1. Get the byte pointed to by HL.
2. Subtract it from the byte pointed to by DE.
3. Increment DE and HL.
4. Exit if the Zero condition is false.
5. Decrement C. (Add -1.)
6. Go to Step 1 if C is not equal to zero.
7. Exit with the Zero condition true.

The Zero flag is set. Other flags reflect the result of the last significant operation.

BINARY FIELD SHIFT LEFT BFSL

Op Code: 075

This instruction shifts a field of bytes in memory left one bit position as if all of the bytes made up one continuous word. HL is decremented after each byte shifts. This instruction is interruptible after each byte is processed.

Entry: HL = location of LSB of the field.
C = the field width (1 through 16; 0 or 16 implies 16).
Carry = bit shifted in on right.

Exit: HL = location of MSB minus one of the field.
C = modulo 16 zero.
A = indeterminate.

Carry bit contains the most significant bit of the MSB. All other flags are indeterminate.

BINARY FIELD SHIFT RIGHT — BFSR

Op Code: 111 075

This instruction shifts a field of bytes in memory right one bit position, as if all of the bytes made up one continuous word. HL is incremented after each byte shifts. This instruction is interruptible after each byte is processed.

Entry: HL = location of MSB of the field.
C = the field width (1 through 16; 0 or 16 implies 16).
Carry = bit shifted in on left.

Exit: HL = location of LSB plus one of the field.
C = modulo 16 zero.
A = indeterminate.
Carry = bit shifted out on the right.

Carry flag contains the least significant bit of the LSB. All other flags are indeterminate.

BINARY FIELD LEFT TO RIGHT OPERATIONS — BFLR(op)

Mnemonic	Op Code
BFLRAD	111 006
BFLRAC	111 016
BFLRSU	111 026
BFLRSB	111 036
BFLRND	111 046
BFLRXR	111 056
BFLROR	111 066

These instructions perform the indicated operation between the field pointed to by the HL pair and the field pointed to by the DE pair, leaving the result in the field pointed to by the DE pair. These instructions increment HL and DE after each byte transfer. This instruction is interruptible after each byte is processed.

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

Exit: HL = location of last byte plus one of operand.
DE = location of last byte plus one of accumulator field.
C = modulo 16 zero.
Carry = carry or borrow out of the operation.

Flags reflect the results of the last operation. All logical operations cause the Carry flag to reset.

6.2.1 Transmitter Logic

The transmitter logic idles in the non-transmit mode (RCV mode). Whenever a character is loaded into the Data Output Register and the Data In line pair is at a mark, the transmit logic will load the transmit shift register and transmit bit counter, thus transmitting the character. When the transmit shift register is loaded, the start, stop, control/data and parity bits are also loaded. Whenever a character is loaded into the Data Output Register, the transmit logic goes into the transmit mode and transmits the character. After the character is transmitted, the transmit logic reverts to the receive mode. However, during DMA output cycles, the transmit logic goes into the transmit mode as the first word is loaded into the Data Output register and remains in the transmit mode until the last character of the DMA output cycle has been transmitted.

6.2.2 Receiver Logic

The receiver logic assembles characters transmitted by the addressed peripheral, checks for parity errors and then latches the character into the Data Input register. The receiver logic, upon receiving a start bit, enables the gated oscillator and sets the Data Out line pair to a space if a second character cannot be handled by the hardware immediately following the first. In the non-DMA mode, only single characters can be handled, while in the DMA mode two characters are needed to make up a word for a DMA transfer. Thus in the DMA mode, the Data Out line pair remains at a mark until the start bit of the second character is detected. Once the DMA transfer has written the word from the Data In register into memory, the Data Out line pair will be driven to a mark allowing another byte or word of data, if any, to be transmitted.

6.3 Microbus Interface Module

The Microbus Interface Module (MIFM) provides the electronic hardware necessary to connect the internal common bus to the external microbus. The MIFM contains a programmable polling feature that provides enhanced system operational capability such as processor interrupt. The block data transfer rate is 500 kilobytes per second, and the maximum block length is 256 bytes. A block diagram of the MIFM is shown in Figure 6-2.

For detailed information on ports, registers, and programming, see the respective product specifications for the 9310 Disk Drive (Document No. 60876) and the 1401 Diskette Drive (Document No. 61031).

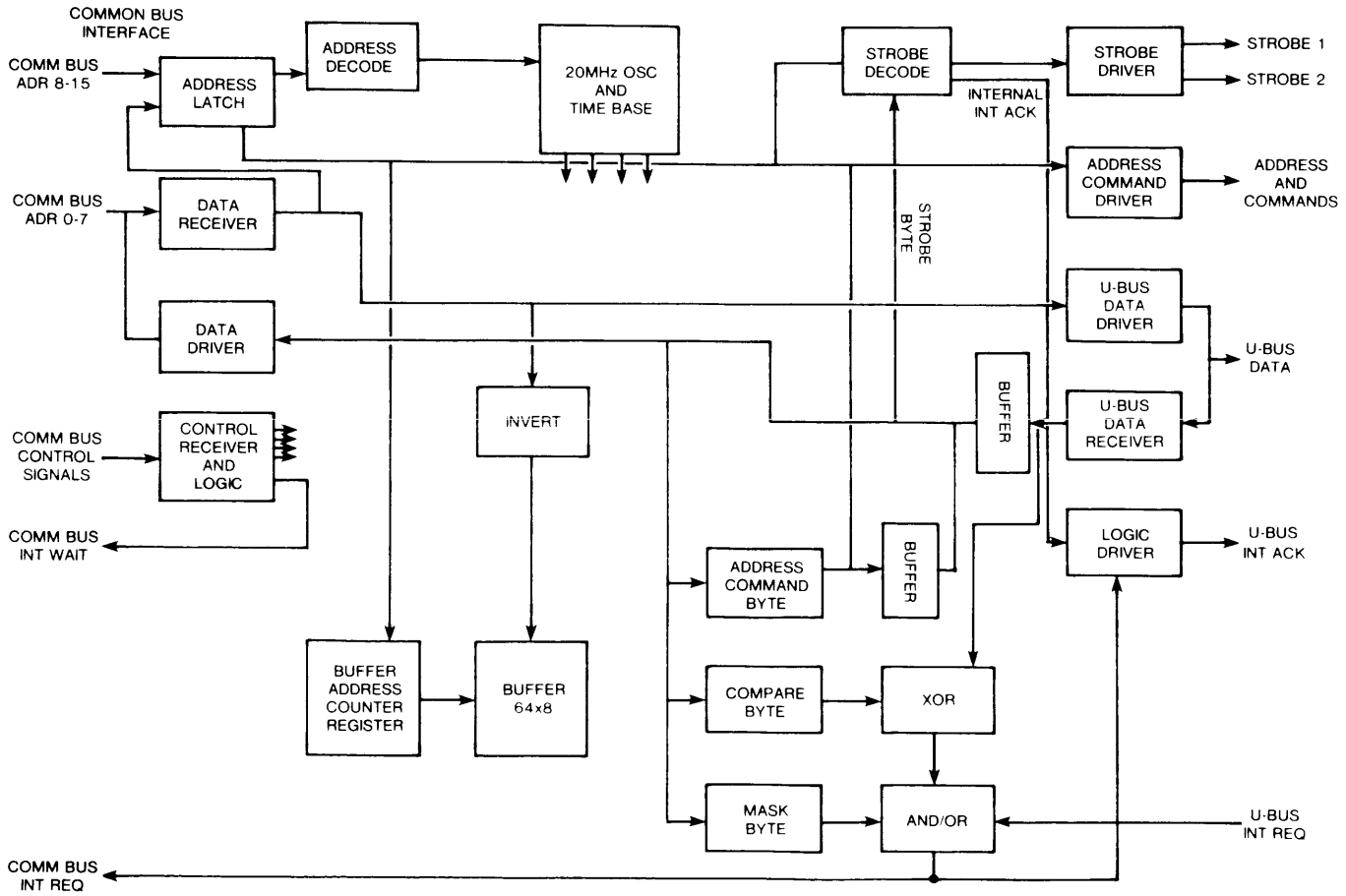


Figure 6-2: Microbus Interface Board

6.3.1 Common Bus Interface

The common bus provides internal synchronous interface between the CPU, memory, and peripherals. The MIFM uses the vectored interrupt capability on the common bus. All peripheral devices attached to the MIFM have their microbus interrupt lines wired to one common bus line, CBIREQ0. The MIFM decodes its own selectable address and enables other common bus signals to perform their functions.

The interrupt sequence provides a means whereby a peripheral device may asynchronously gain the attention of the processor. The interrupt originates either in the peripheral device or in the MIFM status polling logic. The processor handles the interrupt scheduling in its master interrupt controller. It responds to the interrupt with a specially coded read instruction to determine the nature of the interrupt and to identify the appropriate interrupt handling routine.

6.3.2 Microbus Interface

The microbus is an external parallel interface bus that connects the peripheral to the MIFM. The microbus is composed of an eight-bit command and address bus, an eight-bit bidirectional data bus, two command strobes, an interrupt acknowledge strobe, an interrupt request line, and a +5 volt power indication.

During an output cycle, the MIFM loads the command/address register and data lines with the appropriate information and enables the bus drivers. The peripheral device completes the specified command and is ready to accept additional commands every two microseconds. Commands requiring longer than this are associated with a "busy" status bit.

During an input cycle, the MIFM loads the command/address register with the appropriate information and enables the bus drivers. The peripheral device disables its bus drivers after detecting the trailing edge of the transfer strobe.

An interrupt cycle is initiated whenever a peripheral device pulls the microbus IREQ line low. The processor may respond at any time to the IREQ by initiating the microbus IACK strobe. The IREQ line is independent of all other activity on the microbus.

6.3.3 Data Transfer

All data transfer between the processor and the MIFM is under software control. Six I/O instructions are provided for data transfer: DMPIN, DMPOUT, BLKIN, BLKOUT, UBIN, and UBOUT. (These instructions are explained fully in Section 5.11.7 of this manual.) The MIFM requires 1K words of I/O address space. Allocation is determined by jumpers or switches in the MIFM. This provides addressing flexibility and allows multiple MIFM modules to reside on the same common bus.

6.3.4 Polling Function

The MIFM includes timing, control, and storage logic to provide a programmable peripheral device polling function. This function permits peripheral devices that do not support interrupts to be used in a real-time environment and enhances the operation of those that do support interrupts.

The MIFM polling function is organized around a 64-byte random access memory (RAM) that provides the needed storage for address/command, strobe, compare, and mask bytes. Polling, when enabled, is initiated by the start of a processor memory cycle and is controlled by an internal time base generator. Polling logic handles one interrupt at a time.

Scanning from the buffer is done in a sequential fashion; however, loading of the buffer is under software control and the order of device addresses is arbitrary. Bytes in the buffer may be written/read individually or on a block basis. The MIFM buffer must be initialized before polling is enabled.

The polling sequence generates a microbus operation of 600 ns (400 ns plus 100 ns for both setup and hold). Timing is initiated when the start of a memory cycle is detected on the common bus and polling is enabled. The sequence of events after initiation is controlled by the time base generator in the MIFM to ensure completion of the polling cycle before the start of a processor I/O cycle or the next memory cycle.

7.3 Memory

The MPCA has the capability of addressing 2K X 8 of static RAM and 8K x 8 of ROM. The MPCA microprocessor accesses ROM and RAM by performing Z80 memory cycles.

RAM resides at location 020000-023777 and is addressed by the CPU at I/O 070000-0737777. It provides for the individual Receive and Transmit FIFOs, translation tables, and associated staging buffers for each port. The RAM also contains all pointers, status bytes, command bytes, interrupt information bytes, and the MPCA controller stack. Both the CPU and the Z80 have access to the RAM.

The ROM resides at location 00000-07777. Approximately 2K of ROM is used for program execution by the MPCA controller. The remaining 2K segment is used for on-board diagnostics. The CPU does not have access to ROM.

7.4 USART

The MPCA provides interface between the CPU and each of the serial RS-232C channels via eight-bit data buffers. Each serial port is driven by its own Universal Synchronous Receiver/Transmitter (USART) integrated circuit that converts the eight-bit bytes to and from the CPU to serial data. On the MPCA side, the USARTs feed 64-byte FIFO buffers which in turn feed 32-byte staging buffers. In the translaton mode, the receive FIFOs are reduced to 32 bytes to make room for the translation table.

7.5 Baud Rate Generators

Programming of baud rates is done by the Z80 upon request by the CPU. Transmit baud rate clocks are generated by external chip baud rate generators. Receive baud rates are generated by the USART internal baud rate generator. Any programming difference between the two baud rate generators is transparent to the CPU.

7.6 CPU Interface

The CPU has access to the MPCA through Base Page I/O addressing for control related data only, and through extended I/O addressing for character transfers as well as additional control commands.

Addressing of the MPCA at the board level is accomplished by the CPU writing out a board select code prior to initial data and control transfers. This address is common on all MPCA cards in the system, and each card will sample and compare this code to see if it has been selected. Once a board has been selected, it will remain so until another MPCA board select code is written out. Each MPCA card shares the same interrupt request line.

Commands for the MPCA are written by the CPU to on-board RAM space in extended I/O. After the command data has been written, the CPU will issue a maskable command service interrupt to the MPCA controller by writing to an address in base page I/O. Command information can take up to four bytes in the RAM command section depending on what instruction is to be implemented. All instructions will use byte 0 of the command section in RAM.

7.7 Interrupt Structure

Interrupt-related information is stored primarily in the on-board RAM in the CPU's extended I/O space. The CPU polls each MPCA in the processor to determine which boards (and which ports on each board) are requesting service. Once this has been determined, the CPU services the interrupt and then writes the Interrupt Service command. This causes the interrupt request bits of the ports serviced to be reset. If an interrupt is pending for a port, the Z80 will wait until the Interrupt Service command has been written before filling receive staging buffers or emptying transmit staging buffers. If the interrupt request line is not being asserted for a port on the same MPCA card (in the case of multiple MPCAs), then the Z80 will write the port service code and assert the interrupt request line if staging buffer service is needed. If the interrupt request line is high, the Z80 will write the port service code to the address set aside in on-board RAM.

7.8 Firmware

Operating firmware for the MPCA is contained in on-board ROM. The firmware is responsible for initialization, diagnostics, polling loop, and command execution.

The initialization sequence is invoked by a common bus POR, or by a restart or reset from the CPU. This routine brings the USARTs to a known state and initializes the on-board RAM.

The polling loop is the main operating loop in the firmware. It runs continuously except when interrupted by the CPU to execute special commands. (Command interrupts are disabled between individual port polling sequences.) The loop polls each port in turn for needed USART, FIFO, and staging buffer service.

7.9 Diagnostics

MPCA diagnostic code resides in on-board ROM. It allows the MPCA microprocessor to test board components. Internal testing is done for the MPCA RAM, ROM, USARTs, various hardware registers, and for the microprocessor itself. Hardware failures are reported to the CPU, and the polling loop is entered if possible.

PART 8
MULTIFUNCTION COMMUNICATIONS ADAPTER (MFCA)

8.1 General

The Multifunction Communications Adapter (MFCA) is a printed circuit board that resides in the 8600 processor's internal card cage. The MFCA provides a means of bi-directional information transfer between the processor and an RS-232C compatible communications channel with reverse channel. The MFCA may be connected to an external modem or an RS-366 compatible Automatic Calling Unit (ACU). The processor can be equipped with a maximum of two MFCAs. The MFCA includes a microprocessor; a serial input/output channel, a counter/timer circuit, and a local memory with parity. A block diagram of the MFCA is shown if Figure 8-1.

The 8600 MFCA includes the following features:

- Synchronous/asynchronous operation, full or half duplex.
- BISYNC, SDLC, HDLC, ADCCP, and GENSYNC protocols.
- Programmable baud rates (110 to 56K).
- Programmable internal or external clock.
- Programmable NRZ or NRZI data.
- Programmable sync characters and stop bits.
- Hardware CRC generation/checking.

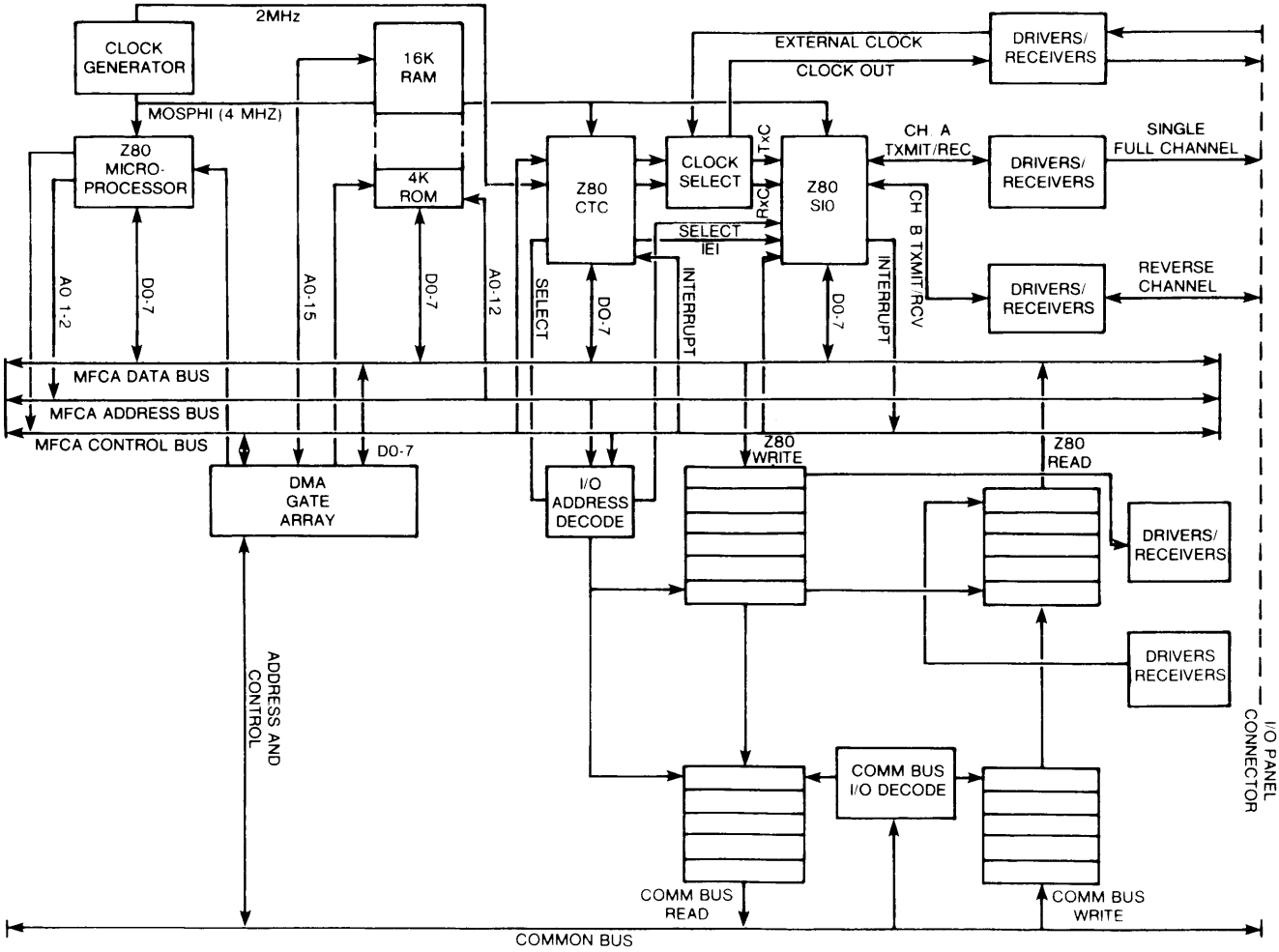


Figure 8-1: Multifunction Communications Adapter Board

8.2 Microprocessor

The MFCA contains an on-board Z80A microprocessor that runs at 4 MHz and uses its own internal address and data buses for communication with the other components on the board. The microprocessor derives its clock from a crystal controlled clock source contained on the MFCA. It supports maskable and non-maskable interrupts along with DMA (Direct Memory Access) and has its own internal refresh logic for use with dynamic RAMs. (Note: In the following sections, the MFCA microprocessor is referred to as the Z80 and the 8600 central processor is referred to as the CPU.)

8.3 Serial Interface

The MFCA contains a Z80A SIO (Serial Input/Output) and supports one full communication channel and one reverse channel. In addition to the communication channel, the SIO provides an RS-366 compatible ACU. The drivers and receivers are located on the MFCA I/O panel. One version of the I/O panel is available: it supports the RS-232C channel.

8.4 Counter/Timer Circuit

Baud rates are program selected through Z80 I/O commands and are provided by a Z80A Counter/Timer Circuit (CTC). Separate programmable transmit and receive clocks are provided. The CTC derives its clock from the 4 MHz source used to drive the Z80 and the internal timing of the CTC. The CTC also acts as a real-time clock and an interrupt controller.

8.5 Memory

The memory space is organized as 16K X 9 RAM, 4K X 8 ROM, and 16K DMA addressing space. The RAM provides storage for the code that is down-line loaded from the CPU memory. The RAM also provides buffer space for received and transmitted data. Dynamic RAM supports odd parity and begins at address 040000 in Z80 memory space.

The ROM resides at location 000000-07777 and contains system firmware and self-test diagnostics. The firmware initializes the MFCA and facilitates the downloading process of CPU resident code into the MFCA RAM. A checksum, stored in the last byte of the ROM, is used to check ROM integrity.

The MFCA uses memory-mapped DMA for rapid transfer of data to and from CPU main memory. The DMA is transparent to the Z80 and begins at address 0140000.

Access to the MFCA RAM is provided only to the Z80 through the internal address and data buses. MFCA operational code is transferred into MFCA RAM from main memory through a combination of I/O registers, MFCA firmware, and DMA. The I/O registers appear in the I/O address space of both the Z80 and the CPU and are loaded by the CPU with the download command and registers. The firmware monitors these registers, accepts the command parameters, and then begins loading the MFCA RAM by accessing CPU memory through DMA.

The DMA controller consists of the Z80 in combination with a custom gate array common bus state controller. When accessed as memory, the DMA controller asserts wait to the Z80 and then gains control of the common bus as a bus master using the priority transfer scheme. As soon as the needed byte is transferred to or from CPU main memory, the Z80 is taken out of wait and continues processing.

8.6 Interrupt Structure

The MFCA may be operated in polled or interrupt mode. The Z80 itself supports both maskable and non-maskable interrupts from various sources including the I/O interface to the CPU, CTC, and SIO. The interrupt to the CPU is issued via the common bus interrupt signal level 6 (CBIREQ6). Each MFCA in the 8600 (maximum of two) is tied in common to this interrupt level, and conforms to the common bus timings. Interrupts to the CPU through thes common bus are used for CPU-to-MFCA communication. Internal interrupts to the Z80 are used for specialized functions of the Z80 and for communication between the Z80 and special devices such as the SIO.

8.7 Firmware

The MFCA ROM-resident firmware drives the I/O interface and provides initialization, downloading, execution, and diagnostic commands as well as various essential routines such as reset, power-up, and abort. It also provides a software protocol necessary to allow the passing of these commands and data back and forth between the MFCA and the CPU.

8.8 Diagnostics

The MFCA includes on-board diagnostic capabilities implemented with hardware and firmware.

The firmware diagnostics provide self-test routines to check out every major hardware block in the MFCA including a check of the ROM by testing the checksum loaded in the last byte of the ROM.

PART 9
SYSTEM FIRMWARE

9.1 Introduction

The 8600 processor has 4K of system ROM that resides on the CP/RIM board. The system ROM is addressed only in System Mode at addresses 0170000-0177777. The major functions of system ROM are:

- Initialization
- Diagnostics
- System RAM Vectors
- Initial Program Loader
- Keyboard/Display Routines
- Debug

These modules are presented in the following sections of Part 9.

9.2 Initialization

System initialization consists of power-on reset (POR) and restart. On power-up, the CP transfers control to the POR routine by causing a jump to the POR Trap Vector. The routine then performs a series of functions to bring the processor into an operational mode. Upon completion, the POR sequence passes control to the restart routine, which performs a save of the system state and determines whether a debug or bootload request has occurred. Restart can also be initiated from the keyboard or from a software routine.

9.3 Diagnostics

Diagnostics in system ROM are used to detect, isolate, and recover from faults in the 8600. Fault testing is done upon POR. The diagnostic routine checks the processor, sector tables, ROM, RAM, RAM buffers, RIM interface, KDS, PIO, and MIFM.

Once the POR diagnostics have successfully run to completion, control is transferred to the operating firmware to complete initialization. If any component is found to be faulty, a status message is issued with the sign-on message. If an operable system cannot be configured, a status message will appear on the screen and control will be transferred to debug.

9.4 System RAM Vectors

The system RAM vectors shown below may be trapped by software. If not, they transfer control to system firmware default routines. With the exception of the One Millisecond Interrupt, these vectors are non-maskable.

Memory Address	Vector Type	Default Action
0167400	Memory Parity Error	*E1 MEMORY PARITY ERROR*
0167406	Input Parity Error	*E2 INPUT PARITY ERROR*
0167414	Output Parity Error	*E3 OUTPUT PARITY ERROR*
0167422	Write Protect Violation	*E4 WRITE PROTECT ERROR*
0167430	Access Protect Violation	*E5 ACCESS PROTECT ERROR*
0167436	Privileged Instruction Violation	*E6 INSTRUCTION ERROR*
0167444	One Millisecond Interrupt	POPs stack and jumps to zero
0167452	System Call	*E7 INSTRUCTION ERROR*
0167460	Breakpoint	Saves status and enters debug
0167466	Unassigned Instruction	*E8 INSTRUCTION ERROR*
0167474	Sector Table Parity Error	*E9 SECTOR PARITY ERROR*
0167502	Power Failure Trap	*POWER FAIL*
0167510	Halt	*HALT*

Figure 9-1: System RAM Vectors

9.5 IPL Block Loader

The loader searches for the presence of operational devices from which to perform an Initial Program Load (IPL). An IPL can be performed from any compatible device attached to the PIO, MIFM, or RIM. The loader searches peripheral storage devices for a valid IPL block. The search is performed in the following order:

A—Tapes (9301)
B—Disks (9301)
C—Drive 1 of 9302 extended disks
D—Drive 2 of 9302 extended disks
E—Drive 3 of 9302 extended disks
F—Drive 4 of 9302 extended disks
G—Removable disks (9320)
H—Drive 0 of 1401 and 1403 diskettes
J—Drive 1 of 1401 and 1403 diskettes

If a tape, disk, or diskette is not in place, the corresponding drive is skipped.

When a functioning device is found on-line with media in place, the search sequence stops and a block load from the on-line device is performed. A check of the loaded data is performed to determine its validity. If the IPL block is invalid, the search sequence continues. This is not a wrap-around search, and control is transferred to the RIM loader routine if the search sequence completes without finding a valid IPL block. Peripheral device search may be passed by holding the KBD and CTRL keys down when initializing Restart. The loader is initiated via external entry point 0170115.

9.6 Keyboard/Display Routines

The KDS module contains several subroutines that are used by system ROM to control the keyboard and display functions of the 8600 processor. The following routines have external entry points.

Entry Point	Routine Name	Description
0170070	$86KEYIN	Input Translated Keyboard Entry
0170073	$86KDSII	Initialize the Keyboard and Display
0170076	$86CHRLD	Load the Display Character Font
0170101	$86DSPII	Initialize the Display
0170104	$86DSPLY	Display the Character String Pointed to by HL
0170107	$86CRSLD	Blink the Cursor at Screen Coordinates in DE
0170112	$86CLOC	Calculate the Display Buffer Address
0170115	$86RSTRT	Reboot the Machine
0170120	$86DOSKY	Input Untranslated Keyboard Entry

9.6.1 $86KEYIN

Entry point: 0170070

Registers: Entry: None.
Exit: A has keyboard character. H, L, and BC are scratched. All others are preserved.

Flags: The zero flag is set if no character is available. The zero flag is not set if the character is presented. All others are indeterminate.

Stack: Three levels are used.

Exit: Return to calling routine.

This subroutine will obtain a character from the keyboard. Depending upon the entry point used, the character will be translated or untranslated. If the translated entry point is used, the A register will contain the ASCII character on exit. Otherwise, the A register will contain the character as entered.

9.6.2 $86KDSII

Entry point: 0170073

Registers: A, B, C, D, E, H, and L are scratched at exit.

Stack: Three levels are used.

Exit: Return to the calling routine.

This subroutine initializes the keyboard and display to a blank screen, cursor off, and the abbreviated, default character font.

9.6.3 $86CHRLD

Entry point: 0170076

Registers: Entry: HL points to font load table.
Exit: All registers are scratched.

Stack: Three levels are used.

Exit: Return to calling routine.

This subroutine loads the character font RAM from a character font data string pointed to by HL. The screen is blanked during the font load operation and is restored upon completion.

Format of font data string is:

ASCII, 12 bytes of font
ASCII, 12 bytes of font
.
.
.
ASCII, 12 bytes of font

9.6.4 $86DSPII

Entry point: 0170101

Registers: Entry: no requirement.
Exit: A, D, E, H, and L are scratched.

Stack: Two levels are used.

Exit: Return to calling routine.

This subroutine initializes all display line pointers, sets the screen to normal mode with block cursor, and blanks the entire screen.

9.6.5 $86DSPLY

Entry Point: 0170104

Registers: Entry: DE points to the beginning cursor location [D = Horizontal (0 to 79), E = Vertical (-12 to 11.) HL points to the string to be displayed. B contains the display attributes.

Exit: DE points to the ending cursor position + 1. HL points to ending string position +1. All others are scratched.

Stack: Five levels are used.

Exit: Return to calling routine.

The DISPLAY subroutine displays, on the screen, the character string pointed to by HL. The screen location for the start of the display may be in DE, or it may be embedded in the string before the first displayable character. The display options are available by setting their corresponding bits in the B register high:

Bit 0	Underline
Bit 1	Two Level
Bit 2	Blink
Bits 3-5	Not used
Bit 6	Non-destructive blanks
Bit 7	Inverse video

In addition to characters to be displayed, the character string may contain the following embedded control sequences. All codes are represented in octal.

$NS AAA AAA $NS (0203) = New String Address follows, where AAA AAA is the new address.

$H HHH $H (011) = New horizontal cursor position follows, where HHH is the new horizontal position.

$V VVV $V (013) = New vertical cursor position follows, where VVV is the new vertical position.

$RU $RU (023) = Roll screen up one line.

$RD $RD (024) = Roll screen down one line.

$EEOL $EEOL (022) = Erase to End of Line.

$EL $EL (015) = End of Line. Carriage return and line feed with roll-up one line if already on the bottom line and ends the string.

$EEOF $EEOF (021) = Erase to End of Frame. Erase to end of this line and all screen lines below this one.

$BP $BP (007) = Beep.

$F $F (033) = Force display of next character.

$CK $CK (0207) = Click.

$HA $HA (0211) = Horizontal adjustment follows.

$VA $VA (0213) = Vertical adjustment follows.

$HU $HU (0223) = Home Up. The cursor returns to the home position, the upper left-hand corner.

$HD $HD (0224) = Home Down. The cursor returns to the home down position, the lower left-hand corner.

$O $O (0233) = New Options follow. The option bits are described above.

$ES $ES (003) = End of string.

Each display string must be terminated by a $ES character (003) or by a $EL (015).

9.6.6 $86CRSLD

Entry Point: 0170107

Registers: Entry: DE cursor coordinates (see 9.6.5).

Exit: DE unchanged.
A, B, and C are scratched.
X, H, and L are preserved.

Stack: Three levels are used.

Exit: Return to calling routine.

This subroutine positions the cursor to the screen coordinates passed to it in DE.

The $86CRSLD subroutine positions and displays the cursor to the screen coordinates contained in DE. It is invoked by several firmware routines and may be invoked by a software call to $86CRSLD. Loading DE to an invalid cursor location, such as -1, will turn the cursor off.

9.6.7 $86CLOC

Entry point: 0170112

Registers: Entry: DE contains cursor coordinates (see Section 9.6.5).

Exit: DE contains the screen buffer address. C,A is scratched. All others are preserved.

Stack: One level is used.

Exit: Return to calling routine.

This subroutine converts cursor coordinates to the corresponding screen buffer address.

9.6.8 $86RSTRT

Entry Point: 0170115

Registers: Entry: None.

Exit: None.

This subroutine reboots the system when called externally.

9.6.9 $86DOSKY

Entry point: 0170120

Registers: Entry: None.
Exit: A has keyboard character. H, L, and C are scratched. All others are preserved.

Flags: The zero flag is set if no character is available. The zero flag is not set if the character is presented. All others are indeterminate.

Stack: Three levels are used.

Exit: Return to calling routine

This subroutine will obtain a character from the keyboard. Depending upon the entry point used, the character will be translated or untranslated. If the translated entry point is used, the A register will contain the ASCII character on exit. Otherwise, the A register will contain the character as entered.

9.7 Debug

The immediate accessibility of debug creates a flexible debugging interface between user and machine. The 8600 debug routines are similar in function to the 5500/6600 debug routines; however, some commands have been changed, added, or deleted.

9.7.1 Entry to Debug

There are six methods of entry to debug:

1. Manual depression of the DSP, CTRL, and INT keys, followed by a release of the CTRL or INT key.

2. Execution of a dynamic breakpoint set through debug.

3. Execution of a breakpoint instruction embedded in the user program.

4. At the completion of a firmware interrupt trap routine.

5. Execution of a return instruction following a debug call command.

6. As a consequence of an irrecoverable error during diagnostic execution.

9.7.2 Saving the Machine State

Upon entry, debug saves the active register set, the condition flags, the processor control register, the KDS screen control/interrupt byte, and the base register.

When entry to debug is accomplished by one of the first four methods listed above, the processor will push the flags and the status register onto the stack following the program counter. Debug will retrieve the status and save it. The processor will disable interrupts and switch to system mode.

When a return instruction is executed following a debug call command, the status register is not available and the status register value that was last saved will be used. This method of entry will not work with a user mode routine, because the RET will not force the system out of user mode and user memory space.

In all cases, debug will switch to the debug stack to preserve the system stack. All debug commands use the contents of the registers upon entry to debug.

Upon exit from debug (through the C, E, or J commands), all registers and flags are restored to the values they contained upon entry. If these values were altered, then the altered values are restored. This means that the machine state is totally restored to the condition at debug entry, except for the values that were intentionally altered. The two bottom levels of the stack are scratched.

9.7.3 Display Format

The 8600 debug maintains two current addresses (CURADR), one for memory access and the other for I/O space. The basic debug display consists of five lines in the lower right corner of the screen:

RRRRRR	=	CURADR offset from origin
AAAAAA	=	CURADR absolute address
* NNN	=	The value stored at CURADR
MMMMMM	=	LSB,MSB address formed at CURADR
nnnnnn*	=	Command input line

The top two lines are displayed in inverted video if I/O space is selected. The relative address (RRRRRR) is computed by subtracting the origin bias from CURADR, and is displayed only if the origin bias is non-zero and origin mode is selected. The asterisk (*) represents the display or input of ASCII alpha characters.

9.7.4 Command Syntax

Debug command syntax uses the following notation:

nnn Indicates an optional sequence of octal digits not to exceed the number of n's given.

(nnn)nnn — Indicates an optional sequence of octal digits of either three or six n's in length. If input argument contains more than eight bits of significance, special results will occur. In general, two bytes of memory will be affected by the command: either a register pair or a memory address in LSB,MSB format.

nnnnnn — Indicates a 16-bit argument. If no digits are present, default values will be used.

12345 — Indicates a special command whose accidental execution is inhibited by the requirement that it contain this unique argument.

9.7.5 Input Command List

The complete set of debug commands is presented below. Command definitions are given for unshifted and shifted values.

Unshifted Debug Commands

nnnnnnA — Set current I/O space address to nnnnnn and display that location. If memory space is selected, switch to I/O space with no origin. If nnnnnn is not given, use last current I/O space address.

nnnnnnB — Set a breakpoint at nnnnnn. If nnnnnn is not given, set a breakpoint at CURADR.

nnnnnnC — Call the given or current address. The machine state is restored before control is passed to the subroutine. A RETURN from the called subroutine results in debug being re-entered and the machine state being saved.

nnnnnnD — Decrement CURADR (I/O space or memory space as currently selected) by one or nnnnnn.

nnnnnnE — Continue execution from nnnnnn. If nnnnnn is not given, use top system stack entry. Status is restored to the condition existing prior to debug entry.

nnnnnnI — Increment CURADR (I/O space or memory space as currently selected) by one or nnnnnn.

nnnnnnJ — Jump to given or current address.

nnnK — Set the control register to nnn. An error will result if nnn is not specified. (This command sets the control register, not the saved value.)

(nnn)nnnM — Modify the contents of the current address location (I/O space or memory space as currently selected). If nnnnnn × 0377, modify two bytes, LSB followed by MSB. If no argument is specified, it is treated as if 000 had been entered.

nnnnnnN — Set current memory address to nnnnnn absolute. If I/O space is selected, set memory space with no origin.

nnO — Set origin table pointer and origin mode. If nn is not specified, origin mode will be cleared for memory or I/O space.

nnnnnnP — Load the base register with the upper eight bits of nnnnnn—0100000. (This command loads the base register, not the saved value.)

12345Q — Load the sector table selected by the control register. CURADR points to a table whose first entry contains the following information:

1. The number of entries to be loaded into the sector table is in the four last significant bits.
2. The offset of the first entry into the sector table is in the four most significant bits.

Note: During power-up, system firmware initializes the sector tables so that entries 015 and 016 (octal) point to RAM locations 0150000 through 0167777 used by the debugger and other firmware routines. Changing these entries causes the debugger to enter an undefined state due to the loss of its RAM memory.

R — Perform Alpha/Beta Switch. The saved registers are reloaded, the switch is performed, and the newly active registers are saved.

nnS — Display stack entry nn. If nn is not given, display stack entry 0. An error will result if nn > 037.

nnnT — Display sector table entry nnn where the first n selects the sector table (0-3), and the second and third n's select the entry (0-15). If nnn is not given, sector table 0/entry 0 will be displayed.

12345T — Start memory self-test.

nnnY — Modify or display the saved system status.

Z — Display all registers and register pairs (in the format shown below):

```
FFF AAA BBB CCC DDD EEE HHH LLL XXX
        BBBCCC  DDDEEE  HHHLLL  XXXAAA STKP
```

Shifted Debug Commands

(nnn)nnna	Modify saved Register A value to nnn and display. If no nnn is specified, display the saved value.
nnnb	Modify/Display Register B value.
(nnn)nnnc	Modify/Display Register C value.
nnnd	Modify/Display Register D value.
(nnn)nnne	Modify/Display Register E value.
nnf	Modify/Display condition code flags.
(nnn)nnnh	Modify/Display Register H value.
nnnnnni	Set addressing bias to nnnnnn and select I/O space.
nnnk	Alter the saved control register value to nnn and display. If nnn is not specified, display only.
nnnl	Modify/Display Register L value.
nnnnnno	Set addressing bias to nnnnnn and select memory space.
nnnnnnp	Load the saved base register with the upper eight bits of nnnnnn — 0100000.
p	Display base register (saved value).
nnr	POP stack (nn) times. Error if out of range.
nnnnnns	PUSH nnnnnn onto the stack.
12345t	Go to KDS invokable test.
12345u	Go to invokable PIO loopback test.
nnnx	Modify/Display Register X.
nnnnnn ENTER	Set relative address in memory or I/O space. CURADR = relative address + origin bias. If nnnnnn is not specified, set to current address.
CANCEL	Cancel the command input line.
BACKSPACE	Backspace one space on input line.
(nnn)nnn.	Modify memory and increment CURADR. If nnnnnn < 0400, modify one byte and increment by one. If nnnnnn > 0377, modify two bytes and increment by two. No nnnnnn is treated as 000.
(nnnnnn)^	Same as " . " but save nnnnnn. If no nnnnnn, use the last nnnnnn saved.
#	Clear all active debug set breakpoints.
?	Display processor identification data (processor type, macro ROM version, microcode version).

The processor will beep and return to the command interpreter when an error occurs.

APPENDIX A
ANCILLARY EQUIPMENT

A.1 General

In addition to the primary components presented in previous sections of this manual, the 8600 processor includes a power supply and a motherboard. These two items are discussed in this Appendix.

A.2 Power Supply

The power supply is a single-ended, pulse width modulated, forward converter that operates at 40 KHz. The power supply resides in the processor's internal card cage and operates directly from the rectified AC line using a voltage doubler at 120 VAC or a full wave bridge rectifier at 240 VAC. The 5 volt secondary of the 40 KHz power transformer is rectified, filtered, and applied to the processor. The control loop is closed on the +5 VDC output. The +12 and -12 volt outputs are derived from series regulators that receive rectified and filtered 16 VDC from additional secondary windings. A block diagram of the power supply is given in Figure A-1.

The power supply operates from nominal power of either 120 VAC or 240 VAC, 50/60 Hz. The power input circuit is field changeable for operation at either voltage. The power input circuit consists of two fuses, a line filter, and a power switch. Voltage is taken from the power input circuit to run the processor cooling fan. All output voltages are regulated.

The power supply produces the following voltages at the specified maximum rated current:

+5 VDC at 23 amps
+12 VDC at 4 amps
- 12 VDC at 1 amp

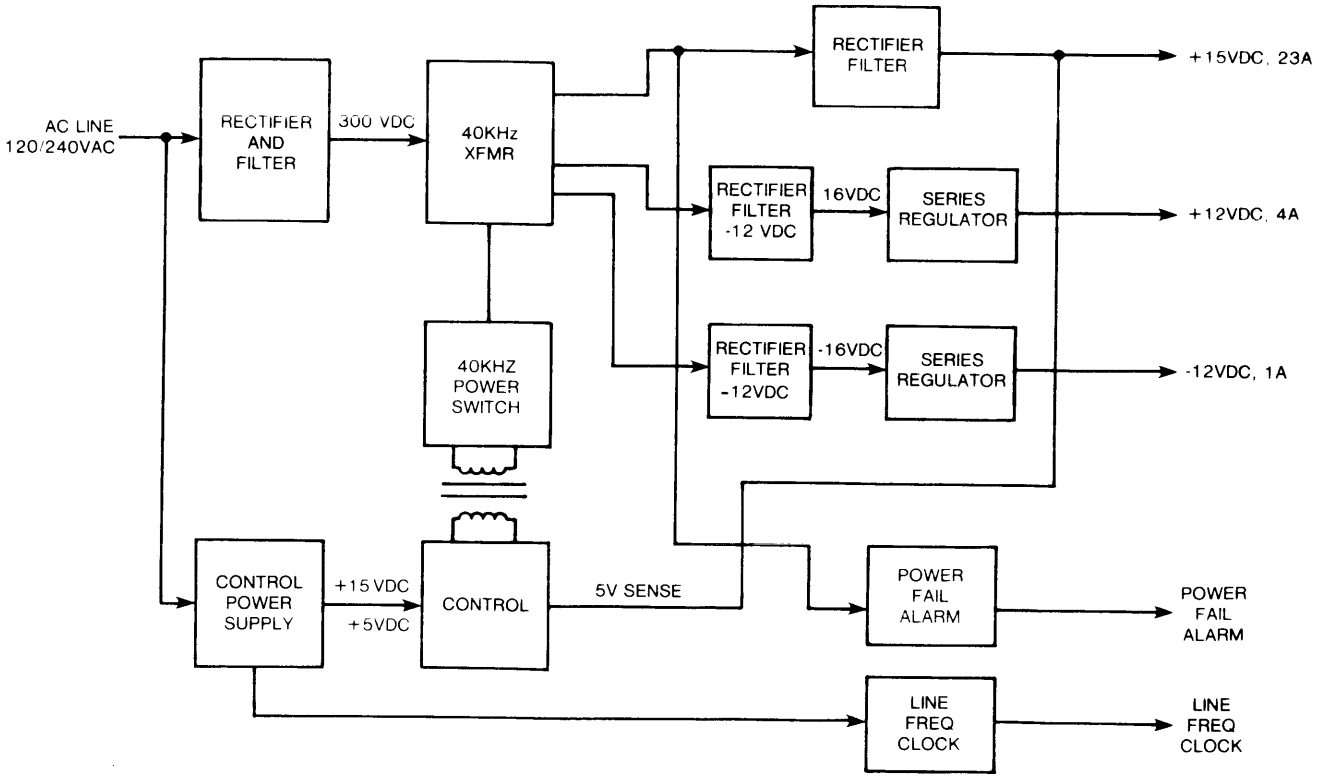


Figure A-1: Power Supply

A.2.1 Protection Circuits

The power supply has five protection circuits that prevent damage from certain conditions. The circuits are explained below.

Output over voltage circuit — ensures that the +5 VDC output voltage does not exceed 7.0 VDC.

Output over current circuit — protects against a continuous overload or short circuit to ground on any output.

Input under voltage circuit — protects against input voltages of less than 85 VAC or less than 170 VAC.

Input over voltage circuit — protects against input voltage surges of 125 percent of nominal line voltage for no longer than ten seconds

Thermal protection circuit — protects against a heatsink temperature of greater than 165 degrees F.

If a circuit detects an undesirable operating condition, it will shut off the power supply and generate a power fail alarm signal.

A.2.2 Power Fail Alarm

The power fail alarm circuit senses the primary AC voltage through the 40 KHz power transformer. When the threshold is detected, an alarm signal is produced. Good power is guaranteed for two milliseconds after the alarm signal is produced.

A.3 Motherboard

The motherboard is a double-sided printed wiring board that contains ten 88-pin edge connectors. One is dedicated to the power supply, while the remaining nine implement the common bus.

A.3.1 Signal Configuration

Motherboard signals are routed parallel with one another on the top side of the PWB. A guard land is placed between those signals that are located closest to one another to reduce cross-talk between adjacent signals. The guard lands are connected to ground via feed-throughs at one point only, and are open-ended at the opposite end of the board.

All signals and power are supplied to the individual cards through means of 88-pin, wave solderable edge connectors. All contact pins have a three-amp rating and are gold plated.

A.3.2 Open Collector Signals

The motherboard supports both open-collector type and three-state type signals. Most of these signals require a passive pull-up resistor. These resistors are supplied by the motherboard by means of 1K ohm SIP packages and are located on one side of the signal land.

A.3.3 Ground Plane

The ground plane on the bottom side of the motherboard serves three functions. It reduces noise in the system, provides a uniform connection between all signals attached to it, and provides a uniform characteristic impedance for all logic signals.

A.3.4 Power and I/O Interconnect

In addition to the nine slots dedicated for use by the processor logic cards, the motherboard provides a tenth 88-pin edge connector for the power supply.

The +5 volt current is distributed to the logic cards along two buses that are located at opposite ends of the top side of the PC board. The width of the buses minimizes the voltage drop to the farthest side of the board.

The +12 and -12 volt signals are single buses on the top side. Of the 88 pins available on the power supply connector, 36 are for +5 volts, 6 are for +12 volts, 2 are for -12 volts, 42 are for voltage return (ground), and 2 provide the signals CBALARM/ and CBLFCLK from the power supply.

A.3.5 DMA Priority Daisy Chain

The processor uses a daisy chain priority system for implementing direct memory access between various peripherals, which means that a particular peripheral has a DMA priority based upon its physical location in the motherboard.

The system requires three signals for implementation: CBDMAREQ/, CBPRII, and CBPRIO. CBDMAREQ/ is common between all connectors; the other two signals are not. In general, each CBPRIO of a particular connector is connected to the CBPRII of the next connector. This forms the chain. Each logic card is responsible for using this system (if DMA is required) or for passing it along to the next card by shorting CBPPRIO together (if DMA is not required).

APPENDIX B
INSTRUCTION TIMINGS

The following are instruction timings for the 8600 and the 6600. All times are represented in microseconds (μs).

Instruction	8600 Timing	6600 Timing
L(rd)M	1.50	1.75
L(rd)M(rp)	2.25	2.60
LM(rs)	1.50	1.75
LM(rs)(rp)	2.25	2.60
L(rd)(rs)	0.75	1.00
L(r)data	1.50	1.45
AD(rs)	0.75	1.15
AC(rs)	0.75	1.15
SU(rs)	0.75	1.15
SB(rs)	0.75	1.15
ND(rs)	0.75	1.15
XR(rs)	0.75	1.15
CR(rs)	0.75	1.15
CP(rs)	0.75	1.00
AD(rs)(rd)	1.50	2.00
AC(rs)(rd)	1.50	2.00
SU(rs)(rd)	1.50	2.00
SB(rs)(rd)	1.50	2.00
ND(rs)(rd)	1.50	2.00
XR(rs)(rd)	1.50	2.00
)R(rs)(rd)	1.50	2.00
CP(rs)(rd)	1.50	1.85
ADM	1.50	2.10
ACM	1.50	2.10
SUM	1.50	2.10
SEM	1.50	2.10
NDM	1.50	2.10
XRM	1.50	2.10
ORM	1.50	2.10
CPM	1.50	1.95
ADM(rd)	2.25	2.95
ACM(rd)	2.25	2.95
SUM(rd)	2.25	2.95
SEM(rd)	2.25	2.95
NDM(rd)	2.25	2.95
XRM(rd)	2.25	2.95
ORM(rd)	2.25	2.95
CPM(rd)	2.25	2.80
AD data	1.50	1.60
AC data	1.50	1.60
SU data	1.50	1.60
SB data	1.50	1.60
ND data	1.50	1.60
XR data	1.50	1.60
)R data	1.50	1.60
)P data	1.50	1.45

Instruction	8600 Timing	6600 Timing
AD(r)data	2.25	2.45
AC(r)data	2.25	2.45
SU(r)data	2.25	2.45
SB(r)data	2.25	2.45
ND(r)data	2.25	2.45
XR(r)data	2.25	2.45
OR(r)data	2.25	2.45
CP(r)data	2.25	2.30
SLC	1.00	1.15
SRC	1.00	1.15
SRE	0.75	1.15
SLC(r)	1.75	2.00
SRC(r)	1.75	2.00
SRE(r)	1.50	2.00
JMP loc	2.25	2.05
Jcc loc	2.25	2.25
Jcc loc (fall thru)	1.75	1.10
EJMP loc	3.50	3.40
NOJ loc	1.75	1.00
NOP	0.75	0.70
CALL loc	3.75	2.20
Ccc loc	3.75	2.45
Ccc loc (fall thru)	1.75	1.20
RET	3.25	1.30
Rcc	3.25	1.50
Rcc (fall thru)	1.00	0.80
UR	4.00	2.45
EUR	4.00	3.15
IN		5.00
IN(r)		5.85
PIN		5.00
PIN(r)		5.85
EX(exp)		7.00
EX(r)(exp)		7.85
EX BEEP		7.00
EX CLICK		7.00
MIN		2.95 + 8.30N
MOUT		2.95 + 8.30N
BETA	4.75	1.20
ALPHA	4.75	1.20
DI	1.00	1.20
EI	1.00	1.20
POP	2.25	1.45
POP(rp)	3.00	2.30
PUSH	2.25	1.15
PUSH(rp)	3.00	2.00
PUSH loc	3.75	2.05

Instruction	8600 Timing	6600 Timing
BT(A=B=0)	1.50 + 1.50N	6.70 + 1.60N
BT(A,B=0)	1.50 + 2.25N	6.00 + 2.35N
BTR(A=B=0)	2.25 + 1.50N	7.85 + 1.60N
BTR(A,B=0)	2.25 + 2.25N	6.95 + 2.35N
BCV(A=B=0)	1.25 + 3.00	7.55 + 2.50N
CBV(A,B=0)	1.25 + 3.00	6.85 + 2.50N
BCP (if match)	1.75 + 2.00N	5.35 + 1.95N
CBP (mismatch)	2.25 + 2.00N	4.85 + 1.95N
BFAC	1.00 + 2.25N	5.35 + 2.15N
BFSB	1.00 + 2.25N	5.35 + 2.15N
DFAC	2.00 + 3.25N	6.20 + 3.45N
DFSB	2.00 + 3.25N	6.30 + 2.90N
BFSL	1.25 + 2.00N	3.00 + 1.70N
BFSR	2.00 + 2.00N	3.40 + 1.55N
STKS	1.25 + 3.25N	1.55 + 1.70N
STKL	1.25 + 3.25N	3.60 + 1.70N
REGS	10.00	9.10
REGL	7.25	9.85
CCS	1.00	1.65 to 2.45
CCS (r)	1.75	2.50 to 3.30
INCP HL	1.25	1.40 or 1.95
INCP HL,A	1.25	1.55 or 2.10
INCP (rp)	1.50	2.45 or 3.00
INCP (rp),2	2.50	2.50 or 3.05
INCP (rp),A	1.50	2.40 or 2.95
DECP HL	1.25	1.40 or 1.95
DECP HL,A	1.25	1.55 or 2.10
DECP (rp)	1.50	2.45 or 3.00
DECP (rp),2	2.50	2.50 or 3.05
DECP (rp),A	1.50	2.40 or 2.95
DL DE, HL	2.25	2.50
DL BC, HL	3.00	3.95
DL BC, BC	3.00	3.75
DL BC, DE	3.00	3.90
DL DE, BC	3.00	3.75
DL DE, DE	3.00	3.90
DL HL, BC	3.00	3.75
DL HL, DE	3.00	3.90
DL HL, HL	2.25	2.50
DS DE, HL	2.25	2.50
DS BC, HL	3.00	3.95
DS BC, DE	3.00	3.90
DS DE, BC	3.00	3.75
DS HL, BC	3.00	3.75
DS HL, DE	3.00	3.90
PL (r),loc	2.25	2.20
PS (r),loc	2.25	2.20
DPL (rp),loc	3.75	3.80
DPS (rp),loc	3.75	3.80

Instruction	8600 Timing	6600 Timing
INCI(dsp),(idx)	5.50	3.65 or 5.10
DECI(dsp),(idx)	5.50	3.65 or 5.10
INCI*(dsp),(idx)	7.00	7.25
DECI*(dsp),(idx)	7.00	7.45
LFII(rp),(dsp), (idx)	4.75	5.60
LFID(rp),(dsp), (idx)	4.75	5.80
LFII(rp),(dsp), (idx)	5.25	6.25
LFID(rp),*(dsp), (idx)	5.25	6.45
BRL	0.75	1.00
BRL(r)	1.50	1.85
STL	2.00 + 1.50N	2.70 + 1.25N
SC	5.00	1.80
BP	5.00	2.00
HALT	5.25	
IMULT	15.75-32.50	26.20-77.55
DIDIV	21.50-31.50	57.75-82.55
IDIV	22.00-32.00	5.75-82.55
DPLR	3.75	3.80
DPSR	3.75	3.80
STLO	3.00 + 1.50N	3.70 + 1.25N
INFO	2.00	2.50
BFLR	2.25 + 2.25N	6.80 + 2.15N
D(op)M(rp)	3.00 or 3.25	4.60 to 5.65
D(op)P(rp),loc	3.75 or 4.00	5.15 to 6.20
D(op)I(rp),data1	3.00 or 3.25	4.00 to 5.05
DM(op)(rp)	4.50 or 4.75	5.30 to 6.35
DMCP(rp)	4.50	
P(op)(r),loc	3.00	3.40 (3.25 for CP)
LLDEL	6.75	9.40
LLINS	9.00	10.80
COMP(rp)	2.25	3.70 or 4.75
COMPS(rp)	2.50	4.15 or 5.20
LKA	1.25	
LAS	1.25	
CBOUT	1.75	
CBOUT(r)	2.50	
CBIN	1.75	
CBIN(r)	2.50	

Instruction	8600 Timing	6600 Timing
CBSOUT	2.50	
CBSOUT(r)	3.25	
CBSIN	2.50	
CBSIN(r)	3.25	
STKMV	2.00	
IRET	4.75	
STR	2.25	
BLKIN	1.75 + 2.00N	
BLKOUT	1.75 + 2.00N	
MBLKIN	1.75 + 2.00N	
MBLKOUT	1.75 + 2.00N	
MS INTR	5.00	1.80
VECTORED INTR	7.75	
RESTART INTR	5.25	1.80
ERROR INTR	7.50	1.80

Notes on the instruction timings:

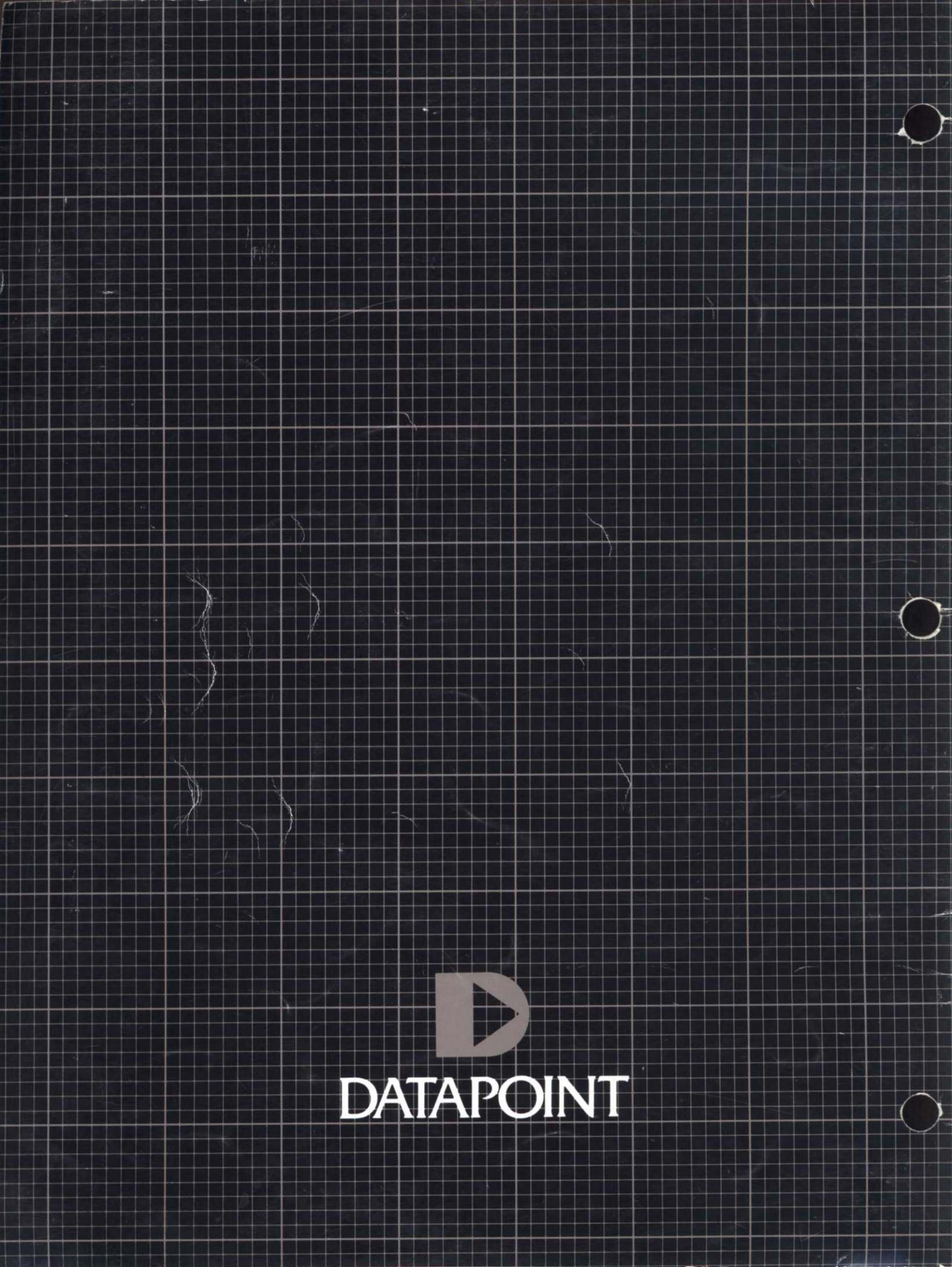
All Common Bus (CB) cycles include one wait state. Instructions include CBOUT, CBOUT(r), CBIN, CBIN(r), CBSOUT, CBSOUT(r), CBSIN, BLKIN, BLKOUT, and VECTORED INTERRUPT.

This page intentionally left blank.

APPENDIX C
8600 COMMON BUS I/O ADDRESSES

The following addresses are for the common bus I/O. All addresses not mentioned are reserved for future use.

I/O ADDRESS	FUNCTION
000-001	CP/RIM Interrupt Controller
0002	Aux Enable
0003	CP/RIM LEDs
0004	CP Status (Alarm & Burn-in)
010-017	Peripheral I/O
030-031	KDS
034-037	Printer Port USART
040	RIM #0 Enable (CP/RIM Board)
041	RIM 1-5 Enable
042-043	RIM 0 STAT-CMD
044-045	RIM 1 STAT-CMD
046-047	RIM 2 STAT-CMD
050-051	RIM 3 STAT-CMD
052-053	RIM 4 STAT-CMD
054-055	RIM 5 STAT-CMD
070-073	MPCA
200-207	MFCA #1
210-217	MFCA #2
2000-3777	UBUS Card
72000-73777	MPCA 1K Dual Port Memory
074000-077777	2K RIM Buffer
140000-143777	Screen Buffer (if in screen mode) Font Buffer (if in character front mode)
144000-147777	Attribute Buffer


DATAPOINT