DATAPOINT

Preliminary

Product Specification and Hardware Reference Manual

8603/8605 PROCESSOR

Document No. 61636

August 1983

PRELIMINARY
8603/8605 PRODUCT SPECIFICATION AND HARDWARE REFERENCE MANUAL

August 1983

Document No. 61636. Rev. 8/83.

System features and technical details are subject to change without notice.

TABLE OF CONTENTS

PART 1

GENERAL FEATURES

PART 2

INTERNAL BUS ARCHITECTURE

PART 3

KEYBOARD AND DISPLAY SUBSYSTEM (KDS)

PART 4
MEMORY

PART 5
PROCESSOR

PART 6
RESOURCE INTERFACE MODULE

PART 7
MICROBUS INTERFACE MODULE (MIFM)

PART 8
PERIPHERAL INPUT/OUTPUT MODULE

PART 9
PARALLEL BUS ADAPTER

PART 10
MULTIPORT COMMUNICATIONS ADAPTER (MPCA)

PART 11
MULTIFUNCTION COMMUNICATIONS ADAPTER (MFCA)

PART 12
SYSTEM FIRMWARE

APPENDIX A
ANCILLARY EQUIPMENT

APPENDIX B

APPENDIX C

LIST OF FIGURES

LIST OF TABLES

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.

Warning: This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause interference to radio communications. It has been tested and found to comply with the limits for a Class A computing device pursuant to Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such interference when operated in a commercial environment. Operation of this equipment in a residential area is likely to cause interference, in which case the user, at his own expense, will be required to take whatever measures may be required to correct the interference.

INTRODUCTION

This manual is designed to provide the programmer or other sophisticated user with the detailed information necessary for systems level programming of the DATAPOINT 86ØØ Business Computer.

Other manuals that may be helpful include the Product Specifications and Operator's Guides for individual components in the system as well as the appropriate software User's Guides.

PART ONE
GENERAL FEATURES

1.1 Introduction

The DATAPOINT 8600 is a versatile, high-performance processor featuring up to 512K bytes of memory. The 8600 supports both of the DATAPOINT operating systems (DOS, the Disk Operating System and RMS , the Resource Management System™) and is compatible with a wide range of peripherals. Including a large, easy-to-read amber display screen and a typewriter-style keyboard, the 8600 may function as part of a stand-alone system, as the host processor for a DATASHARE® Business Timesharing System, or as a member of an Attached Resource Computer® (ARC™) local area network.

Printed circuit boards contain all of the 8600's logical subassemblies; these boards are contained in an internal nine-slot card cage and linked by the DATAPOINT common bus. The basic 8600 processor includes a three-card central processor, 128K or 256K bytes of memory, a Keyboard/Display Subsystem (KDS) module, a serial port, and an internal Resource Interface Module (RIM). The 8600 also includes an internal power supply that provides DC power for the keyboard, display, and logic boards.

The 8603 processor is designed to perform as an ARC applications processor for tasks requiring full processor power in a single workstation. The only configurable option for an 8603 is memory size (a maximum of 256K of parity memory).

The 8605 processor--with 256K or 512K of ECC (Error Correction Code) memory--supports all feature options, including additional ECC memory, the Multiple Port Communications Adapter (MPCA), the Multiple Function Communications Adapter (MFCA), the Parallel Bus Adapter (PBA), and the PIO or Microbus disk interface. The 8605 is also available in system configurations with the DATAPOINT 9301, 9315, 9324, and 9325 disk drives for stand-alone or ARC operation. An 8603 can be converted to an 8605 with a field upgrade kit.

Figures 1-1 and 1-2 illustrate the 8603 and 8605 processors. The following sections introduce the basic elements of the 8600 processor family.

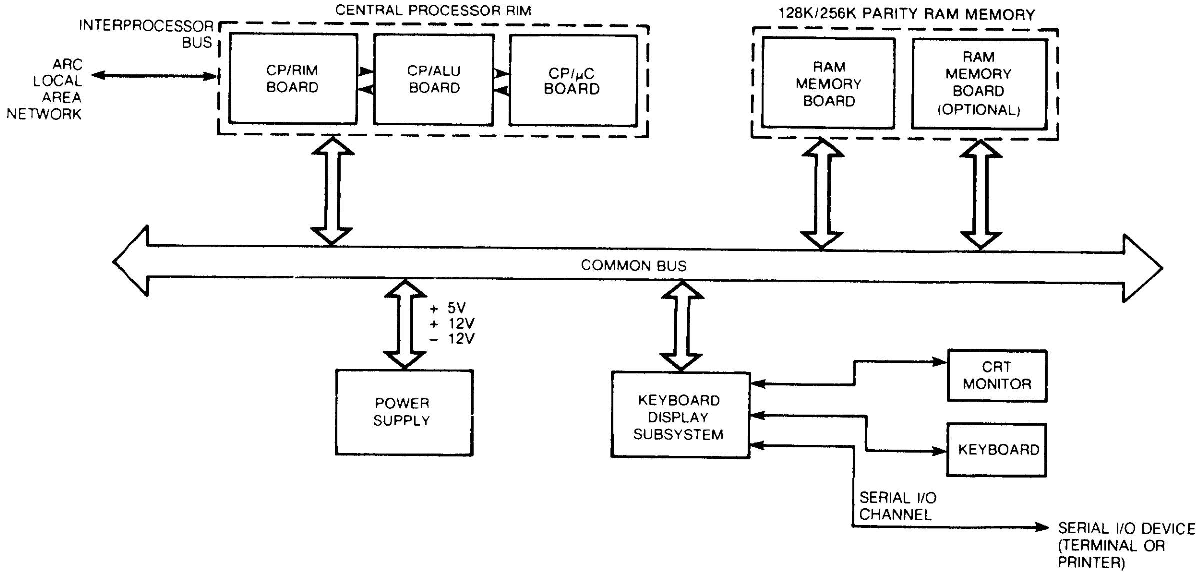
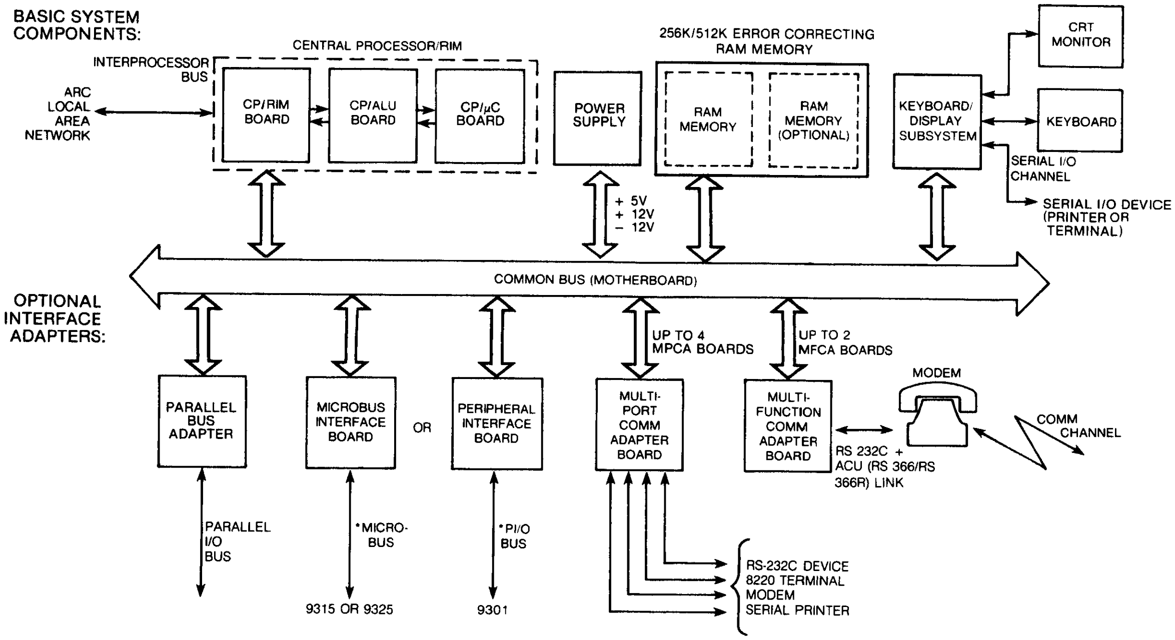


Figure 1-1: 86Ø3 Processor Block Diagram



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

Figure 1-2: 86Ø5 Processor Block Diagram

1.2 Internal Bus Architecture

The 8600 uses an internal bus to provide the interface between the processor, memory, Keyboard/Display Subsystem, and the internal option cards. 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 Input/Output (PIO) bus or the microbus. An external bus is linked directly to one of two types of internal logic boards (either a PIO module or a Microbus Interface Module, as explained in Section 1.6), which in turn is linked to the processor and main memory through the Common Bus.

1.3 Keyboard/Display Subsytem

The Keyboard/Display Subsystem (KDS) provides the interface between the keyboard and the processor. The KDS also controls the display screen and includes a serial I/O port for interface to a DATAPOINT serial printer or 8200-type 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, which includes multi-key roll-over for ease of typing, is connected to the processor housing by a one-meter cord.

The display screen uses a cathode ray tube illuminated by a magnetic deflection technique. It displays 1920 characters, organized as 24 rows of 80 characters each. Character attributes such as inverse video, underlining, two-level video, and blink are available on a character-by-character basis under software control. The display uses a standard set of 128-characters or a user-defined font. The viewing area is 5.4 by 8.9 inches (13.7 by 22.6 cm), and standard characters are produced using a 7- by 9-dot matrix in a 9- by 12-dot field; the display refresh rate is line synchronized at 60/50 frames per second. Display brightness can be set to one of 16 levels from the keyboard; beep and click are available under software control.

1.4 Memory

The basic 8603 includes 128K bytes of parity memory and is expandable to 256K. An 8605 includes 256K or 512K of ECC (Error Correction Code) memory. Memory modules are linked to the Common Bus and may be addressed by word or byte under software control.

1.5 Processor

The 86ØØ's central processor is implemented on three logic boards. The CP/mC (Microcode) board contains the instruction-decoding ROMs, the control store, and the sequencer. The CP/ALU (Arithmetic Logic Unit) encompasses most of the processor registers and manages the data flow within the machine. The CP/RIM (Resource Interface Module) contains the system and auxiliary ROM, the sector tables, and an integral RIM (the interface to the DATAPOINT Attached Resource Computer local area network).

The processor employs the DATAPOINT instruction set, and is driven by vectored interrupts, which means that it responds directly to interrupting devices. Access protect or write protect (or both) 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 the main memory and can use a stack exchange for multiple stacks. System and auxiliary ROMs contain system functions such as power-up, keyboard and display drivers, and interrupt, debug, and diagnostic routines.

1.6 Disk Interface Options

The 86Ø5 communicates with disk systems 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 an interface to the DATAPOINT series of disk drives through the PIO bus; the MIFM module provides an interface to the Datapoint 931Ø, 9315, 9324, and 9325 disk drives and the 14Ø1 diskette drive through the microbus. Either interface may be ordered with the 86Ø5.

1.7 Multiport Communications Adapter (MPCA) Option

The MPCA logic board supports four serial asynchronous RS-232-C communications ports; each port has full duplex transmit and receive capability. Data transfer rates are software-programmable from 5Ø to 19,2ØØ baud; character lengths and the number of stop bits are individually programmable. The MPCA provides parity bit generation and detection and includes a microprocessor, local memory, baud rate generators, and four UARTs (Universal Asynchronous Receiver/Transmitters). The 86Ø5 processor can be configured with a maximum of four MPCAs.

3T

1.8 Multiple Function Communications Adapter (MFCA) Option

The MFCA logic board provides a means of synchronous or asynchronous bidirectional information transfer between the processor an an RS-232-C compatible communications channel with reverse channel. The MFCA may be connected to an external modem and an RS-366 compatible Automatic Calling Unit (ACU) and can communicate over switched or leased lines using software-loadable communications protocols such as BISYNC, SDLC, HDLC, ADCCP, or GENSYNC. Baud rates are programmable, from 11Ø to 19.2K. The MFCA includes a microprocessor, a serial input/output channel, a counter/timer circuit, and 16K bytes of parity-protected local memory. The 86Ø5 can be configured with a maximum of two MFCAs.

1.9 Parallel Bus Adapter Option

The Parallel Bus Adapter is used to connect peripheral devices that contain DATAPOINT parallel interfaces to the 86Ø2 and 86Ø5 processors. These peripheral devices include printers, magnetic tape drives, and communications adapters.

The Parallel Bus Adapter implements all of the signals required for compatibility with the DATAPOINT 55ØØ and 66ØØ processor I/O buses (the Parallel Bus Adapter does not supply power to external devices). The adapter supplies +5V at 1.8 amperes (+5%, -1Ø%VDC) on the I/O bus to power the handshake circuits in a maximum of eight peripheral devices.

The logic is composed of data receivers and drivers, strobe drivers, parity generators and checkers, the time base, and diagnostic logic. Due to the functions performed by the Parallel Bus Adapter, only one may be configured with an 86Ø5.

1.1Ø Ancillary Equipment

In addition to the components discussed above, the 86ØØ includes an internal power supply and a motherboard. The power supply is housed in the internal card cage and provides DC power for the keyboard, the Keyboard/Display Subsystem, and the logic boards. The motherboard is a nine-card backplane that implements the 86ØØ Common Bus and a tenth card slot which is reserved for the power supply.

1.11 General Specifications

Power Requirements:
12Ø or 24Ø VAC (+/-1Ø%)
5Ø or 6Ø Hz (+/-1 Hz)
23Ø watts (785 BTU/hour, maximum)

Equipment Dimensions:

Keyboard

Width:	2Ø.ØØ inches	(5Ø.8 cm)
Height:	2.75 inches	(7.Ø cm)
Depth:	8.9Ø inches	(22.6 cm)

Processor:

Width:	2Ø.ØØ inches	(5Ø.8 cm)	
Height:	13.2Ø inches	(33.5 cm)	without base
Depth:	14.5Ø inches	(36.8 cm)	

Weight: 6Ø.ØØ pounds (22.4 kg)

Operating Environment:
5Ø to 1ØØ degrees Fahrenheit
1Ø to 38 degrees Celsius
2Ø to 9Ø percent relative humidity, noncondensing

1.12 Peripherals

The 86ØØ accommodates a wide variety of peripherals, including the DATAPOINT 93Ø1, 931Ø, 9315, 9324, and 9325 disk drives, the 14Ø1 diskette drive, printers, and communications equipment. Refer to the DATAPOINT Equipment Catalog (Model Code 6ØØØ1) for a complete description of peripherals.

1.13 Model Codes

86Ø3	Applications Processor, 128K-256K parity memory
86Ø5	Stand-alone/Data Resource Processor, 256K-512K ECC memory

NOTE: The 86Ø3 and 86Ø5 supersede the earlier 86Ø1 and 86Ø2 processor models.

PART 2
INTERNAL BUS ARCHITECTURE

2.1 General

The 8600 uses an internal bus, referred to as the Common Bus, as an interface between the processor, memory, and peripheral attachments. The Common Bus provides continuous instruction cycles. Each instruction cycle contains three or more clock cycles, which are referred to as T-states. Timing flexibility is provided by optional wait states.

The first T-state (T1) provides address and control information on the Common Bus, which is latched with an address strobe. Address is replaced with data during T2, and a read or write signal is provided along with a Transfer Strobe. In T3, data is transferred and then removed from the bus.

2.2 Interrupts

The Common Bus uses vectored interrupts. In this scheme, eight interrupt requests are provided. Each of these requests generates a two-byte entry address when acknowledged. The interrupt routine may then poll individual devices if more than one is sharing the interrupt address. For interrupt devices that require more than one interrupt, multiple vectors may be created through cascade operation.

The eight interrupt requests are listed below:

CBIREQ 0	MIFM Module
CBIREQ 1	Keyboard
CBIREQ 2	Printer Port
CBIREQ 3	MPCA
CBIREQ 4	PIO Module
CBIREQ 5	RIM
CBIREQ 6	MFCA
CBIREQ 7	Reserved

2.3 Direct Memory Access

Direct memory access is a technique by which a peripheral device may obtain control of the Common Bus for direct data storage or retrieval, thus facilitating high-speed data transfers between peripherals and memory. The peripheral device provides all required address registers, count registers, and timing logic. The processor relinquishes the bus in response to a request; the peripheral returns control of the bus to the processor after the required sequences have been performed.

2.4 Common Bus Signals

2.4.1 Multiplex Signals

The multiplex signals contain different data during the address and data portions of the cycle. Signals are controlled by the bus master during address. During data time, the data source device gates data or status information onto the bus. The signals (with address and data times) are given below:

	Address Time	Data Time
CBAD8-21	Address positive true	Data positive true
CBADØØ-Ø7	Address positive true	Data positive true
CBBW/PE/	Byte mode (1)	Memory Parity Error (Ø)
	Word mode (Ø)	No memory Parity Error (1)
CBMC/PA/	Memory cycle (Ø)	Data Parity available (Ø)
	Not memory cycle (1)	Not applicable (1)
CBRW/PL	Read cycle (1)	Odd parity for data bits Ø-7
	Write cycle (Ø)	Positive true
CBBPIO	I/O cycle page Ø (Ø)	Not applicable
CBIO/CC/	I/O cycle (Ø)	Memory Error detected
	Not I/O cycle (1)	by memory board (Ø)
CBSRE/	System ROM addressing(Ø)	Not applicable
CBRIH/	Inhibit RAM (Ø)	Valid only prior to CBRD/or CBWR/; data time strobe
CBSS/PH	Not applicable	Even parity for bits 8-15, positive true

2.4.2 Cycle Controls

The cycle control signals provide direct control of all bus timing.

CBAS/	Address Strobe - Negative True
CBRD/	Read Strobe - Negative True
CBWR/	Write Strobe - Negative True
CBTS/	Transfer Strobe - Negative True
CBMCLK	Memory Clock
CBIWAIT/	Input/Output Wait - Negative True
CBMWAIT/	Memory Wait - Negative True

2.4.3 Interrupt Control Signals

CBINTA/	Interrupt Acknowledge - Negative True
CBINTCY/	Interrupt Cycle - Negative True
CBINT/	Interrupt Request - Negative True
CBIREQØ-7/	Interrupt Request - Negative True

PART 3
KEYBOARD AND DISPLAY SUBSYSTEM (KDS)

3.1 General

The Keyboard and Display Subsystem (KDS) is a logic board that provides the interface between the keyboard and the central processor. The KDS also controls the display screen 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 3-1.

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

3.2 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 three feet (one meter) from the processor. Figure 3-2 illustrates the general purpose keyboard. Keyboard coding is given in Table 3-1.

3.2.1 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 upper-case character code. When this key is not pressed, the keyboard produces lower-case character code.

SHIFT LOCK -- The Shift Lock key locks the Shift key in the upper-case position. When Shift Lock is activated, the LED on this key cap is illuminated.

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

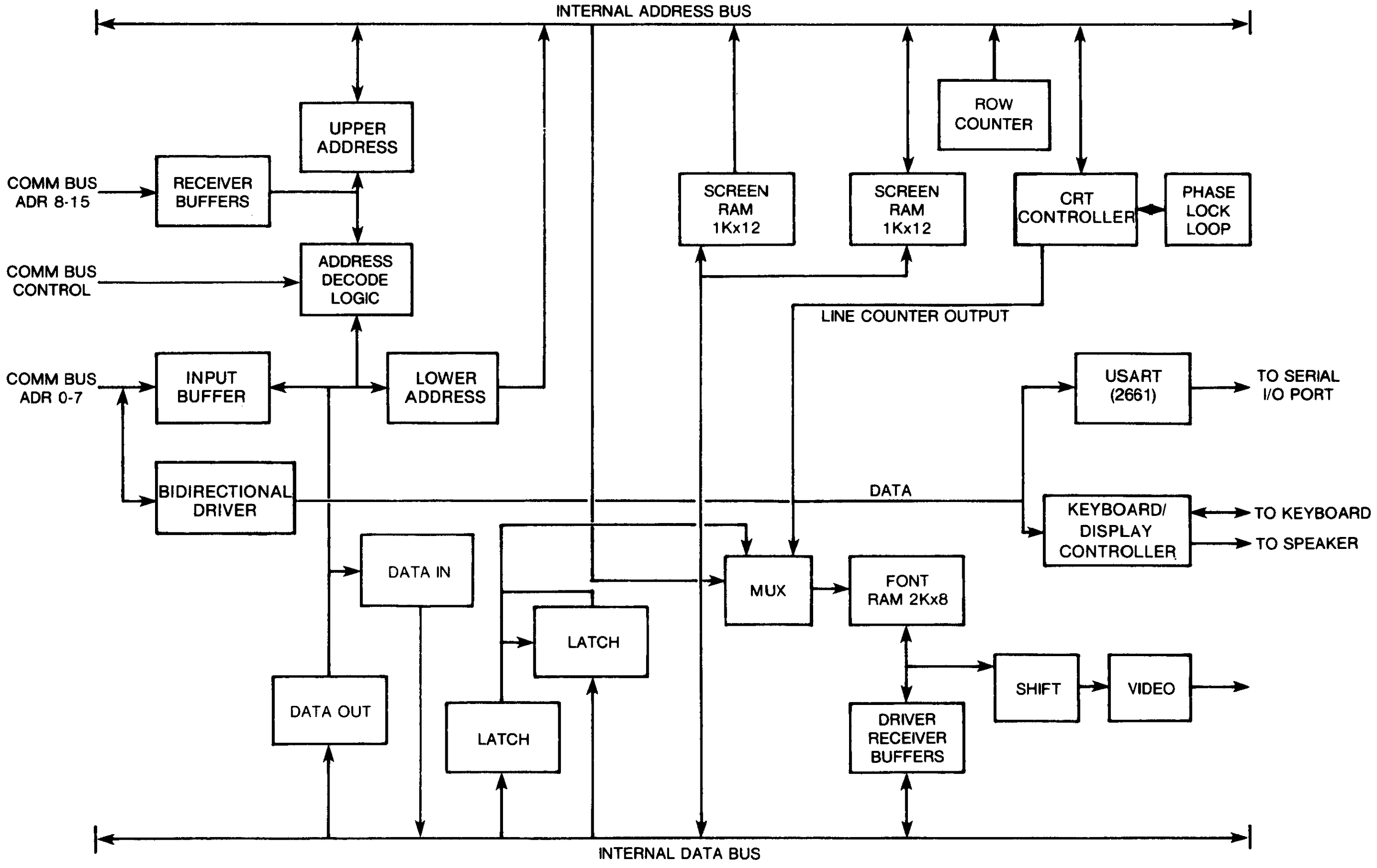
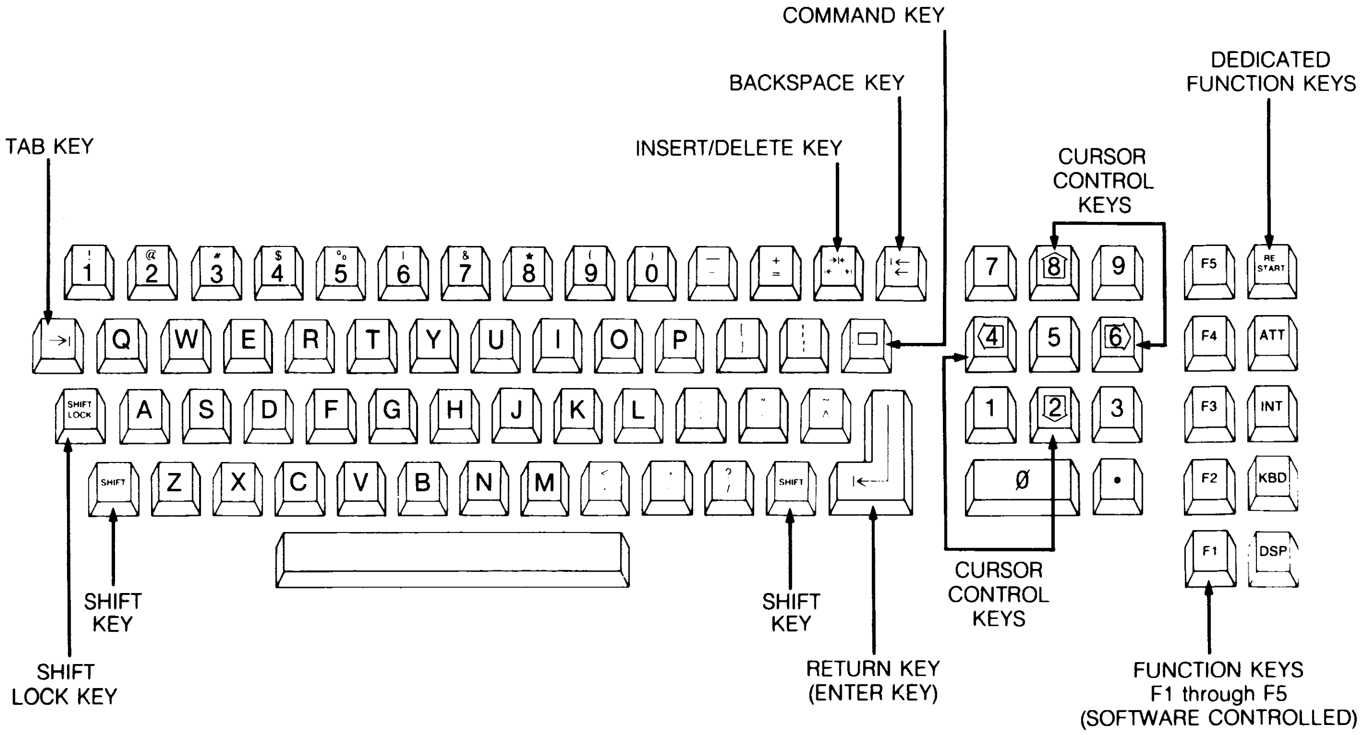


Figure 3-1: KDS Module



NOTE: ARROWS POINT TO EDIT
AND CONTROL KEYS

Figure 3-2: General Purpose Keyboard

INSERT/DELETE -- The Insert/Delete key is used in word processing to open text for an insertion 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 a backslash ( ) when unshifted and an accent grave ( ) 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.

3.2.2 Special Key Sequence Controls

The 86ØØ processor has six functions that can be initiated through the use of special keyboard keys. These functions are listed below, followed by a group of keys that must be held down to control the functions (when more than one key is named, the keys should be held down together):

Restart	CTRL, INT (release CTRL or INT)
Restart (RIM boot when disk is attached)	KBD, DSP, CTRL, INT (release CTRL or 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.
Restart Lockout	Same as Keyboard Lockout, except that the shifted ENTER key is pressed. To check for the presence of Restart Lockout, depress CTRL and DSP, then release DSP. A beep sounds if lockout is set.
Restart Unlock	Same as Restart Lockout.*
Keyboard Unlock	Same as Keyboard Lockout.*
Debug	DSP, CTRL, INT (release CTRL or INT).

*Restart Unlock and Keyboard Unlock can be used interchangeably.

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 still being held down) increases the brightness by one level. After a brightness level of zero has been reached, this command is ignored.

To decrease brightness, press the CTRL key followed by the KBD key. Every release of KBD (while CTRL is depressed) decreases the brightness by one level. After a brightness level of 15 has been reached, this command is ignored.

Table 3-1 General Purpose Keyboard Coding

A	1Ø1	a	141	Ø	Ø6Ø	:	Ø53
B	1Ø2	b	142	1	Ø61	;	Ø73
C	1Ø3	c	143	2	Ø62	<	Ø74
D	1Ø4	d	144	3	Ø63	=	137
E	1Ø5	e	145	4	Ø64	>	Ø76
F	1Ø6	f	146	5	Ø65	?	Ø77
G	1Ø7	g	147	6	Ø66	@	Ø42
H	11Ø	h	15Ø	7	Ø67	[	1ØØ
I	111	i	151	8	Ø7Ø	]	14Ø
J	112	j	152	9	Ø71	^	135
K	113	k	153	Space	Ø4Ø	_	Ø75
L	114	l	154	!	Ø41	{	174
M	115	m	155	"	Ø52	|	Ø46
N	116	n	156	#	Ø43	}	134
O	117	o	157	$	Ø44	~	175
P	12Ø	p	16Ø	%	Ø45	Tab	Ø33*
Q	121	q	161	&	Ø47	Tab	233
R	122	r	162	'	Ø72	Return	Ø15*
S	123	s	163	(	Ø51	Return	215
T	124	t	164	)	ØØØ	Backspace	Ø1Ø*
U	125	u	165	*	Ø5Ø	Backspace	21Ø
V	126	v	166	+	177	Insert	133*
W	127	w	167	,	Ø54	Delete	173
X	13Ø	x	17Ø	-	Ø55	Command	136*
Y	131	y	171	.	Ø56	Command	176
Z	132	z	172	/	Ø57		

Numeric Pad

Symbol	Unshifted	Shifted
.	256	256
Ø	26Ø	36Ø
1	261	361
2	262	362
3	263	363
4	264	364
5	265	365
6	266	366
7	267	367
8	27Ø	37Ø
9	271	371

Function Keys

Symbol	Unshifted		Shifted	
	D**	R**	D**	R**
F5	3ØØ	32Ø	34Ø	32Ø
F4	3Ø2	322	342	322
F3	3Ø4	324	344	324
F2	3Ø6	326	346	326
F1	31Ø	33Ø	35Ø	33Ø
INT	3Ø1	321	341	321
ATT	3Ø3	323	343	323
CTRL	3Ø5	325	345	325
KBD	3Ø7	327	347	327
DSP	311	331	351	331

*Unshifted representation

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

NOTE: These are key codes as presented directly from the keyboard and, in general, undergo translation in firmware or software. See Chapter 1Ø for key codes as presented from system firmware keyin routines.

3.3 Display

The display uses a magnetic deflection technique with an amber cathode-ray-tube screen. It provides a display of 192Ø characters, organized as 24 rows of 8Ø 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, 7 x 11 for lower-case descenders--see figure 3-3). The display/refresh rate is line synchronized at 6Ø/5Ø frames per second.

The screen refresh memory is organized as 2K bytes in I/O memory space located at address Ø14ØØØØ 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 ØØØØ and ØØØ1). Any of the 192Ø 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 ØØØ2-ØØ61, with the first character address pointer for the first row at ØØØ2 and ØØØ3, and the first character address pointer for the last row at ØØ6Ø and ØØ61. 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 (Ø14ØØØØ).

3.3.1 Display Character Format

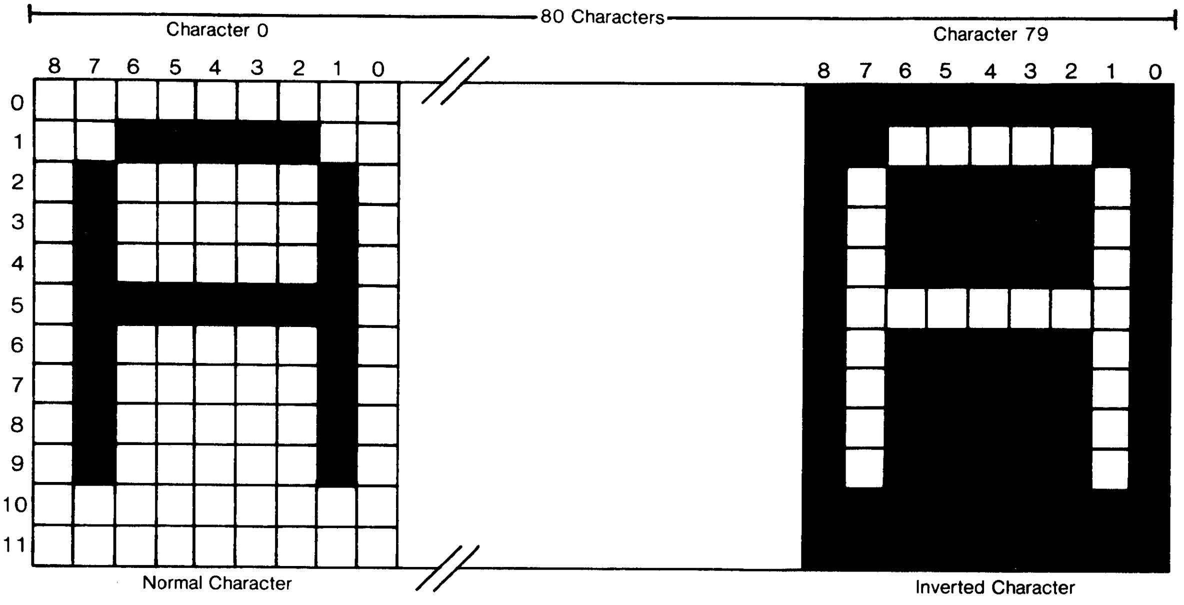
Every display character is stored in the screen memory as a byte using its ASCII value. The lower seven bits represent the ASCII value of the display character; the eighth bit is a video attribute. The seven ASCII bits are used to obtain the start address of the character in the display font RAM. The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

3.3.2 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 blonk are discussed in Section 3.5.3.



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

Figure 3-3: Character Row

3.3.3 Character Font Load

Displayed characters are generated from a loadable RAM memory, providing for foreign-language character sets and other user-selected character set variations. The RAM is down-line loaded through optional RCM software by the main program. The character generation RAM is a 2Ø48 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 Ø14ØØØØ 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	1Ø	9	8	7	6	5	4	3	2	1	Ø

The top line of the display character is defined by the eight bits of data at the start address. The next nine 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.

3.4 Serial I/O Port

The KDS is equipped with one standard RS-232-C serial interface for connection to a DATAPOINT printer or workstation. The baud rate is software controlled from 5Ø to 192ØØ 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.

3.5 KDS Programming Considerations

3.5.1 Keyboard and Screen

Status and commands for the keyboard and display are accessed by reading from base page I/O address Ø31 for status or writing to it for commands; the status word format follows. Once the appropriate command has been issued and executed by the KDS, any requested status is placed in the Data Bus Buffer (DBB) at base page I/O address Ø3Ø.

```
MSB  ----------------------------------------     LSB
      |B7 | B6 | B5 | B4 | B3 | B2 | B1 | BØ |
      ----------------------------------------
        |    |    |    |    |    |    |    |_____CBF
        |    |    |    |    |    |    |__________IBF
        |    |    |    |    |    |_______________FØ
        |    |    |    |    |____________________F1
        |    |    |    |_________________________Keyboard Key
        |    |    |______________________________Display Key
        |    |___________________________________DTX
        |________________________________________Keyboard
                                                   Character
```

The following descriptions explain the bits.

Bit Ø -- Output Buffer Full (OBF). This is set by hardware whenever the KDS writes 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 FØ 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 Ø 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.

3.5.1.1 Commands

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

READ REQUEST COMMANDS

```
MSB   -------------------------------------   LSB
      ! B7 | B6 | B5 | B4 | Ø | Ø | Ø | Ø |
      -------------------------------------
```

The bits are defined as follows:

Bit Ø-Bit 3 -- ØØØØ informs the KDS that it is a read request command.

Bit 4-Bit 7 -- ØØØØ commands the KDS to reset.

ØØØ1 commands the KDS to present the key-down status in the DBB.

ØØ1Ø commands the KDS to present the screen status byte in the DBB.

Ø1ØØ commands the KDS to present the keyboard character into the DBB.

1ØØØ commands the KDS to present the keyboard type in the DBB.

The format for the screen status byte is shown on the following page:

```
MSB-------------------------------------------   LSB
   | B7 | B6 | B5 | B4 | B3 | B2 | B1 | BØ |
   -------------------------------------------
     |    |    |    |    |    |    |    |_____Reserved
     |    |    |    |    |    |    |__________Reserved
     |    |    |    |    |    |_______________Reserved
     |    |    |    |    |____________________Reserved
     |    |    |    |_________________________Ø=Block Cursor
     |    |    |                            1=Underline Cursor
     |    |    |____________________________Ø=No Control Key
     |    |                                 Filter, 1=Control
     |    |                                 Key Filter
     |    |_________________________________Ø=Normal Screen,
     |                                      1=Blank Screen
     |__________________________________    Ø=Screen Mode
                                            1=Font Load Mode
```

The keyboard type byte is shown below:

```
MSB -----------------------------------------   LSB
    | B7 | B6 | B5 | B4 | B3 | B2 | B1 | BØ |
    -----------------------------------------
      |    |    |    |    |    |    |    |____Reserved
      |    |    |    |    |    |    |_________Reserved
      |    |    |    |    |    |______________1=General
Purpose
      |    |    |    |    |                  Keyboard, Ø=Any
other
      |    |    |    |    |                  keyboard
      |    |    |    |    |___________________Reserved
      |    |    |    |________________________Reserved
      |    |    |_____________________________Reserved
      |    |__________________________________Reserved
      |_______________________________________Reserved
```

The key-down status byte is shown below. Each status bit is a one when the corresponding key is depressed and remains a one until the key is released.

```
MSB   -----------------------------------------   LSB
      | B7 | B6 | B5 | B4 | B3 | B2 | B1 | BØ |
      -----------------------------------------
        |    |    |    |    |    |    |    |____F1 Key
        |    |    |    |    |    |    |_________F2 Key
        |    |    |    |    |    |______________F3 Key
        |    |    |    |    |___________________F4 Key
        |    |    |    |________________________F5 Key
        |    |    |_____________________________CTRL Key
        |    |__________________________________ATT Key
        |_______________________________________INT Key
```

SCREEN COMMANDS

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

```
MSB                                                LSB
     ____________________________________________
     | B7 | B6 | B5 | B4 | Ø | Ø | 1 | Ø |
     --------------------------------------------
        |    |    |    |    |________________|
        |    |    |    |            |
        |    |    |    |    Screen Control Word
        |    |    |    |__________Underline/Block Cursor
        |    |    |_______________Control Key Filter
        |    |____________________Blank/Normal Mode
        |_________________________Load/Normal Mode
```

Bits Ø-3 -- ØØ1Ø informs the KDS that it is a screen control word.

Bit 4 -- When this bit is set to 1, the KDS sets the underline cursor mode. When it is a Ø, the KDS sets the block cursor mode (on powering up, the cursor is set to the block mode).

Bit 5 -- When this bit is set to 1, the KDS does not send control key codes to the master. When this bit is reset to Ø, all key codes are sent to the master, except for the special key sequences (see Section 3.2.2).

Bit 6 -- When this bit is set to 1, the KDS blanks the screen. When it is set to Ø, 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 powering up, normal mode is set).

Brightness Control

```
MSB                                                  LSB
     ______________________________________________
     | B7 | B6 | B5 | B4 | Ø | Ø | 1 | 1 |
     ----------------------------------------------
     |___________________|  |___________________|
               |                      |
          Brightness        Brightness Control Byte
```

Bits Ø-3 -- ØØ11 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 sixteen levels, Ø being the brightest and 15 being the dimmest.

RESTART CLEAR COMMANDS

```
MSB                                                      LSB
        _________________________________________________
        ! X | X | X | X | Ø | 1 | Ø | Ø |
        -------------------------------------------------
        |_______________________|

|_____________________________________Reserved
```

Bits Ø-3 -- Ø1ØØ 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.

3.5.2 Speaker Commands

Speaker control is initiated by writing to base page I/O address Ø31. The byte format is as follows:

```
MSB                                              LSB
          ________________________________________
          | B7 ! B6 | B5 ! B4 | Ø | Ø | Ø | 1 |
          ----------------------------------------
            !    |  !_________|
            !    !       |
            |    |   Reserved
            |    |
            |  Beep Control
            |
          Click Control
```

Upon receipt of this command, the KDS initiates the beep or click operation. The beep is a 12ØØHz tone generated for a duration of 25Ø milliseconds. The click is an 8ØØ microsecond pulse with a minimum of 8ØØ microseconds guaranteed between clicks.

Bits Ø-3 -- ØØØ1 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.

3.5.3 Video Attributes

Underline, two-level video, and blink are programmed by writing to the attribute RAM. Screen character locations run from Ø14Ø1ØØ to Ø143777, and attribute RAM extends from Ø144ØØØ to Ø147777. Attribute data for a particular screen byte resides at a Ø4ØØØ offset from the screen RAM location; therefore, screen data at location Ø14Ø1ØØ has its attributes at extended I/O address Ø1441ØØ and so on. The bit positions for attributes are shown below:

```
MSB                                             LSB
     _________________________________________
     | B7 | B6 | B5 | B4 | B3 | B2 | B1 | BØ |
     -----------------------------------------
       |    |    |    |    |    |    |    |___Underline
       |    |    |    |    |    |    |________Two-level
       |    |    |    |    |    |_____________Blink
       |    |    |    |    |__________________Reserved
       |    |    |    |_______________________Reserved
       |    |    |____________________________Reserved
       |    |_________________________________Reserved
       |______________________________________Reserved
```

As mentioned in Sections 3.3.1 and 3.3.2, the inverse video attribute bit is controlled by manipulation of the most significant bit of the screen RAM byte for a particular character.

3.5.4 Serial I/O Port

The KDS serial I/O port is addressed through base page I/O address Ø34 to Ø37.

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 filling of the receive holding register by a received character causes an interrupt to the CPU on Interrupt Request Line Two. The CPU then polls the status register of the USART and determines whether it was a receive or transmit interrupt. The status register bit positions are shown in Figure 3-4.

Secondary Receive Line Signal Detect is connected to Clear To Send on the USART. Clear To Send must be low, and TXEN in the command register (Figure 3-5) must be set in order to transmit. If the printer is busy, it pulls this line high, and transmission is halted until it is taken low again. This signal is not used for a terminal and no connection is necessary for transmission.

The RS-232-C signal Data Set Ready is connected to the Data Set Ready on the USART and appears as bit 7 in the status register (Figure 3-4). It is connected to Data Terminal Ready on the printer (power on).

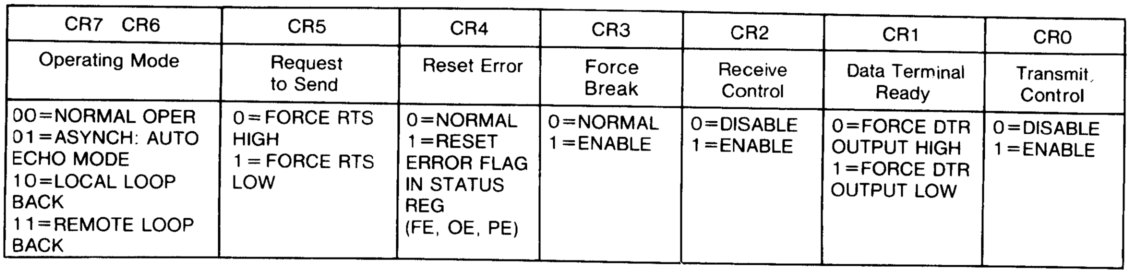
Data Carrier Detect must be low, and RXEN in the command register (Figure 3-5) must be set for the receiver to operate. This signal is used only for terminals.

Loading the mode and command registers initializes the USART. The first read or write to address Ø36 addresses mode register 1; the second read or write to that address addresses mode register 2 (Mode register 1 bit definitions are shown in Figure 3-6). It contains parity type and control and character length; unused bits must be programmed as shown. Mode register 2 is broken down in Figure 3-7. It contains the baud rate selection. Once again, unused bits must be programmed as specified.

Manipulation of the command register is accomplished by access to address Ø37. This register is shown in Figure 3-5.

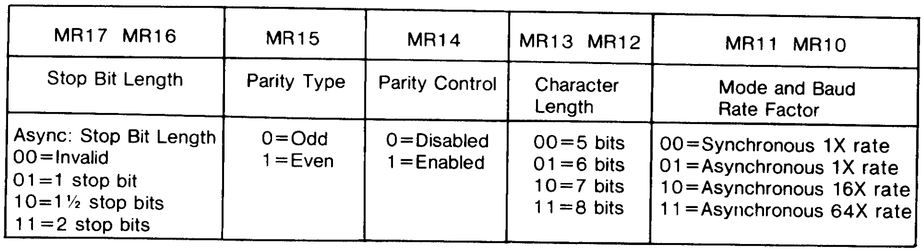
SR7	SR6	SR5	SR4	SR3	SR2	SR1	SR0
Data Set Ready	Data Carrier Detect	Framing ERROR	Overrun	Parity ERROR	TxEMT/ DSCHG	RxRDY	TxRDY
0=DSR INPUT IS HIGH 1=DSR INPUT IS LOW	0=DCD INPUT IS HIGH 1=DCD INPUT IS LOW	0=NORMAL 1=FRAMING ERROR	0=NORMAL 1=OVERRUN ERROR	0=NORMAL 1=PARITY ERROR	0=NORMAL 1=CHANGE IN DSR OR DCD, OR TRANSMIT SHIFT REGISTER 16	0=RECEIVE HOLDING REG EMPTY 1=RECEIVE HOLDING REG HAS DATA	0=TRANSMIT HOLDING REG BUSY 1=TRANSMIT HOLDING REG EMPTY

Figure 3-4: Status Register



CR7 CR6	CR5	CR4	CR3	CR2	CR1	CR0
Operating Mode	Request to Send	Reset Error	Force Break	Receive Control	Data Terminal Ready	Transmit, Control
00=NORMAL OPER 01=ASYNCH: AUTO ECHO MODE 10=LOCAL LOOP BACK 11=REMOTE LOOP BACK	0=FORCE RTS HIGH 1 = FORCE RTS LOW	0=NORMAL 1=RESET ERROR FLAG IN STATUS REG (FE, OE, PE)	0=NORMAL 1=ENABLE	0=DISABLE 1=ENABLE	0=FORCE DTR OUTPUT HIGH 1=FORCE DTR OUTPUT LOW	0=DISABLE 1=ENABLE

Figure 3-5: Command Register



MR17 MR16	MR15	MR14	MR13 MR12	MR11 MR10
Stop Bit Length	Parity Type	Parity Control	Character Length	Mode and Baud Rate Factor
Async: Stop Bit Length 00=Invalid 01=1 stop bit 10=1½ stop bits 11=2 stop bits	0=Odd 1=Even	0=Disabled 1=Enabled	00=5 bits 01=6 bits 10=7 bits 11=8 bits	00=Synchronous 1X rate 01=Asynchronous 1X rate 10=Asynchronous 16X rate 11=Asynchronous 64X rate

Figure 3-6: Mode Register 1

MR27 - MR24											MR23 - MR20
	TxC	RxC	Pin 9	Pin 25		TxC	RxC	Pin 9	Pin 25	Mode	Baud Rate Selection
0000	E	E	TxC	RxC	1000	E	E	XSYNC	RxC/TxC	sync	See baud rates in table 1.
0001	E	I	TxC	1X	1001	E	I	TxC	BKDET	async	
0010	I	E	1X	RxC	1010	I	E	XSYNC	RxC	sync	
0011	I	I	1X	1X	1011	I	I	1X	BKDET	async	
0100	E	E	TxC	RxC	1100	E	E	XSYNC	RxC/TxC	sync	
0101	E	I	TxC	16X	1101	E	I	TxC	BKDET	async	
0110	I	E	16X	RxC	1110	I	E	XSYNC	RxC	sync	
0111	I	I	16X	16X	1111	I	I	16X	BKDET	async	

Figure 3-7: Mode Register Mode 2

PART 4
MEMORY

4.1 General

The main memory of the 8600 processor is implemented in printed circuit boards housed in the processor's internal card cage. Two different types of memory are available, each in two different sizes. Parity memory may be configured with either 128K or 256K bytes; error correction code (ECC) memory is available in either 256K or 512K bytes (parity and ECC memory may not be mixed). The memory module is linked to the Common Bus and arbitrates memory-cycle and memory-refresh requests.

The memory may be requested dynamically to perform in either the word- or byte-access mode. The word-access mode is restricted to addressing even boundaries only; the memory-access mode may be accomplished through bus master cycles or using Direct Memory Access.

Section 4.2 describes the parity memory module; the ECC memory module is described in section 4.3.

4.2 Parity Memory

Memory cycles consist of word writes and reads, byte writes and reads, and refresh. These cycles are discussed in the following sections. Timings are listed below:

Read Cycle	750 ns (typical)
Write Cycle	750 ns (typical)
Refresh Cycle	250 ns (typical)
Access Time	300 ns (maximum)

The 128K parity memory module is schematically illustrated in Figure 4-1.

4.2.1 Word Read (Parity Memory)

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 from 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 (Parity Memory)

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 receives them. During even byte reads, the corresponding odd byte is driven to the MSB of the bus, although it is assumed no device receives it.

4.2.3 Word Write (Parity Memory)

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 (Parity Memory)

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.

4.2.5 Refresh Cycles (Parity Memory)

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.

A refresh is performed by selecting one of 128 rows in the memory and strobing the chip with the Row Address Strobe. 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

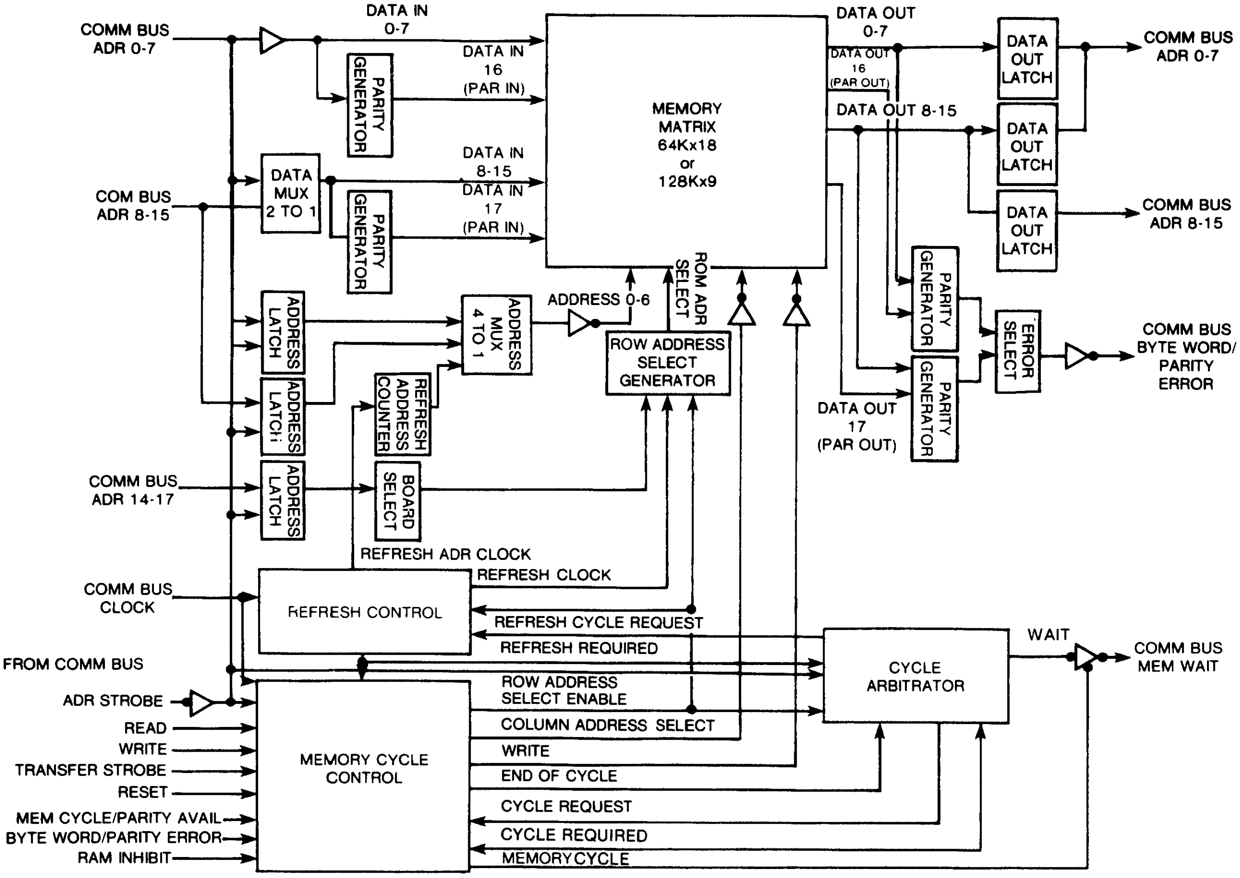


Figure 4-1 Parity Memory Module

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.

4.2.6 Cycle Arbitration (Parity Memory)

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 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.

4.2.7 Error Detection (Parity Memory)

The memory carries parity over each eight-bit byte. When a write to memory occurs, a ninth bit (for each byte) is generated using 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.

4.3 ECC Memory

Each ECC memory module contains 256K or 512K bytes and uses six check bits for each word (16 bits) of memory. Word accesses are performed only on even boundaries; the ECC control logic corrects all single-bit errors and flags double-bit errors. Figure 4-2 is a schematic illustration of the ECC memory module. Timings are listed below:

Read Cycle	75Øns (typical)
Write Cycle	75Øns (typical)
Refresh Cycle	25Øns (typical)
Access Time	3ØØns (typical)
Read and Correct Cycle	1ØØØns (typical)

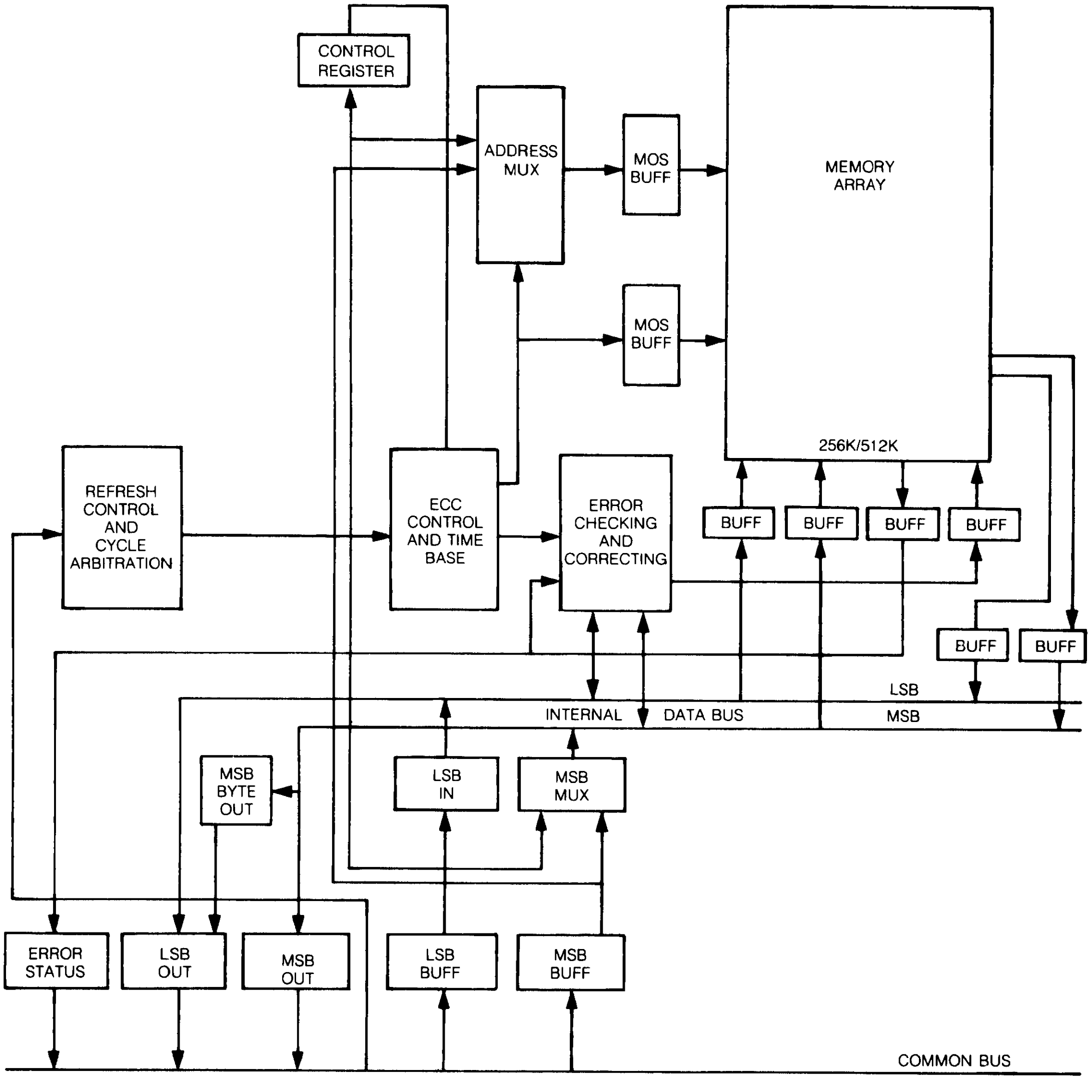


Figure 4-2 ECC Memory Block Diagram

4.3.1 Word Read (ECC Memory)

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 from 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. 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.

4.3.2 Byte Read (ECC Memory)

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 receives them. During even byte reads, the corresponding odd byte is driven to the MSB of the bus, although it is assumed no device receives it.

4.3.3 Word Write (ECC Memory)

All ECC memory writes are performed as read-modify-write cycles. At the beginning of the write cycle, the 22-bit word is read in by the ECC control logic, which then compares the 16 bits of data to the six check bits to confirm that the data is correct.

After performing the check, the new 16-bit word is used to generate new check bits; all 22 bits are then written to the memory array.

4.3.4 Byte Write (ECC Memory)

The byte write cycle is similar to the word write cycle for the ECC memory except that after checking the old contents of the 16-bit word, one byte is replaced with the new data while the unaddressed byte is held in a register. The two bytes are then recombined into a 16-bit word, which is used to generate six new check bits. The two bytes and their check bits are then written to the memory array as a 22-bit word.

4.3.5 Refresh Cycles (ECC Memory)

Refresh and refresh arbitration are performed by a custom gate array. An internal counter is driven by the synchronous 4 MHz System Clock, and this determines when a refresh is required. If a Read or Write cycle is in progress at the time the refresh is needed, then the refresh is blocked until the current cycle is completed. The refresh can then proceed, with the strobes provided by the custom gate array.

Similarly, a memory cycle is blocked if a refresh cycle is under way. There are 128 refresh cycles performed every two milliseconds or 256 refresh cycles performed every four milliseconds (depending on the RAM).

4.3.6 Error Detection (ECC Memory)

Error detection and correction are performed by means of generating a six-bit Hamming code accross 16 bits of data during writes. While reading the data, a check is made across the entire 22-bit word (16 bits of data, 6 bits for checking) for errors.

A single LSI chip does all of the bit generation and error detection performed by the ECC memory. It is capable of single-bit error correction and double-bit error detection (when a double-bit error is detected, an error condition signal is placed on the Common Bus).

4.3.7 Diagnostic Features (ECC Memory)

The ECC memory board contains several diagnostic features. There are two write registers and two read registers for diagnostic purposes.

These registers are located at I/O address octal Ø6Ø and octal Ø62.

4.3.7.1 Status Register

This register, when read, contains the six check bits and two bits reflecting the error status of the last memory read.

```
 MSB                                LSB
---------------------------------------------
|ME | ER | CK5 | CK4 | CK3 | CK2 | CK1 | CKØ|
---------------------------------------------
  |   |    |     |     |     |     |      |
  |   |    |     |     |     |     |      ------Check bit Ø
  |   |    |     |     |     |     -------------Check bit 1
  |   |    |     |     |     -------------------Check bit 2
  |   |    |     |     -------------------------Check bit 3
  |   |    |     -------------------------------Check bit 4
  |   |    -------------------------------------Check bit 5
  |   ------------------------------------------Error
  ----------------------------------------------Multierror
```

ER - This bit is negative true and indicates that an error occurred on the last memory read. If this bit is true and the ME bit is false, then a correctable error has occurred. If both this bit and the ME bit are true, then a non-correctable error has occurred.

ME - This bit is negative true and indicates that a multiple error occurred on the last read.
CKØ-5 - These six bits are the check bits from memory on the last read.

4.3.7.2 Control Register

This register is a read/write register. It controls the error correction unit diagnostic modes and processor notification of single-bit errors.

```
MSB                                  LSB
--------------------------------------
| Ø | LE | Ø | Ø | DE | D2 | D1 | PU |
--------------------------------------
  |    |    |   |    |    |    |    |
  |    |    |   |    |    |    |    -------Pass Through
  |    |    |   |    |    |    ------------Diagnostic 1
  |    |    |   |    |    -----------------Diagnostic 2
  |    |    |   |    ----------------------Disable Single-Bit
  |    |    |   |                          error
  |    |    |   ---------------------------Always zero
  |    |    -------------------------------Always zero
  |    |                                   Latched Single-Bit
  |    ------------------------------------error
  -----------------------------------------Always zero
```

PU - When set to zero, this bit turns off all error correction and detection operations for ECC memory data. This is known as the pass-through mode and overrides the D1 and D2 bits below.

The D1 and D2 bits act together to set four modes of ECC memory operation.

D2, D1=ØØ NORMAL MODE. This is the normal operating mode with all error correction and detection functions turned on.

D2,D1=Ø1 DIAGNOSTIC GENERATE MODE. This mode takes the contents of the diagnostic latch (see section 4.3.7.3) and substitutes them for the normally generated six check bits. Normal correction takes place on read.

D2,D1=1Ø DIAGNOSTIC CORRECT MODE. This mode substitutes the contents of the diagnostic latch and substitutes them for the normally read six check bits. In this mode, the check bits are normally generated on write.

D2,D1=11 INITIALIZE MODE. Used during initialization to write zeroes to memory with good check bits while disabling error detection.

DE - When set to one, this bit allows single-bit errors detected to cause a Common Bus memory parity error interrupt. This bit should be cleared for normal single-bit correction to take place.

LE - This bit is a one if a single bit error has occurred since the bit was last cleared. This bit is operational only when DE is set to zero, but is cleared by setting DE to a one.

4.3.7.3 Diagnostic Check Bit Latch

This is a write-only register used in conjunction with the Diagnostic Generate and Correct Modes described in section 4.3.7.2.

```
MSB                                     LSB
--------------------------------------------
| Ø | Ø | CK5 | CK4 | CK3 | CK2 | CK1 | CKØ|
--------------------------------------------
  |   |    |     |     |     |     |     |
  |   |    |     |     |     |     |     ------Check bit Ø
  |   |    |     |     |     |     ------------Check bit 1
  |   |    |     |     |     ------------------Check bit 2
  |   |    |     |     ------------------------Check bit 3
  |   |    |     ------------------------------Check bit 4
  |   |    ------------------------------------Check bit 5
  |   ---------------------Reserved-Need to be written as Ø
  -------------------------Reserved-Need to be written as Ø
```

PART 5
PROCESSOR

5.1 General

The 8600 processor is composed of three printed circuit boards housed in the internal card cage:

CP/mC (Microcode) -- contains the instruction decode ROMs, 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 and auxiliary ROMs, interrupt controller, sector tables, and a RIM for interface to a Datapoint ARC system. The older CP/RIM I board has a maximum memory addressing capability of 256K, while the CP/RIM II can address larger amounts (up to 512K).

Block diagrams of the CPU board set are shown in Figures 5-1 and 5-2. This chapter covers processor features in detail; the RIM and its functions are described in Chapter 6.

5.1.1 CP/RIM I & II Differences

The following list details the differences between the CP/RIM I board and the CP/RIM II board.

* The sector tables of the CP/RIM I board have been extended in the CP/RIM II board to expand the addressing capability. System addressing capability is under software control.

* On the CP/RIM II board, the diagnostic bit has been removed from the CP/RIM status register and a Sector Table Select readback bit has been added. Bits 2 through 5 of the status register now reflect the current state of diagnostic LEDs.

* The top-of-board connector on the CP/RIM I board has been changed to a conventional 40-pin header type on the CP/RIM II board.

* The eight-bit RIM identification switches are mounted on the top edge of the CP/RIM II card to allow them to be set without having to remove the board from the card cage.

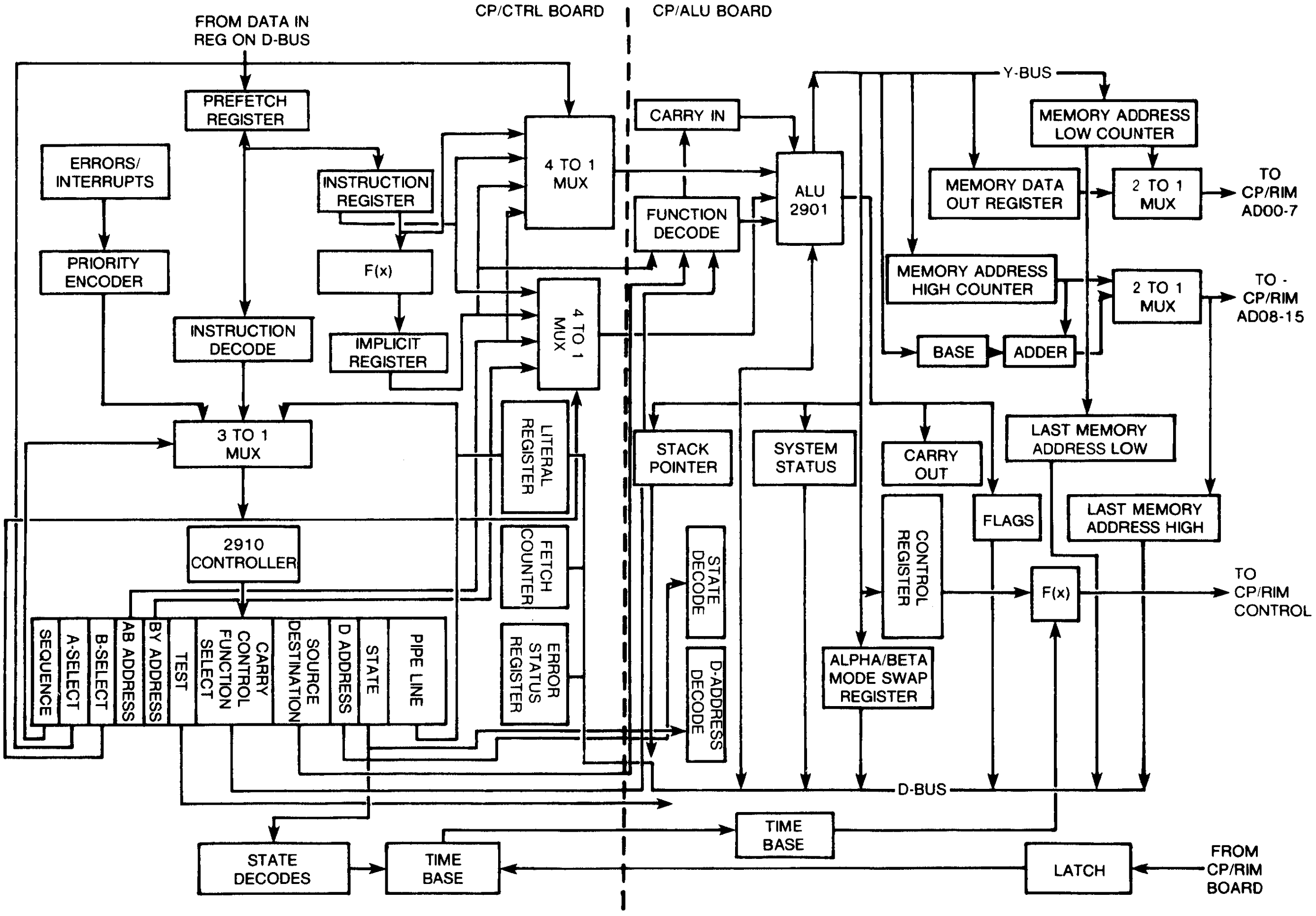


Figure 5-1: CP/ALU and CP/CTRL Boards

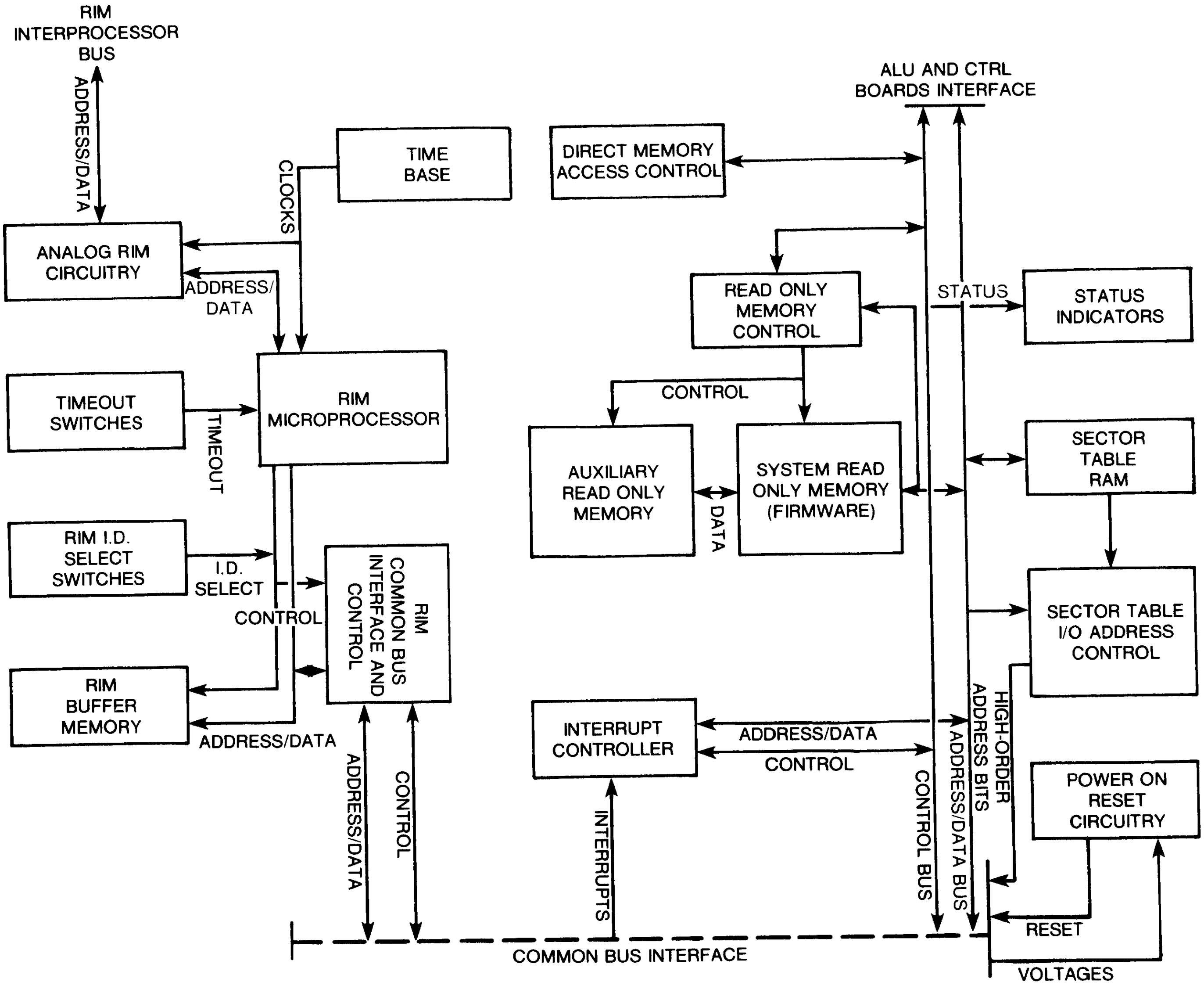


Figure 5-2: CP/RIM Board

The recommended method to determine the presence of a CP/RIM II from software involves using the LED read-back feature. If the LED values are changed in the Output Indicator Port but no change is registered in the Input Indicator Port, then a CP/RIM I is present. If the programmed change is registered, a CP/RIM II is present. See Section 5.9 for a complete description of the CP/RIM Indicator Port.

5.2 Registers

5.2.1 User Registers

The 16 user-accessible registers are implemented as two banks of eight, referred to as the 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, and 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.

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.

5.2.3 System Status Register

The system status is an eight-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 set an interrupt trap.

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

Bit 5: User Mode. When equal to 1, execution of a priviledged instruction code causes 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 and flags are selected; otherwise, alpha registers and flags are selected.

Bits 3-Ø: 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 are always set to the same value, so that Millisecond and Vector Interrupts are always enabled and disabled together. The system status is read back by the LAS instruction and is also placed on the stack after most interrupts.

5.2.4 System Control Register

The system control register is an eight-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 the system data sector table on data read or write memory cycles.

Bit 6: Enable User Segments. When equal to 1, it selects the 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 Ø1ØØØØØ to Ø137777 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 Ø1ØØØØØ to Ø137777 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 Ø in the load data, only bits Ø and 1 are loaded. Bit 2 is always read back as a 1.

Bits 1-Ø: 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:

ØØ	System Instruction Sector Table
Ø1	System Data Sector Table
1Ø	User Instruction Sector Table
11	User Data Sector Table

All control register bits are cleared to zero after the system is reset. The System Control register is read back by the LAK instruction.

5.2.5 Base Register

The Base Register is an eight-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 the logical address is in the range from Ø1ØØØØØ to Ø137777.

5.3 Sector Tables

The 86ØØ 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 and the contents of the control register.

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) or write protected (or both).

Gaining access to the MSB or LSB of each sector table word entry on the CP/RIM II is controlled by a bit in the CP/RIM Indicator Port (see section 5.9). This bit is set by writing a Ø (for LSB) or a 1 (for MSB) to base page I/O address ØØ5. The state of this select function may be read back in the corresponding bit location from the CP/RIM Indicator Port located at base page I/O address ØØ4. The sector table instructions act only on the selected half of each sector table entry, but always access the LSB in the case of the CP/RIM I. The format of the LSB and MSB of a sector table entry are outlined on the following page.

Sector Table
Least Significant Byte (LSB)
(CP/RIM I and II)

```
-----------------------------------------
| B7 | B6 | B5 | B4 | B3 | B2 | B1 | BØ |
-----------------------------------------
  |    |    |    |    |    |    |    |____ADDR Bit 16
  |    |    |    |    |    |    |_________ADDR Bit 17
  |    |    |    |    |    |______________Access Enable
  |    |    |    |    |___________________Write Enable
  |    |    |    |________________________ADDR Bit 12
  |    |    |_____________________________ADDR Bit 13
  |    |__________________________________ADDR Bit 14
  |_______________________________________ADDR Bit 15
```

Sector Table
Most Significant Byte (MSB)
(CP/RIM II only)

```
-----------------------------------------
| B7 | B6 | B5 | B4 | B3 | B2 | B1 | BØ |
-----------------------------------------
  |    |    |    |    |    |    |    |____ADDR Bit 18
  |    |    |    |    |    |    |_________ADDR Bit 19
  |    |    |    |    |    |______________ADDR Bit 2Ø
  |    |    |    |    |___________________ADDR Bit 21
  |    |    |    |________________________not used -|
  |    |    |_____________________________not used  |-|
  |    |__________________________________not used  | |
  |_______________________________________not used -| |
                                                      |
                                                      |
                                Indeterminate on read-back
```

5.4 Address Generation

The processor transforms a 16-bit logical address into an 18-bit physical address on CP/RIM I (22 bits on the CP/RIM II) through the use of the sector tables.

The sector tables consist of high-speed memory which allows the based logical address to be transformed into a physical address (entries in the sector tables are detailed in Section 5.3). Each entry is protected by a parity bit that is generated when the sector tables are written to and checked with every memory access. Errors in the comparison produce a sector table parity error that provides a system-level interrupt to the processor.

Addresses are generated based on the type of cycle (Input/Output or Memory) that the processor is to perform.

I/O cycles and system ROM accesses use the the 16 unsectored address/data lines to form the address. Memory cycles use one of the four sector tables and map address lines A12 to A15 through the sector tables to produce the physical address.

5.5 Stack

The processor has a 32-entry stack that is located in main memory and is addressed 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.

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 can be addressed 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 86ØØ 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 predefined address in system ROM. The format of the information pushed on the stack is shown in Figure 5-3.

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

```
 All Interrupts                  Error Interrupts
 Except Error     Higher Memory
                    Addresses
                                               ---|
                               ______________     |
                               |error address|    |
                               |     MSB     |    |
                               |-------------|    |--|
                               |error address|    |  |
                               |     LSB     |    |  |
 _______________               |-------------|-----  |
|PC -- MSB      |              |PC -- MSB    |       |
|---------------|              |-------------|       |
|PC -- LSB      |              |PC -- LSB    |       |
|---------------|              |-------------|       |
|condition code |              |error status |       |
|     save      |              |  code       |       |
|---------------|              |-------------|       |
|system status  |stack pointer |system status|       |
----------------after interrupt---------------       |
                                                 Present
                                     except for Unas-
                                     signed and Privi-
                                     leged Instruction
                                     Violation
```

Figure 5-3: Stack Information After Interrupt

5.7.1 Millisecond Interrupt

The 86ØØ maintains a timer that runs continuously and times out every 1ØØØ 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 is taken only if the Millisecond Interrupt Enable bit in the system status is a one and the processor finishes executing an instruction. After the interrupt trap occurs, instruction execution resumes at Ø17ØØØØ 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 86ØØ, 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 5-3. After the error interrupt trap occurs, instruction execution resumes at Ø17ØØØ4 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 Section 12.11 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 privileged 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 5-4). 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                                 Lower Memory
Table Base Address                                 Addresses

               + ØØØ  Common Bus Interrupt Ø

               + ØØ4  Common Bus Interrupt 1

               + Ø1Ø  Common Bus Interrupt 2

               + Ø14  Common Bus Interrupt 3

               + Ø2Ø  Common Bus Interrupt 4

               + Ø24  Common Bus Interrupt 5

               + Ø3Ø  Common Bus Interrupt 6

               + Ø34  Common Bus Interrupt 7
                                                  Higher Memory
                                                    Addresses
```

Figure 5-4: 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 86ØØ, the following initialization sequence should be used:

```
[VTABLE (lsb).AND.Ø34Ø].OR.PICIW1---->CBPICØ
VTABLE (msb)------------------------->CBPIC1     I/O port
                                                 address
PICIW4------------------------------->CBPIC1     I/O port
                                                 address
```

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

```
     CBPICØ     EQU  ØØØØ     I/O Port Address for PIC,
                              Register Ø

     CBPIC1     EQU  ØØØ1     I/O Port Address for PIC,
                              Register 1

     PICEOI     EQU  ØØ4Ø     End of Interrupt Code

     PICIW1     EQU  ØØ37     PIC INIT WORD #1

     PICIW4     EQU  ØØ14     PIC INIT WORD #4
```

PIC interrupt masking 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 5-5). Each bit in the mask byte inhibits the corresponding interrupt level when set to a one and enables the interrupt when set to zero.

I7	I6	I5	I4	I3	I2	I1	IØ

Ix = Mask bit for Interrupt level x

Figure 5-5: 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 routine (normally by an IRET instruction), the PIC must be cleared by sending End-of-Interrupt as follows:

PICEOI------>CBPICØ (I/O port address)

5.8 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.

5.9 CP/RIM Indicator Port

5.9.1 Input Indicator Port

The input indicator port, located at I/O address ØØ4, contains status information pertaining to CP/RIM operation. Bit definitions are as follows (an LED is a light-emitting diode):

```
-----------------------------------------
| B7 | B6 | B5 | B4 | B3 | B2 | B1 | BØ |
-----------------------------------------
  |    |    |    |    |    |    |    |____Sector Table
  |    |    |    |    |    |    |         Select Readback;
  |    |    |    |    |    |    |         1=MSB, Ø=LSB
  |    |    |    |    |    |    |
  |    |    |    |    |    |    |_________Power Fail Impend-
  |    |    |____|____|____|              ing; 1=Alarm
  |    |              |
  |    |              |___________________CP/RIM LEDs;
  |    |                                  Ø=a lighted LED
  |____|
     |____________________________________Reserved; Read
                                          back in indeter-
                                          minate state
```

These bits are set to a binary 1 when true, and are cleared by CBRESET/. A zero represents a lit LED for bits 2 through 5. Bit 2 on the CP/RIM I reflects the diagnostic mode jumper position, and is unalterable. This bit is a guaranteed level when readback on the CP/RIM I board, but it represents the status of the least significant LED on the CP/RIM II board. Bit 1 (alarm) is the only bit available on the CP/RIM I.

5.9.2 Output Indicator Port

In addition to the input status port, a second port located at base page address ØØ3 (4 bits) is used to turn on and off the CP/RIM card-edge LEDs. This port consists of four LEDs that display one of 16 system status codes. Writing a binary Ø to any of these bits (Ø-3) turns on the corresponding LED. The state of each LED is available in the input status port bits 2 through 5 (on the CP/RIM II only). The LEDs are initialized to the on state by POR, and are used by system firmware to show POR diagnostic information.

5.1Ø 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 Ø17ØØØØ-Ø177777. See Appendix A of this manual for a complete description of system ROM.

Auxiliary ROM may be up to 12K bytes, starting at Ø1ØØØØØ. 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.

PART 6
RESOURCE INTERFACE MODULE

6.1 General

The Resource Interface Module (RIM) provides the standard interface to a DATAPOINT ARC local network. All 8600Cs have a RIM on the CP/RIM board (see figure 5-2). The RIM on the CP/RIM board is always RIM0.

A RIM is addressed using the Status/Mask Register and the Command/MAR Register, which are located at the I/O addresses specified in Figure 6-1.

The RIM buffer starts at I/O address 074000 and extends to 077777. It should be noted that the RIM buffer on a CP/RIM in an 8600 system is located at I/O address 74000 through 74377, 75000 through 75377, 76000 through 76377, and 77000 through 77377. The RIM ID is selected by an 8-bit switch on the CP/RIM.

I/O addresses 040 and 041 are used to enable or disable the various RIM buffers which can exist in a system; only one RIM buffer in a system can be enabled at a time. Enabling one RIM buffer automatically disables all other RIM buffers in a system; any write to I/O address 040 enables the buffer of RIM 0.

RIM I/O Allocation

Address	Function
Ø4Ø	RIM Ø Buffer Enable
Ø42	RIM Ø Status/Mask Register
Ø43	RIM Ø Command/MAR Register
Ø74ØØØ-Ø74777	RIM Buffer on enabled RIM

Figure 6-1: RIM I/O Allocation

6.2 Register Description

6.2.1 Read Status Register

Execution of this command places the contents of the status register on the data bus (AD7 - ADØ) during the read portion of the processor's read cycle. The RIM register contents are defined as follows:

BIT 7 Receiver inhibited (RI)
This bit, if set high, indicates that a packet has been deposited into the RAM buffer page nn as specified by the last ENABLE RECEIVE TO PAGE nn command. The setting of this bit can cause an interrupt through INTR if enabled during a WRITE INTERRUPT MASK command. No messages are received until an ENABLE RECEIVE TO PAGE nn command is issued. After any message is received, the receiver is automatically inhibited by setting this bit to a logic one.

BIT 6 Extended Timeout Status 2 (ETS2)
This bit reflects the current logic value tied to the ET2 input pin (pin 1).

BIT 5 Extended Timeout Status 1 (ETS1)
This bit reflects the current logic value tied to the ET1 input pin (pin 3).

BIT 4 Power On Reset (POR)
This bit, if set high, indicates that the RIM has received an active signal on the POR input (pin 4Ø). The setting of this bit causes a nonmaskable interrupt through INTR.

BIT 3 (TEST)
This bit is intended for test and diagnostic purposes. It is a logic zero under any normal operating conditions.

BIT 2 Reconfiguration (RECON)
This bit, if set high, indicates that the reconfiguration timer has timed out because the RX input was idle for 78.2 microseconds. The setting of this bit can cause an interrupt through INTR if enabled by the WRITE INTERRUPT MASK command. The bit is reset low during a CLEAR FLAGS command.

BIT 1 Transmit Message Acknowledged (TMA)
When this bit is set high, it indicates that the packet transmitted as a result of an ENABLE TRANSMIT FROM PAGE nn command has been positively acknowledged. This bit should be considered valid only after the TA bit (bit Ø) is set. Broadcast messages are never acknowledged.

BIT Ø Transmitter Available (TA)
This bit, when set high, indicates that the transmitter is available for transmitting. This bit is set at the conclusion of an ENABLE TRANSMIT FROM PAGE nn command or upon the execution of a DISABLE TRANSMITTER command. The setting of this bit can cause an interrup through INTR if enabled by the WRITE INTERRUP MASK command.

After power-up, the status register assumes the following state:

BIT 7 (RI) set to logic "1"
BIT 6 (ETS2) not affected
BIT 5 (ETS1) not affected
BIT 4 (POR) set to a logic "1"
BIT 3 (TEST) set to a logic "Ø"
BIT 2 (RECON) set to a logic "Ø"
BIT 1 (TMA) set to a logic "Ø"
BIT Ø (TA) set to a logic "1"

6.2.2 Write Interrupt Mask

The RIM is capable of generating an interrupt signal when certain status bits become true. A write to the MASK register specifies which status bits can generate the interrupt. The bit positions in the MASK register are in the same position as their corresponding status bits in the STATUS register with a logic one in a bit position enabling the corresponding interrupt. The setting of the TMA, EST1, and EST2 status bits never causes interrupts. The POR status bit causes a nonmaskable interrupt regardless of the value of the corresponding MASK register bit. The MASK register takes on the following bit definition:

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT Ø
Receive Inhibit	XXX	XXX	XXX	XXX	RECON TIMER	XXX	Trans. Avail.

The three maskable status bits are anded with their respective mask bits, and the results, along with the POR status bit, are or'ed to produce the processor interrupt signal INTR. This signal returns to its inactive low state when the interrupting status bit is reset to a logical zero or when the corresponding bit the the MASK register is reset to a logical zero Ø. To clear an interrupt generated as a result of a Power On Reset or Reconfiguration occurrence, the CLEAR FLAGS command should be used. To clear an interrupt generated as a result of a completed transmission (TA) or a completed reception (RI), the corresponding masks' bits should be reset to logical zeroes.

6.2.3 Write RIM Commands

Execution of the following commands is initiated by performing a processor I/O write with the written data defining the following commands:

ØØØØØØØØ Reserved for future use.

ØØØØØØØ1 DISABLE TRANSMITTER
This command cancels any pending transmit command (transmission that has not yet started) when the RIM next receives the token. This command sets the TA (Transmitter Available) status bit when the token is received.

ØØØØØØ1Ø DISABLE RECEIVER

This command cancels any pending receive command. If the RIM is not yet receiving a packet, the RI (Receiver Inhibited) bit is set the next time the token is received. If packet reception is already underway, reception runs to its normal conclusion.

ØØØnnØ11 ENABLE TRANSMIT FROM PAGE nn

This command prepares the RIM to begin a transmit sequence from RAM buffer page nn the next time it receives the token. When this command is loaded, the TA and TMA bits are set to a logical one; the TA bit is set to a logical one upon completion of the transmit sequence. The TMA bit will have been set by this time if the RIM has received an acknowledgement from the destination RIM. This acknowledgement is strictly hardware level which is sent by the receiving RIM before its controlling processor is even aware of message reception. It is also possible for this acknowledgement to get lost due to line errors and other causes. This implies that the TMA bit is not a guarantee of proper destination reception.

bØØnn1ØØ ENABLE RECEIVE TO PAGE nn

This command enables the RIM to receive data packets into RAM buffer page nn and sets the RI status bit to a logical zero. If "b" is a logical one, the RIM also receives broadcast transmissions (a broadcast transmission is a transmission to ID zero). The RI status bit is set to a logical one upon the successful receipt of a message.

ØØØØc1Ø1 DEFINE CONFIGURATION

If c is a logical one, the RIM is configured with a 2K external RAM buffer and short as well as long packets can be handled. If c is a logical zero, the RIM is configured with a 1K RAM buffer and only short packets (less than 256 bytes) can be handled.

ØØØrp11Ø CLEAR FLAGS

If P is a logical one, the POR status flag is cleared.
If R is a logical one, the RECON status flag is cleared.

NOTE: All other combinations of written data are not permitted and can result in incorrect chip or network operation (or both).

6.3 Operation

6.3.1 Transmit

During a transmit sequence, the RIM fetches data from the Transmit Buffer, a 256-byte segment of the RAM buffer. During a receive sequence, the RIM stores data in the receive buffer, also a 256-byte segment of the RAM buffer. The processor I/O command which enables either the RIM receiver or the RIM transmitter also initializes the respective buffer page register. The format of the buffer is shown in figure 6-2 (N is the data packet length, SID is the source ID, and DID is the destination ID).

Buffer Offset	Format
0	SID
1	DID
2	Count=256-N
	Not Used
Count	Data Byte 1
	Data Byte 2
	* * *
	Data Byte N-1
255	Data Byte N
511	Not Used

Figure 6-2: RAM Buffer Packet Configuration

PART 7
MICROBUS INTERFACE MODULE (MIFM)

7.1 General

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 maximum block data transfer rate is 4ØØ kilobytes per second, and the maximum block length is 256 bytes. A block diagram of the MIFM is shown in Figure 7-1.

For detailed information on ports, registers, and programming, see the respective product specifications for the 931Ø Disk Drive (Document No. 6Ø876), the 9315 Disk Drive (Document No. 61382), and the 9324/9325 Disk Drive (Document No. 616Ø9) as well as for the 14Ø1 Diskette Drive (Document No. 61Ø31).

7.2 The Microbus

7.2.1 Microbus Signals

The Microbus provides four bits of peripheral device address (AØ-A3). Four command lines provide unique commands for each of two transfer strobes (USTB 1 and 2); a third strobe (IACK) is used to acknowledge an interrupt request. The contents of the command and address lines are 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 bidirectional lines are used to transfer data to and from the peripheral device under control of the processor and the MIFM (the drivers are of the 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 so commanded by the MIFM.

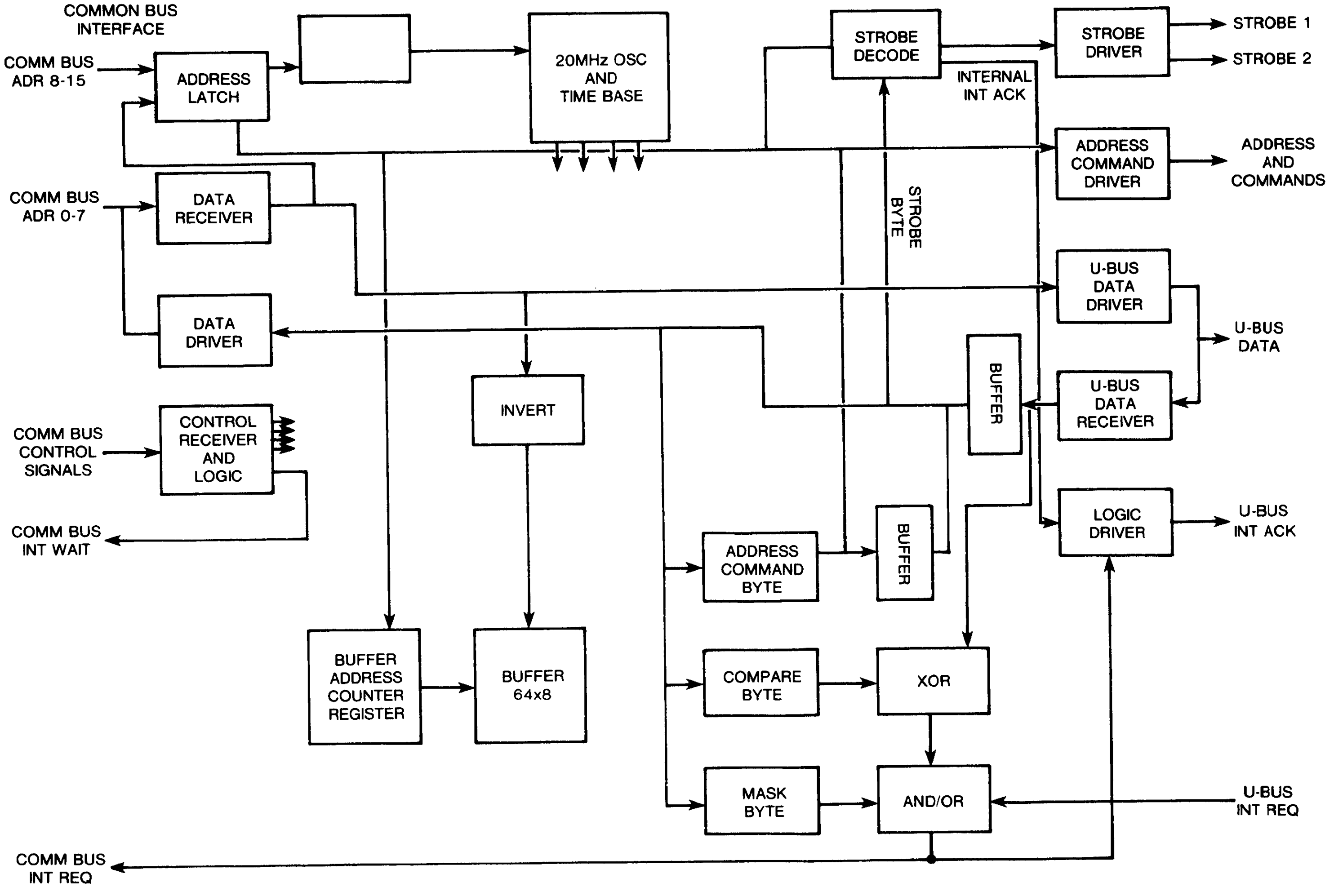


Figure 7-1: Microbus Interface Module (MIFM)

7.2.2 Microbus Timing

The address, command, and data lines are stable for at least 1ØØ ns before the leading edge of the Transfer Strobe, and remain so for at least 1ØØ ns after the trailing edge (the transfer strobes are a minimum of 4ØØ ns wide). During a Microbus input cycle, the peripheral device puts stable data on the data lines at least 1ØØ ns prior to the end of the Transfer Strobe and holds it for 1ØØ 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 14ØØ ns, which is required when fifteen peripheral devices that use the Microbus interrupt line are present on the bus.

7.2.3 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 placed 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 data transfer commands at the rate of one every two microseconds. Commands requiring longer than this are associated with a busy status bit.

7.2.4 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 highest-priority 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. Priority is based on a peripheral's physical location in the Microbus daisy-chain; devices nearest the processor have the highest priority.

7.3 Software Interface

All data transfer between the processor and microbus peripherals is under software control utilizing six I/O instructions: CBIN, CBOUT, BLKIN, BLKOUT, MBLKIN, and MBLKOUT.

In addition to the normal processor-peripheral command and data sequences, the MIFM provides an automatic peripheral polling mode of operation. In this mode, a sequence of status read commands is sent out over the Microbus. Status is sampled and checked (exclusive-OR'ed and masked) to determine the necessity for generating a processor interrupt. The polling mode is general (that is, no peripheral device dependencies) and entirely software programmable. The primary purpose of the polling logic is to provide a processor interrupt capability for those peripheral devices that do not implement interrupts.

7.3.1 MIFM Instruction Coding

This section defines the coding of the information that is loaded into the D, E, and A (or specified) registers when executing the CBIN, CBOUT, MBLKIN, MBLKOUT, BLKIN, and BLKOUT instructions. These same instructions are used to gain access to the MIFM Poll Buffer and enable and disable the polling operation.

The high-order five bits of the D register are always zero when looking to the MIFM, reflecting the assigned I/O address space. Bits 8 and 9 choose one of four principal types of operations: IACK cycle, STROBE 1 cycle, STROBE 2 cycle, or one of the several other board-related functions.

The E register may contain Microbus information in the case of the Strobe cycles, or bits to be decoded further to determine the board operation. The A register (or any other specified register) contains data to be written to or from the MIFM. Refer to Table 7-1 MIFM Instruction Coding for details.

7.3.2 MIFM Status Byte

The status register on the MIFM is:

7	6	5	4	3	2	1	Ø
Ø	Ø	Ø	Ø	Ø	1	MI	PE

PE
If equal to one, then polling is enabled on the MIFM; powers up disabled.

MI
If equal to one, then a polling interrupt is pending on the MIFM; powers up as a zero.

7.3.3 Polling Function Organization and Operation

The MIFM polling function is organized around a 64-byte Random Access Memory which provides the needed storage for address, command, strobe, compare, and mask bytes. Access to the buffer is controlled by a 6-bit address counter and register; the buffer is loaded from the processor under software control and can also be read by the processor for diagnostic and other purposes. When polling is enabled, bytes are read from the buffer in groups of four and stored in the address, command, strobe, compare, and mask registers. Polling, when enabled, is intitiated by the start of a processor memory cycle and is controlled by an internal time base generator. During a polling cycle, the address and command byte is sent to the peripheral device with the appropriate transfer strobe. The status read-back is exclusive OR'ed with the compare byte and the result is AND'ed with the mask byte. A true result pulls the CBIREQØ line low to send an interrupt request to the processor. This interrupt request is OR'ed with the IREQ line from the Microbus. The interrupt acknowledge logic for the polling function is daisy-chained to the IACK line in the Microbus and therefore has a higher priority than the IACK in any peripheral device.

REGISTERS:

D								E								A								DEFINITION:
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	
0	0	0	0	0	1																			
						1	1	x	x	x	x	x	x	x	x	d7	d6	d5	d4	d3	d2	d1	d0	: IACK STROBE (CBIN) (CBOUT is meaningless)
						1	0	c3	c2	c1	c0	a3	a2	a1	a0	/	/	/	/	/	/	/	/	: STROBE 2 (CBIN, CBOUT, MBLKIN or MBLKOUT)
						0	1	c3	c2	c1	c0	a3	a2	a1	a0	/	/	/	/	/	/	/	/	: STROBE 1 (CBIN, CBOUT, MBLKIN or MBLKOUT)
						0	0																	: MIFM POLL ACCESS
								0	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	: DISABLE POLLING (CBIN)
								0	0	-	-	-	-	0	0	x	x	x	x	x	x	x	x	: ENABLE POLLING (CBOUT)
								0	1	x	x	x	x	x	x	b7	b6	b5	b4	b3	b2	i	e	: MIFM ID/STATUS BYTE (CBIN,CBOUT; see Section 5.3)
								1	0							d7	d6	d5	d4	d3	d2	d1	d0	: READ POLL BUFFER (CBIN) |(Resets poll sequence to
																								| buffer address 0 at end
								1	0							D7	D6	D5	D4	D3	D2	D1	D0	: WRITE POLL BUFFER (CBOUT) | of instruction cycle
										P5	P4	P3	P2	P1	P0									: POLL BUFFER BYTE ADDRESS (64 bytes, see below)
										0	0	0	0	0	0	c3	c2	c1	c0	a3	a2	a1	a0	: ADD/CND BYTE, FIRST DEVICE
										0	0	0	0	0	1	0	0	0	0	0	0	s2	s1	: STROBE BYTE, FIRST DEVICE
										0	0	0	0	1	0	r7	r6	r5	r4	r3	r2	r1	r0	: COMPARE BYTE, FIRST DEVICE
										0	0	0	0	1	1	m7	m6	m5	m4	m3	m2	m1	m0	: MASK BYTE, FIRST DEVICE
										0	0	0	1	0	0	c3	c2	c1	c0	a3	a2	a1	a0	: ADD/CMD BYTE, SECOND DEVICE
										:	:	:	:	:	:	:	:	:	:	:	:	:	:	:
										1	1	1	1	1	1	m7	m6	m5	m4	m3	m2	m1	m0	: MASK BYTE, 16th DEVICE

NOTES:

x :Indicates a don't care bit condition

dx :Indicates a data transfer from the peripheral to the processor **

Dx :Indicates a data transfer from the processor to the peripheral **

/ :Indicates a data transfer in either direction as function of CBIN, CBOUT, MBLKIN, MBLKOUT, and meaning associated with c3-c0, a3-a0

- :Indicates Poll buffer byte address for start of polling sequence

cx :Micro I/O Bus command code **

ax :Micro I/O Bus device address code **

sx :Determines strobe generated in polling s2 = s1 = 0 indicates no polling for specified device address

rx :Bits in compare byte which are XORed with device status byte **

mx :Bits in mask byte which are ANDed with result of XOR operation **

bx :ID byte bits

i :Interrupt request status bit

e :Polling enabled status bit

** :Refer to appendices or other Peripheral Device specification documents for detailed code definitions.

Table 7-1: MIFM Instruction Coding

In the event the polling logic generates an interrupt, polling is halted until the processor issues an IACK strobe to clear the interrupt and services the request, followed by an Enable Polling command to re-enable the polling. In other words, the 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 or read individually or they may be written or read in blocks; the MIFM buffer must be initialized before enabling polling.

7.3.4 Polling Function Programming Considerations

Certain Microbus peripheral devices provide a status latch. In order to successfully poll the status in these devices, it is necessary to issue a two-command sequence -- fetch status (update the latch) and input status. This requires two polling cycles to check peripheral device status (that is, two groups of four bytes out of the 16 groups available). The eight bytes for this case would thus be:

n:	ADD/CMD	Device address and fetch status command
N+1	Strobe	As required by device
N+2	Compare	Don't care
N+3	Mask	All bits zero
N+4	ADD/CMD	Device address and input status command
N+5	Strobe	As required by device
N+6	Compare	As specified by the program
N+7	Mask	As specified by the program

Since the access to the polling buffer from the processor uses the E register as the buffer byte address, it is necessary to use the BLKIN and BLKOUT instructions instead of the MBLKIN and MBLKOUT instructions to perform a block transfer between the processor and the polling buffer. The software has access to any particular byte with the CBIN and CBOUT instructions.

There is no provision for sending a data byte to the peripheral device in the polling mode of operation.

PART 8
PERIPHERAL INPUT/OUTPUT MODULE

8.1 General

The Peripheral Input/Output Module (PIO) allows the 8600 to communicate with externally attached peripherals using the Peripheral I/O bus, which is a high-performance general-purpose serial bus that uses a 12-bit character format at 13.055 Mbits/second. Data transfers may be accomplished on a character-by-character basis or through DMA (direct memory access). When data is transferred through DMA, the PIO becomes the master of the 8600 Common Bus and addresses main memory on a word basis. A block diagram of the PIO module is given in Figure 8-1.

The PIO interface supports the DATAPOINT 9301 series compact disk drives and extensions. See the 9301 Product Specification (Document No. 61173) for detailed information on ports, registers, and programming.

8.2 Peripheral Input/Output Bus

The external Peripheral I/O 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 50 feet (approximately 16 meters). 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.

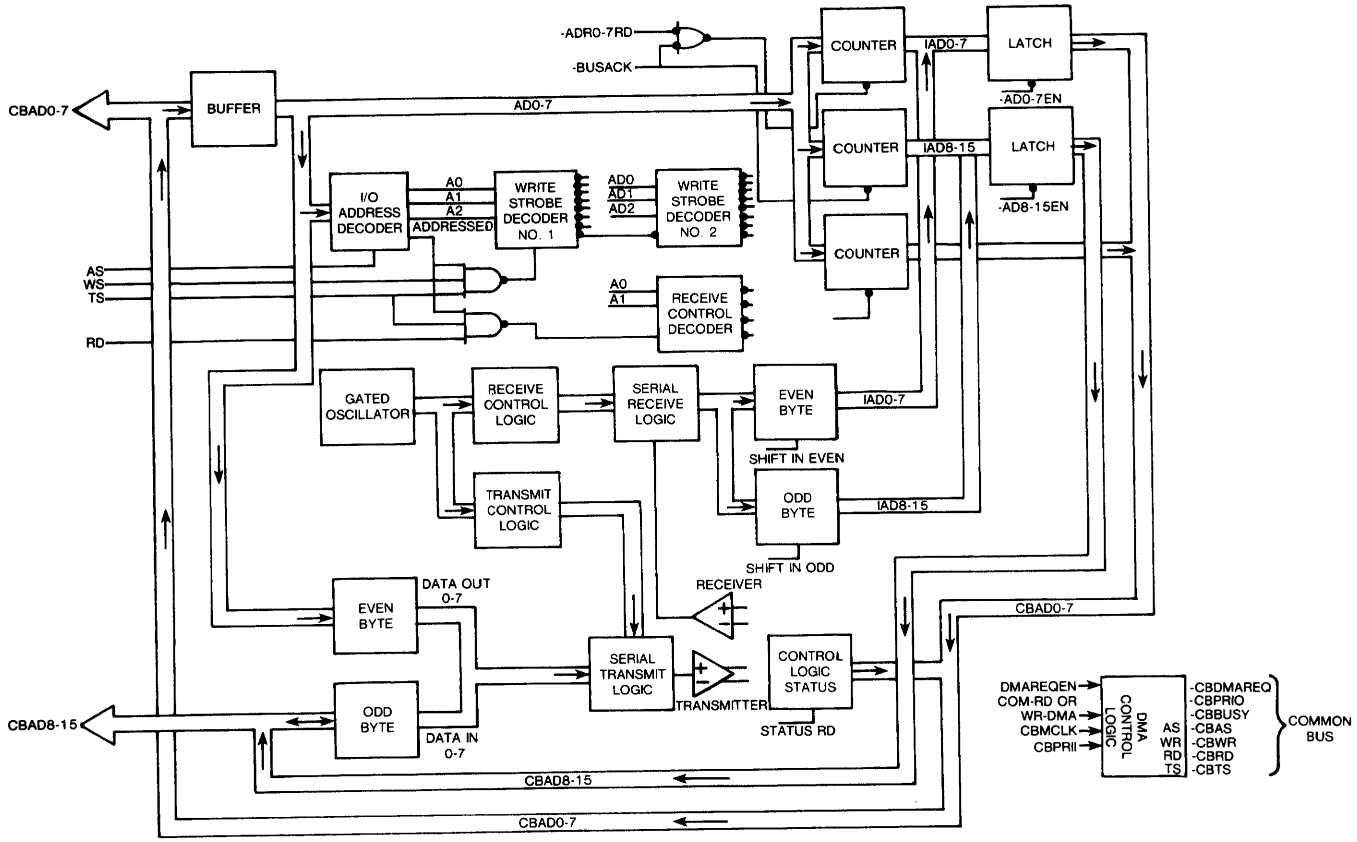


Figure 8-1: Peripheral Input/Output Module

Peripherals are linked to the bus in a multi-drop fashion, and physical connection is by 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 communications from the processor to the peripherals, while the second wire pair is used for transfers 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.Ø55 Mbits per second, which is compatible with the transfer rate of the DATAPOINT 93Ø1 disk. This corresponds to a maximum continuous transfer rate of 1.Ø88 Mbytes per second over the bus. A single parity bit is included in the 12-bit character format for error control. Data is encoded in the 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 is used as a handshake line, which allows the matching of the transfer rate of the processor to any peripheral on a character-by-character basis.

8.2.1 Control Characters

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

8.2.2 Addressing

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

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

Four bits of addressing are provided for addressing 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 must have a unique address before the system is powered up. The processor determines the peripherals that are present by polling all addresses and reading the interface type status byte during power-up and initialization.

8.3 PIO Module 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 loads 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.

8.4 PIO Module 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 is driven to a Mark, allowing another byte or word of data, if any, to be transmitted.

PART 9
PARALLEL BUS ADAPTER

9.1 General

The Parallel Bus Adapter enables the Common Bus master to communicate with external devices attached using the Datapoint parallel bus (see figure 9-1). The Parallel Bus Adapter is linked to the parallel bus to allow byte data or command transfer rates at a maximum of 125 KHz, 8Ø microseconds per byte.

The Parallel Bus Adapter is addressed through extended Common Bus I/O space on a character-by-character basis. The parallel bus I/O commands are emulated in 86ØØ microcode; the commands EX BEEP and EX CLICK are emulated in firmware.

An auxiliary I/O power supply (Model Code 9Ø22) must be used for devices that draw all their power from the Parallel Bus.

9.2 Compatibility

The Parallel Bus Adapter communicates with the Common Bus master over the 86ØØ's Common Bus. Specific devices supported by the parallel bus I/O instruction set may be connected to the 86ØØ's Parallel Bus, providing that the power supply constraints are met (see Section 9.3).

9.3 Electrical Characteristics

The Parallel Bus Adapter implements all the signals required to support the parallel bus. The receiver and driver circuit characteristics are compatible with those given in the Datapoint 6ØØØ and 66ØØ Hardware Reference Manual (Document No. 6Ø853).

The AOUTØ-AOUT8 output drivers and the AINØ-AIN8 input receivers are shown in Figure 9-2. The output strobe drivers are also shown.

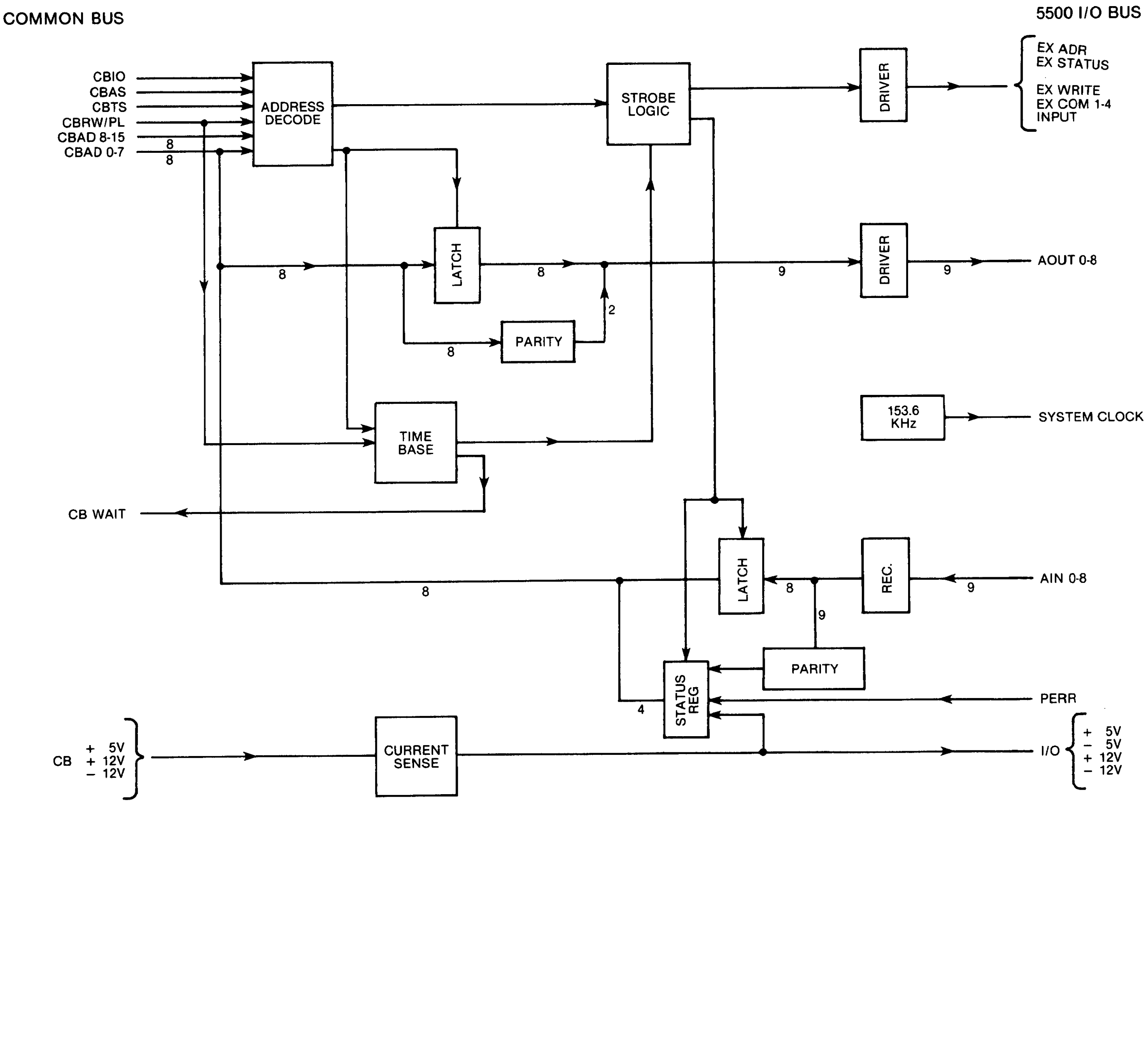


Figure 9-1: Parallel Bus Block Diagram

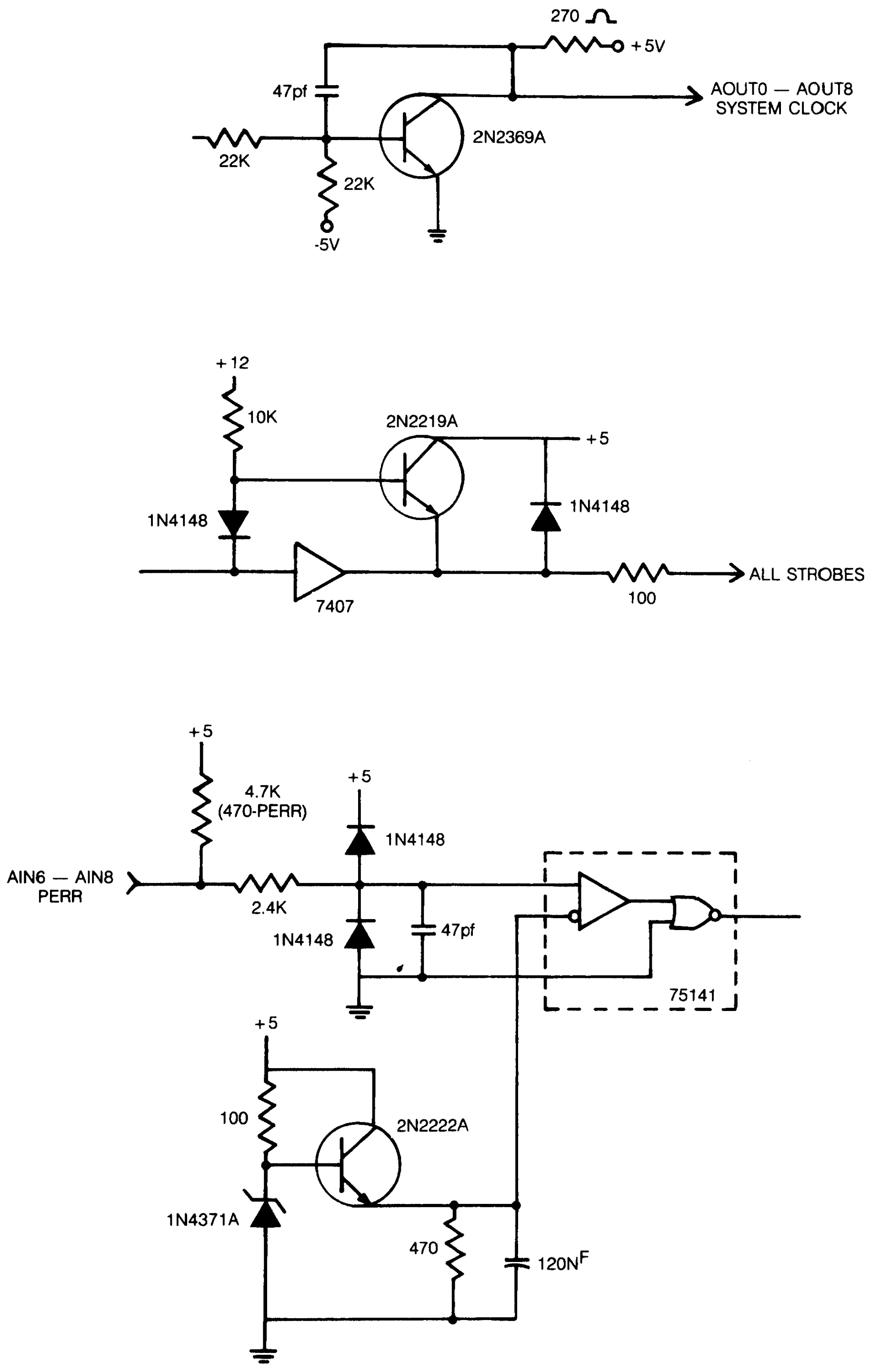


Figure 9-2: Parallel Bus Drivers/Receivers

The Parallel Bus Adapter supplies limited power for external devices on the parallel bus. Due to fuse and cable resistance and the loss in filter inductors present on many external parallel bus devices, the Parallel Bus Adapter does not power devices that draw all their power from the parallel bus. The Parallel Bus Adapter draws +5V, +12V, and -12V from the Common Bus and supplies +5V, -5V, +12V, and -12V to the Parallel Bus Adapter. The +24V line is not supplied by the board. The I/O power lines are fused as follows:

+5V	2.Ø A
+12V	25Ø mA
-5V, -12V	25Ø mA

An auxiliary I/O power supply (Model Code 9Ø22) must be used for devices that draw all their power from the parallel bus. The board does, however, supply sufficient I/O power to operate a maximum of eight devices that power only their output driver circuits from +5V. The +12V, -12V, and -5V lines are provided for sensing purposes only, or to provide negative bias for the AIN driver logic.

The I/O power fuses are located on the top of the board, with a red "fuse blown" LED located next to each fuse. A blown fuse also sets a bit in the status register.

9.4 I/O Address Decode

The Parallel Bus Adapter is addressed through extended common bus I/O space by specially microcoded instructions (the address structure is shown in Figure 9-3). The base address of the board is determined by CBAD 15-11; since only one board per system is allowed, these five bits are fixed. Address bits 1Ø-8 are decoded as the type of parallel bus command to be executed. The lower eight bits are used as data during parallel bus write cycles and are ignored during parallel bus read cycles. This addressing scheme uses 2K of extended Common Bus I/O space.

The I/O address decoded uses the Common Bus advanced read/write signal to distinguish between I/O reads and writes.

CBAD Bits

	15	14	13	12	11	1Ø	9	8	7	6	5	4	3	2	1	Ø	FUNCTION
READ	Ø	Ø	Ø	Ø	1	Ø	Ø	Ø	X	X	X	X	X	X	X	X	IN
	Ø	Ø	Ø	Ø	1	Ø	Ø	1	X	X	X	X	X	X	X	X	PIN (or MIN)
	Ø	Ø	Ø	Ø	1	Ø	1	Ø	X	X	X	X	X	X	X	X	not used
	Ø	Ø	Ø	Ø	1	Ø	1	1	X	X	X	X	X	X	X	X	not used
	Ø	Ø	Ø	Ø	1	1	Ø	Ø	X	X	X	X	X	X	X	X	not used
	Ø	Ø	Ø	Ø	1	1	Ø	1	X	X	X	X	X	X	X	X	not used
	Ø	Ø	Ø	Ø	1	1	1	Ø	X	X	X	X	X	X	X	X	not used
	Ø	Ø	Ø	Ø	1	1	1	1	X	X	X	X	X	X	X	X	not used
WRITE	Ø	Ø	Ø	Ø	1	Ø	Ø	Ø				data					EX ADR
	Ø	Ø	Ø	Ø	1	Ø	Ø	1				data					EX STATUS
	Ø	Ø	Ø	Ø	1	Ø	1	Ø				data					EX DATA
	Ø	Ø	Ø	Ø	1	Ø	1	1				data					EX WRITE
	Ø	Ø	Ø	Ø	1	1	Ø	Ø				data					EX COM1
	Ø	Ø	Ø	Ø	1	1	Ø	1				data					EX COM2
	Ø	Ø	Ø	Ø	1	1	1	Ø				data					EX COM3
	Ø	Ø	Ø	Ø	1	1	1	1				data					EX COM4

Figure 9-3: PBA Address Structure

15	14	13	12	11	1Ø	9	8	7	6	5	4	3	2	1	Ø

```
|<---------BASE---------->|<--Command->|<------data or don't care----->|
          Address
```

I/O Address Code

9.5 Parity

Parity is checked on the parallel bus during I/O reads. The nine DATA IN bits (AINØ-AIN8) are checked for the proper parity on the leading edge of INPUT STROBE. If a parity error occurs during a PIN or a MIN instruction, CBBW/PE (Parity Error) is asserted, causing a nonmaskable interrupt. A parallel bus write cycle generates parity from the eight data Common Bus lines. If the external parallel bus device detects a parity error, it asserts PERR. The board detects PERR and asserts CBBW/PE during any MOUT command.

9.6 Status Register

The Parallel Bus Adapter status register is located at Common Bus address Ø56 (see Table 9-1). The upper five bits of the status register are read-only, high-true bits; the lower three bits of the status register are diagnostic read-write bits. They are cleared on power-up and indicate that the Parallel Bus Adapter is present in the system. Writing a one (1) to these bits allows diagnostics to invert various parity conditions.

B0	Invert CB parity on parallel bus reads
B1	Invert parallel bus PERR
B2	Invert parallel bus input parity
B3	Parallel bus input parity clocked on every input strobe
B4	PERR asserted, bit 4 is set for the duration of PERR, and is reset at the end of a parallel bus cycle
B5	-5V, -12V fuse blown, set until fuse is replaced
B6	+12V fuse blown, set until fuse is replaced
B7	+5V fuse blown, set until fuse is replaced

Table 9-1: Status Register

9.7 Indicators

The Parallel Bus Adapter drives four card-edge LEDs that monitor:

Parallel Bus Adapter addressed	Green
+5 volt fuse blown	Red
+12 volt fuse blown	Red
-5, -12 volt fuse blown	Red

9.8 Parallel Bus Instruction Emulation

The 8600 processor emulates the parallel bus instructions in microcode, with the exception of EXBEEP and EXCLICK, which are handled in firmware. Each instruction has a unique entry point into the microcode, where I/O cycles are initiated to the proper extended I/O address. Peripherals attached to the Parallel Bus Adapter are programmed using the IN/PIN, EX, MIN, and MOUT instructions exclusively. See Section 5.11, Instruction Set, for a detailed description of these instructions.

9.9 I/O Interface

The Parallel Bus Adapter is linked to the 8600 I/O panel using a 50-conductor flat ribbon cable. The cable is connected to the back panel printed circuit board with a mass-terminating 50-pin printed circuit mount connector. The board has a 50-pin printed circuit mounting D-SUB connector to mate to the external Parallel Bus Adapter cable assembly.

PART 10
MULTIPORT COMMUNICATIONS ADAPTER (MPCA)

10.1 General

The Multiport Communications Adapter (MPCA) is a printed circuit board that resides in the 8600 processor's internal card cage. The MPCA houses four serial asynchronous RS-232-C-compatible communications ports. Each port has full-duplex transmit and receive capability; data transfer rates are software programmable from 50 to 19200 baud. The MPCA ports are individually programmable for character lengths and number of stop bits. The MPCA also includes a microprocessor, local memory, four UARTs (Universal Asynchronous Receiver/Transmitter), and baud rate generators. A block diagram of the MPCA is shown in Figure 10-1.

10.2 Microprocessor

The MPCA contains an integral microprocessor which runs at 4 MHz and operates primarily by polling. The polling routine updates the status byte of all ports, and maintains staging buffers and First-In, First-Out (FIFO) buffers. (Note: In the following sections, the MPCA microprocessor is referred to as the Z80 and the 8600 Central Processor is referred to as the CPU.)

10.3 Memory

The MPCA has the capability of addressing 2K X 8 of static RAM and 8K X 8 of ROM. The MPCA microprocessor addresses 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 storage 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.

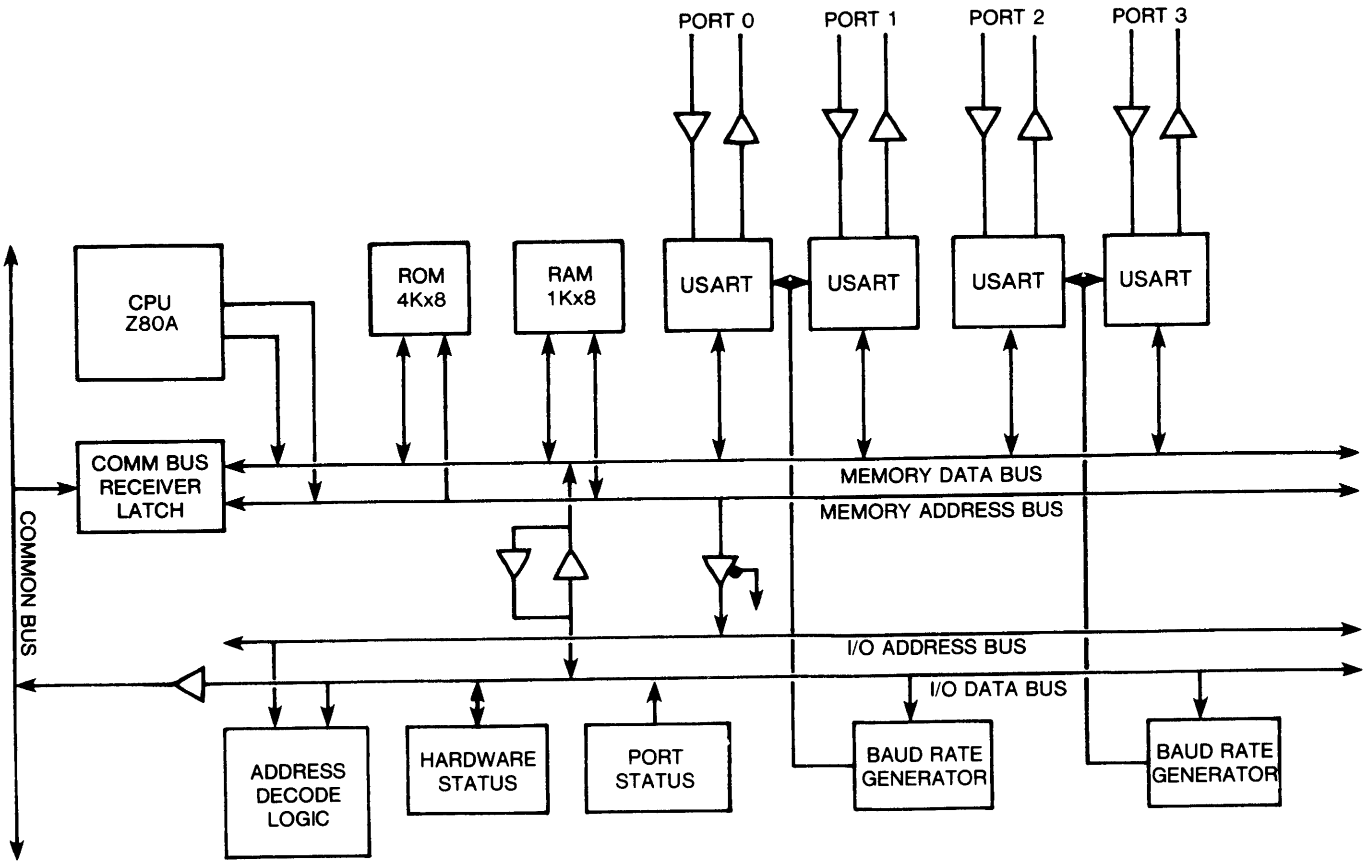


Figure 1Ø-1: Multiport Communications Adapter Board

1Ø.4 UARTs

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

1Ø.5 Baud Rate Generators

Programming of baud rates is done by the Z8Ø upon request by the CPU. Transmit baud rate clocks are generated by external chip baud rate generators; receive baud rates are generated by the UART internal baud rate generator. Any programming difference between the two baud rate generators is transparent to the CPU.

1Ø.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 samples and compares this code to see if it has been selected. Once a board has been selected, it remains 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 issues 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 use byte Ø of the command section in RAM.

1Ø.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 Z8Ø waits until the Interrupt Service command has been written before asserting another interrupt for that port. 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 Z8Ø writes the port service code to the address set aside in on-board RAM, and asserts the interrupt request line.

1Ø.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 command from the CPU. This routine brings the UARTs 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 UART, FIFO, and staging buffer service.

1Ø.9 Diagnostics

MPCA diagnostic code resides in on-board ROM and allows the MPCA microprocessor to test board components. Internal testing is done for the MPCA RAM, ROM, UARTs, various hardware registers, and for the microprocessor itself. Hardware failures are reported to the CPU, and the polling loop is entered if possible. The MPCA has a jumper to put it in continuous diagnostic mode.

PART 11
MULTIFUNCTION COMMUNICATIONS ADAPTER (MFCA)

11.1 General

The Multifunction Communications Adapter (MFCA) is a printed circuit board that resides in the 86ØØ processor's internal card cage. The MFCA provides a means of bidirectional information transfer between the processor and an RS-232-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 MFCA includes a microprocessor, a serial input/output channel, a counter/timer circuit, and local memory with parity. A block diagram of the MFCA is shown in Figure 11-1.

The 86ØØ MFCA inlcudes the following features:

- Synchronous/asynchronous operation, full or half duplex.
- BISYNC, SDLC, HDLC, ADCCP, and GENSYNC protocols.
- Programmable baud rates (11Ø to 19.2K baud)
- Programmable internal or external clock.
- Programmable NRZ or NRZI data.
- Programmable sync characters and stop bits.
- Hardware CRC generation and checking.

11.2 Microprocessor

The MFCA contains an on-board Z8ØA 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 Z8Ø and the 86ØØ Central Processor is referred to as the CPU.)

11.3 Serial Interface

The MFCA contains a Z8ØA 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 and supports the RS-232-C channel.

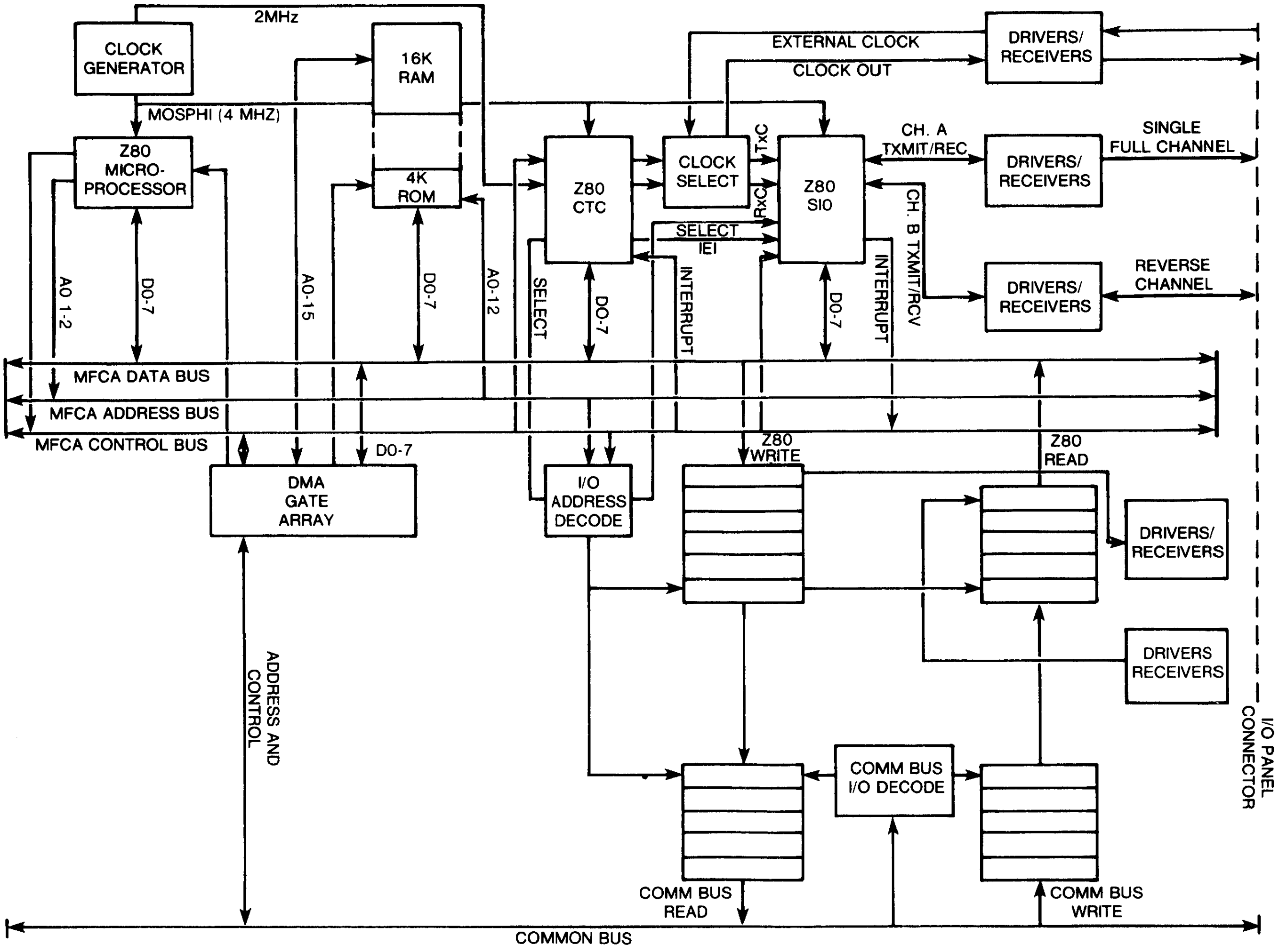


Figure 11-1: Multifunction Communications Adapter Board

11.4 Counter/Timer Circuit

Baud rates are program selected through Z8Ø I/O commands and are provided by a Z8ØA 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 Z8Ø and the internal timing of the CTC. The CTC also acts as a real-time clock and an interrupt controller.

11.5 Memory

The memory space is organized as 16K X 9 RAM, 4K X 8 ROM, and 16K of 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 Ø4ØØØØ in Z8Ø memory space.

The ROM resides at location ØØØØØØ-Ø7777 and contains system firmware and self-test diagnostics. The firmware initializes the MFCA and facilitates the down-loading 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 Z8Ø and begins at address Ø14ØØØØ.

Access to the MFCA RAM is provided only to the Z8Ø 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 Z8Ø and the CPU and are loaded by the CPU with the down-load command and registers. The firmware monitors these registers, accepts the command parameters, and then begins loading the MFCA RAM by addressing CPU memory through DMA.

The DMA controller consists of the Z8Ø in combination with a custom gate array Common Bus state controller. When addressed as memory, the DMA controller asserts wait to the Z8Ø 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 Z8Ø is taken out of wait and continues processing.

11.6 Interrupt Structure

The MFCA may be operated in polled or interrupt mode. The Z8Ø 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 using the Common Bus interrupt signal level 6 (CBIREQ6). Each MFCA in the 86ØØ (maximum of two) is tied in common to this interrupt level, and conforms to the common bus timings. Interrupts to the CPU through the common bus are used for CPU-to-MFCA communication. Internal interrupts to the Z8Ø are used for specialized functions of the Z8Ø and for communication between the Z8Ø and special devices such as the SIO.

11.7 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.

11.8 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.

The MFCA may be placed in a continuous diagnostic loop through the use of a jumper. The MFCA also has the capability to loop back the transmit signals to the receive signals.

PART 12
SYSTEM FIRMWARE

12.1 Introduction

The 8600 processor has 4K of system ROM that resides on the CP/RIM board and an auxiliary ROM whose size is 4K expandable to 12K. 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 12.

12.2 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.

12.3 Diagnostics

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

Once the POR diagnostics have successfully run to completion, control is transferred to the operating firmware to complete initialization. If an operable system cannot be configured, control will be transferred to debug.

The invokable diagnostics, which are available through debug, test the RAM memory, KDS, and PIO.

Table 12-1. Keyboard Codes

The following is a list of keys and the codes they generate when the firmware key entry routine is used.($86DOSKY and $86KEYIN entry points). All ten function keys are ignored.

KEY	U	T	KEY	U	T	KEY	U	T
A	141	141	o	117	117	<	Ø74	Ø74
B	142	142	p	12Ø	12Ø	=	137	Ø75
C	143	143	q	121	121	>	Ø76	Ø76
D	144	144	r	122	122	?	Ø77	Ø77
E	145	145	s	123	123	@	Ø42	1ØØ
F	146	146	t	124	124	[	1ØØ	133
G	147	147	u	125	125	]	14Ø	135
H	15Ø	15Ø	v	126	126	^	135	136
I	151	151	w	127	127		Ø75	137
J	152	152	x	13Ø	13Ø	{	174	173
K	153	153	y	131	131	|	Ø46	174
L	154	154	z	132	132	}	134	175
M	155	155	Ø	Ø6Ø	Ø6Ø	~	175	176
N	156	156	1	Ø61	Ø61	TAB	Ø33*	Ø3Ø*
O	157	157	2	Ø62	Ø62	TAB	ØØ3	Ø3Ø
P	16Ø	16Ø	3	Ø63	Ø63	RETURN	Ø15*	Ø15*
Q	161	161	4	Ø64	Ø64	RETURN	ØØ2	Ø15
R	162	162	5	Ø65	Ø65	BKSP.	Ø1Ø*	Ø1Ø*
S	163	163	6	Ø66	Ø66	BKSP.	ØØ1	Ø1Ø
T	164	164	7	Ø67	Ø67	INSERT	133*	Ø4Ø*
U	165	165	8	Ø7Ø	Ø7Ø	DELETE	173	177
V	166	166	9	Ø71	Ø71	COMMAND	136*	134*
W	167	167	SPACE BAR	Ø4Ø	Ø4Ø	COMMAND	176	14Ø
X	17Ø	17Ø	!	Ø41	Ø41			
Y	171	171	"	Ø52	Ø42			
Z	172	172	#	Ø43	Ø43			
a	1Ø1	1Ø1	$	Ø44	Ø44			
b	1Ø2	1Ø2	%	Ø45	Ø45			
c	1Ø3	1Ø3	&	Ø47	Ø46			
d	1Ø4	1Ø4	'	Ø72	Ø47			
e	1Ø5	1Ø5	(	Ø51	Ø5Ø			
f	1Ø6	1Ø6	)	ØØØ	Ø51			
g	1Ø7	1Ø7	*	Ø5Ø	Ø52			
h	11Ø	11Ø	+	177	Ø53			
i	111	111	COMMA ,	Ø54	Ø54			
j	112	112	DASH -	Ø55	Ø55			
k	113	113	.	Ø56	Ø56			
l	114	114	/	Ø57	Ø57			
m	115	115	:	Ø53	Ø72			
n	116	116	;	Ø73	Ø73			

NUMERIC PAD

KEY	UNSHIFTED		SHIFTED	
	U	T	U	T
.	ØØ4	Ø56	ØØ4	Ø56
Ø	ØØ5	Ø6Ø	Ø21	Ø6Ø
1	ØØ6	Ø61	Ø22	Ø61
2	ØØ7	Ø62	Ø23	Ø62
3	Ø11	Ø63	Ø24	Ø63
4	Ø12	Ø64	Ø25	Ø64
5	Ø13	Ø65	Ø26	Ø65
6	Ø14	Ø66	Ø27	Ø66
7	Ø16	Ø67	Ø3Ø	Ø67
8	Ø17	Ø7Ø	Ø31	Ø7Ø
9	Ø2Ø	Ø71	Ø32	Ø71

U	Untranslated codes obtained using $86DOSKY
T	Translated codes obtained using $86KEYIN
*	Unshifted key
a thru z	Shifted A thru Z

12.4 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	Vector	Default
Address	Type	Action
Ø1674ØØ	Memory Parity Error	* E1 MEMORY PARITY ERROR *
Ø1674Ø6	Input Parity Error	* E2 INPUT PARITY ERROR *
Ø167414	Output Parity Error	* E3 OUTPUT PARITY ERROR *
Ø167422	Write Protect Violation	* E4 WRITE PROTECT ERROR *
Ø16743Ø	Access Protect Violation	* E5 ACCESS PROTECT ERROR*
Ø167436	Privileged Instruction Violation	* E6 INSTRUCTION ERROR *
Ø167444	One Millisecond Interrupt	POPs system status from stack and jumps to zero
Ø167452	System Call	* E7 INSTRUCTION ERROR *
Ø16746Ø	Breakpoint	Saves status and enters debug
Ø167466	Unassigned Instruction	* E8 INSTRUCTION ERROR *
Ø167474	Sector Table Parity Error	* E9 SECTOR PARITY ERROR *
Ø1675Ø2	Power or Thermal Failure Trap	* POWER FAIL *
Ø16751Ø	Halt	* HALT *

Figure 12-1: System RAM Vectors

12.5 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, Microbus, or RIM. The loader performs a wraparound

2.4.4 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

2.4.5 Miscellaneous Signals

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

2.5 Read/Write Cycles

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

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

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

2.6 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.

2.7 Priority Transfer Cycles

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.

search of peripheral storage devices for a valid IPL block. The search is performed in the following order:

DISPLAY KEY UP	DISPLAY KEY DOWN
* PIO Devices	
93Ø1 Tape	93Ø1 Tape
93Ø1 Disk Ø	93Ø1/93Ø2/93Ø3 Disks Ø-4
* Microbus Devices	
931Ø/9315 Disk	931Ø/9315 Disk
14Ø1/14Ø3/9315 Diskette Ø	14Ø1/14Ø3/9315 Diskette Ø-1
9324/9325 Disk Ø	9324/9325/9326/9327 Disks Ø-3
* RIM Devices	
CP RIM Ø	CP RIM Ø

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. Control is transferred to the RIM loader routine if the search sequence completes without finding any other device present. However, if the RIM loader is also unable to find a valid IPL block, it transfers control back to the beginning of the search sequence.

Certain keys, when depressed during the boot scan sequence, cause special actions to occurr. Depression of the ATT key during the boot causes the 931Ø/9315 disk to be ignored even if present (in order to force booting from the 14Ø1 diskette). For the 9325, it forces drive 1 to be checked instead of drive Ø so that the fixed disk may be booted. If both KBD and DSP are held down, local peripherals are ignored and a boot from the RIM is forced.

12.6 Externally Callable Routines

The system ROM contains several subroutines that have external entry points as listed below. The number of stack levels shown does not include the caller's return address entry. On exit, all flags and register X are indeterminate unless otherwise noted.

Entry Point	Routine Name	Description
Ø170Ø62	$86TYPE	86ØØ Type Determination (86Ø1/86Ø2)
Ø170Ø66	$86VER	Version and Revision Determination
Ø170Ø70	$86KEYIN	Input Translated Keyboard Entry
Ø170Ø73	$86KDSII	Initialize the Keyboard and Display
Ø170Ø76	$86CHRLD	Load the Display Character Font
Ø170101	$86DSPII	Initialize the Display
Ø170104	$86DSPLY	Display the Character String Pointed to by HL
Ø170107	$86CRSLD	Blink the Cursor at Screen Coordinates in DE
Ø170112	$86SCLOC	Calculate the Display Buffer Address
Ø170115	$86RSTRT	Reboot the Machine
Ø17012Ø	$86DOSKY	Input Untranslated Keyboard Entry
Ø170123	$CONBOOT	Boot from RIM
Ø170126	$TRKLOAD	Load data from 93Ø1, 93Ø2, or 93Ø3 disk
Ø170131	$ROM1MS	One-Millisecond Interrupt
Ø170134	$ECCSTAT	Read from I/O Space

12.6.1 Determine Firmware Version

Two externally defined entry points into system ROM are defined to allow a software program to determine both the system firmware type and the firmware version and revision levels. These entry points are $86TYPE (Ø17ØØ62) and $86VER (Ø17ØØ66).

On system firmware version one, both of these addresses in system ROM contain two bytes of zeros, indicating that this is version one firmware (although the type of machine -- 86Ø1- or 86Ø2-type -- is not known). On version two and later firmware, these bytes contain pointers to character strings containing version information.

Location $86TYPE contains a two-byte address into auxiliary ROM which has a four-byte character string describing the model type of the system (either 86Ø1 or 86Ø2). The software must ensure that the auxiliary ROM has been enabled before this string may be read, and should ensure that it is disabled when through.

Location $86VER contains a two-byte address into system ROM where a six-byte character string defines the release version and revision. The format is "n.m.pp" where "n" is the version number, "m" the revision number, and "pp" is the pre-release designation. The last three bytes are always blanks for final released firmware.

12.6.2 $86KEYIN

Entry point: Ø17ØØ7Ø

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

Flags: The zero flag is set (T2) if no character is available. The zero flag reset (FZ) if the character is presented. All others are indeterminate.

Stack: Four levels are used.

Exit: Return to calling routine.

This subroutine obtains a character from the keyboard. The A register contains the translated character code on exit (see Table 12-1).

12.6.3 $86KDSII

Entry point: Ø17ØØ73

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

Stack: Five 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 (see Table 12-2 for the default character font).

12.6.4 $86CHRLD

Entry point: Ø17ØØ76

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

Stack: Up to three levels are used.

Exit: Return to calling routine.

Format of the font load table:

First byte: ASCII value (that is, Ø1Ø1 for "A"), followed by 12 bytes of font data used to display the corresponding character.

14th byte: ASCII value, 12 bytes of font data
*
*
*
Last byte: ASCII value, 12 bytes of font data ØØØ.

Last byte: ØØØ (the trailing zero terminates the load).
(The trailing zero terminates the load.)

This subroutine loads the character font RAM from a character font data string pointed to by HL.

The top line of the display character is defined by the first byte after the ASCII value, and the bottom line is defined by the twelfth byte after the ASCII value. The ASCII value can be any value from 1 to 127. Since Ø is used as the termination character, 128 can be used to load the font for Ø. This is possible because bit 7 (MSB) of the ASCII value is ignored (which means that 129 loads font for 1, and so on).

The screen is blanked (blank screen mode) during the font load operation and upon completion it is restored to its entry mode. Also, keyboard status is preserved.

12.6.5 $86DSPII

Entry point: Ø17Ø1Ø1

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

Stack: Four 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. It also initializes the keyboard to control filter mode.

12.6.6 $86DSPLY

Entry Point: Ø17Ø1Ø4

Registers: Entry:

D= Horizontal screen coordinate for the first character to be displayed (Ø to 79).

E= Vertical screen coordinate for the first character to be displayed (-12 to 11).

HL= Pointer to string to be displayed.

B= Display attributes.

Exit:

D= Horizontal screen coordinate past that of the last character displayed unless changed by a control sequence, such as $H, or already on last column (D=79).

E= Vertical screen coordinate of the last character displayed (unless changed by a control sequence such as $V).

HL= Points to the ending string position plus one.

All other registers= Scratched.

Stack: Up to nine levels 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 Ø	Underline
Bit 1	Two Level
Bit 2	Blink
Bits 3-5	Not used
Bit 6	Nondestructive 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 (Ø2Ø3)=New String Address follows, where AAA AAA is the new address (LSB, MSB format)

$H HHH — $H (Ø11)=New horizontal cursor position follows, where HHH is the new horizontal position.

$V VVV — $V (Ø13)=New vertical cursor position follows, where VVV is the new vertical position.

$RU — $RU (Ø23)=Roll screen up one line.

$RD — $RD (Ø24)=Roll screen down one line.

$EEOL — $EEOL (Ø22)=Erase to End of Line. Cursor remains at the same position.

$EL — $EL (Ø15)=End of Line. Carriage return and line feed and end the string. Roll up one line if already on the bottom line.

$EEOF — $EEOF (Ø21)=Erase to End of Frame. Erase to end of this line and all screen lines below this one. Cursor remains at the same position.

$BP — $BP (ØØ7)= Beep.

$F — $F (Ø33)=Force display of next character.

$CK — $CK (Ø2Ø7)=Click.

$HA — $HA (Ø211)=Horizontal adjustment follows.

$VA — $VA (Ø213)=Vertical adjustment follows.

$HU — $HU (Ø223)=Home Up. The cursor returns to the home position, the upper left-hand corner.

$HD — $HD (Ø224)=Home Down. The cursor returns to the home down position, the lower left-hand corner.

$O — $O (Ø233)=New Options follow. The option bits are described above.

$ES — $ES (ØØ3)=End of string.

Each display string must be terminated by a $ES character (ØØ3), or by a $EL (Ø15).

12.6.7 $86CRSLD

Entry Point: Ø17Ø1Ø7

Registers: Entry: DE cursor coordinates (see 12.6.6).

Exit: A, B, and C are scratched.

Stack: Three levels are used.

Exit: Return to calling routine.

The $86CRSLD subroutine positions and displays the cursor to the screen coordinates contained in D and E. Loading D or E to an off-screen location, such as D= -1, turns the cursor off.

12.6.8 $86SCLOC

Entry point: Ø17Ø112

Registers: Entry: DE contains cursor coordinates (see Section 12.6.6).

Exit: DE contains the screen buffer address. C and A are 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. If any of the given coordinates are off screen, this routine returns with registers D and E each set to octal Ø377.

12.6.9 $86RSTRT

Entry Point: Ø17Ø115

Registers: Entry: None.

Exit: Indeterminate.

This subroutine reboots the system.

12.6.1Ø $86DOSKY

Entry point: Ø17Ø12Ø

This subroutine obtains a character from the keyboard. See $86KEYIN routine for entry and exit parameters (see Table 12-1 for character codes).

12.6.11 $TRKLOAD

Entry point: Ø17Ø126

Registers: Entry:

HL	Points to load address
DE	Bits Ø-9 = start cylinder number (Ø-548) Bits 1Ø-15 = start sector number (Ø-23)
C	Bits Ø-4 = number of sectors (Ø-24) Bits 5-7 = head number (Ø-5)
B	Bits Ø-3 = controller number (Ø-15) Bits 4-6 = Drive number (Ø-4) Bit 7 must be zero

Exit: All registers are scratched.

Flags: True carry if there is an error, false carry if no error.
Stack: Up to 19 levels used.
Exit: Return to caller.

This routine addresses the given controller, then loads a block of data from the given disk drive into the 86ØØ's memory.

Version 2 and later firmware returns with true carry if an attempt is made to load data at or above octal Ø1ØØØØØ, or any of the entry parameters is out of range or the given drive or controller is not on line.

12.6.12 $CONBOOT

Entry point: Ø17Ø123

Registers: Entry: None
Exit: All are scratched

Exit: Does not return

This routine invokes the RIM loader.

12.6.13 ROM1MS

Entry point: Ø17Ø131

This entry point may be used to perform the identical action to the default one millisecond interrupt (see section 12-4).

12.6.14 $ECCSTAT

Entry point: Ø17Ø134

Registers:	Entry:	HL=	Pointer to a physical memory address
		DE=	Pointer to an I/O space address
	Exit:	A=	Contents of I/O space address pointed to by DE All others preserved
Stack:	One level used		
Exit:	Return to the calling routine		

$ECCSTAT first performs a read from the address pointed to by HL; the contents of the I/O space address pointed to by DE are then read into register A. Interrupts are disabled during this operation and restored to their entry status before control is transferred back to the caller.

This routine is not available on firmware versions earlier than 2.1 and should not be called. If it is, the "* E8 INSTRUCTION ERROR *" message is displayed on the bottom of the screen and the debugger is invoked.

12.7 Debug

The immediate accessibility of debug creates a flexible debugging interface between user and machine. The 86ØØ debug routines are similar in function to the 55ØØ/66ØØ debug routines; however, some commands have been changed, added, or deleted.

12.7.1 Entry to DEBUG

There are six methods of entry to DEBUG:

(1) Manual depression of the CTRL, INT, and DSP keys, followed by release of INT key before the others.
(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 debugger call command.

(6) Irrecoverable error during diagnostic execution.

12.7.2 Saving the Machine State

In the first four cases of Section 12.7.1, the processor pushes the program counter onto the stack, followed by the condition code flags (also referred to as flags) as the most significant byte, and the system status register as the least significant byte. The processor then disables interrupts and switches to system mode.

In case 5, the status register is not available. In this case, when displaying or modifying the status register (nnnY command), and when restoring the machine state on exit, the debugger uses the status register value which was last saved. It should be noted that case 5 cannot work with a user-mode routine because the RET instruction does not force the system out of user mode and user memory space.

In all cases, after up to four pushes onto the stack (two pushes by the processor as mentioned earlier and two pushes by the debugger itself), the debugger switches to the debugger stack to preserve the remaining entries of the system stack. It also saves the currently active register set, the flags (see the nnnf command for a description of how the flags are saved and restored), the system control register, the KDS screen status byte, and the base register. These are referred to as the saved values. This allows the debugger to use any registers that are needed as long as it restores them to their original conditions upon exit. The registers used during the debugger execution are referred to as active (or current) values. All of the debugger commands modify/display the saved values unless indicated otherwise.

Upon exit from the debugger (using the E command) all registers and flags, stack, and interrupt modes are restored to their saved values. Note that if the user has modified a register (that is, if register A has been modified using the nnna command), upon exit, the debugger loads that register with its modified value. Also, the debugger uses up to four stack entries before it switches to its own stack. (Actually, the debugger uses up to two levels of stack in addition to the two pushes by the processor which were mentioned earlier in this section.) This means that the machine state is totally restored to its condition prior to entry into the debugger, except for values which were intentionally altered (the A register in the example given above), and that the bottom four levels of the stack have been scratched. (The four pushes mentioned earlier in this section will have scratched the bottom four levels of the thirty two level wrap-around stack.)

The C and J commands also cause restoration of the machine state (as described above) with the exception that they leave the vectored interrupts masked off, they do not restore the system status register, and the machine stays in system mode.

12.7.3 Command/Display Format

The 8600 Debugger maintains two current addresses, one for memory access, and the other for I/O space. All references to CURADR in this document indicate the current address which is active at the time.

To relieve the user of the task of having to compute addresses of entries in a buffer, the debugger provides the origin mode. In this mode, the user sets the base address (referred to as origin bias) to the beginning address of the buffer using nnnnnni or nnnnnno command. Then the user may examine or modify entries of the buffer relative to this origin bias using nnnnnn<enter> command. The debugger allows for up to ten different origin biases by providing an origin table of ten entries. The nnO command allows entry to origin mode, and also tells the debugger which origin bias (of the ten allowed) is to be used.

The basic debugger display consists of five lines displayed at the lower right corner of the screen:

RRRRRR	=	CURRENT ADDRESS OFFSET FROM ORIGIN
AAAAAA	=	CURRENT ABSOLUTE ADDRESS
* NNN	=	THE VALUE STORED AT CURADR
MMMMMM	=	LSB,MSB ADDRESS FORMED AT CURADR
nnnnnnn*	=	COMMAND INPUT LINE

* Represents ASCII character for NNN in 3rd line and operator command entry on 5th 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. Upon entry, debugger uses the mode (origin mode or not origin mode) it was in last to determine whether to display the relative address RRRRRR. This means the display may not look the same upon each entry to debugger.

The bottom line of the display is used to edit and input commands to DEBUG. The blinking cursor signifies that the Command Interpreter is awaiting user input.

Data is entered serially into the input display buffer. The cursor is displaced to the right successively as this occurs. The BACKSPACE key erases the character most recently entered, shifting the entry cursor to the left one space. The CANCEL key deletes the entire entry.

Commands which accept input arguments are preceded by the argument which is entered in octal. Not all commands require an input argument. The last character input to the interpreter must be a legal command. Illegal input is ignored, evoking a BEEP from the 8600. Commands are executed upon their entry into the interpreter (no ENTER key is required and the command character is not displayed), with the current contents of the entry line being cleared. Upon command completion the cursor reappears, awaiting further input.

12.7.4 Command Syntax

Below is the notation followed in the command list. Note that all digits are in octal and no leading zero is necessary.

nnn — Indicates an optional sequence of octal digits not to exceed the number of n's shown in the list for that command.

(nnn)nnn — Indicates that the command modifies or displays one byte or two depending on whether the value entered is larger than octal 377. In general, two bytes are affected by the command, either a register pair or a memory address in LSB, MSB format.

nnnnnn — Two bytes are affected. No digits usually cause special action noted in each individual command description.

12345 — There exists a set of special commands whose accidental execution is inhibited by the requirement that they contain this unique argument.

Note: — The user must ensure the number of digits entered for a command does not exceed the number of n's shown in the command list. The debugger does not check the number of digits entered for all commands. For instance, two registers will be modified if a number larger than octal 377 is entered when modifying a register.

12.7.5 Input Command List

Unshifted Debugger Commands

nnnnnnA Set current I/O space address to nnnnnn and display that location. If nnnnnn is not given use last current I/O space address. If in memory space, switch to I/O space and clear origin mode before setting current address. (See Section 12.7.3 for origin mode.)

nnnnnnB Set a breakpoint at nnnnnn. If nnnnnn not given, set a breakpoint at CURADR. (Maximum of ten breakpoints.)

nnnnnnC Call the given or current address. The machine state is restored before the control is passed to the subroutine. See Section 12.7.3 for saving and restoring machine state. Note that this command leaves the vectored interrupts masked off. It also leaves the machine in system mode and does not restore the system status register.

A RETURN from the called subroutine causes reentry into the debugger and saving of the machine state.

nnnnnnD Decrement CURADR (I/O space or memory space as currently selected) by the value nnnnnn. If no nnnnnn, decrement by one.

nnnnnnE Continue execution from nnnnnn. If nnnnnn not given use top system stack entry. Machine state is restored to the condition existing prior to debugger entry. (See Section 12.7.2 for saving and restoring the machine state by debugger.)

nnnnnnI Increment CURADR (I/O space or memory space as currently selected) by the value nnnnnn. If no nnnnnn, increment by one.

nnnnnnJ Jump to nnnnnn. If no nnnnnn, jump to current address. The machine state is restored in the same way as the C command before the jump is performed. Note that upon exit using nnnnnnJ command, the debugger leaves the flags and status register on top of the stack. This means that if the nnnnnnJ command is used to jump into a subroutine, the return address of that subroutine is not on top of the stack as may be expected.

nnnK Set the active control register to nnn. If nnn not given, display the control register. This command sets the active control register, not the saved value.

L Link to the address pointed to by CURADR. This command forms an address using contents of current memory address as LSB, and contents of the next higher address as MSB. It then replaces CURADR with this newly formed address and displays it.

(nnn)nnnM Modify the contents of the current address location (I/O space or memory space as currently selected). If nnnnnn > Ø377, modify two bytes, LSB followed by MSB. If no nnnnnn given, use ØØØ.

nnnnnnN Set current memory address to nnnnnn and display that location. If no nnnnnn is given, use memory address zero. If in I/O space, switch to memory space and clear origin mode before setting current address. See Section 12.7.3 for origin mode.

nnO Set origin mode, then set origin table pointer to nn. If no nn is given, clear origin mode, for either Memory or I/O Space. See Section 12.7.3 for origin mode. This command allows having more than one origin bias at one time and the ability to switch between them. A total of ten origin biases are allowed for memory and I/O spaces together. This command checks to see if an origin bias has already been set for nnth entry of origin table. If so, then it selects or deselects I/O space accordingly (depending on whether nnnnnni or nnnnnno was used to set the origin bias). If no origin bias has been set, then it selects memory space (thus deselecting I/O space).

nnnnnnP Load the active base register with the upper 8 bits of nnnnnn - Ø1ØØØØØ. If no nnnnnn given, display the base register. This command loads the active base register, not the saved value.

12345Q Load the sector table selected by the control register. CURADR points to a table (with single byte entries) whose 1st entry contains the following information:

1. The number of entries to be loaded into the sector table is in the four least significant bits.

2. The offset of the first entry into the sector table is in the four most significant bits.

The remaining data in the table are the arguments to be loaded into the sector table. See Table 12-3 for sector table entry format. 12345Q is used for systems without expanded sector table. See 12345q for loading expanded sector tables.

Note: During power-up, system firmware initializes the sector tables so that entries Ø15 & Ø16 (octal) of system instruction sector table point to logical RAM locations Ø15ØØØØ - Ø167777 used by the debugger & 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 register set switch. The saved registers are reloaded, the switch is performed, then the newly active registers are saved. The ASCII character displayed on the third display line reflects the newly selected register set. (A for ALPHA and B for BETA set.)

nnS Display stack entry nn. Error if nn > Ø37 octal. If no nn given, display entry Ø.

nnnT Display the 8 bit sector table entry, where the most significant n (leftmost n) selects the sector table (Ø-3) and the least significant n's select the entry into that sector table (Octal ØØØ - Octal Ø17). If nnn not given, display sector table Ø, entry Ø. Error if nnn is out of range. For 16-bit sector table entry on systems with expanded sector tables, see nnnt command. Table 12-3 shows the format of sector table entries.

12345T Start memory self-test.

nnnY Modify or display the saved system status register.

Z Display all registers and register pairs. The display format is:

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

Where the letters indicate the register (pairs), SSSSSS is the stack pointer, and FFF indicates a value which if added to itself will restore flags. (See nnnf command for the format.)

Shifted Debugger Commands

(nnn)nnna Modify saved Register A value to nnn and display. If no nnn, just display saved value. If nnnnnn > octal Ø377, modify XA register pair (saved value).

nnnb Modify and/or display register B value.

(nnn)nnnc Modify and/or display register C value. If nnnnnn > octal Ø377, modify BC register pair.

nnnd Modify and/or display register D value.

(nnn)nnne Modify and/or display register E value. If nnnnnn > octal Ø377, modify DE register pair.

nnnf Modify and/or display condition code flags. Debugger saves condition code flags by performing a CCS instruction and then saving the A register. This saved value (which is displayed/modified when nnnf command is entered) is in the following format:

MSB	Bit 7	= carry
	Bit 6	= sign
	Bits 5-2	= Ø
	Bit 1	= not zero and not sign
LSB	Bit Ø	= not zero and not parity

When modifying flags using nnnf command, the user must note that upon exit, debugger performs an add of this saved value to itself in order to restore the flags.

nnnh Modify and/or display register H value.

nnnnnni Set origin bias to nnnnnn and select I/O space, thus deselecting memory space. This command will display the relative address RRRRRR above the CURADR display line. (RRRRRR = CURADR - origin bias) If no nnnnnn is given, display CURADR and relative address using last origin bias. Note that if not in origin mode, the origin bias used and the memory or I/O space selection are not defined.

nnnk Alter the saved control register value to nnn and display. If no nnn, just display. If in I/O space, select memory space, thus deselecting I/O space.

(nnn)nnnl Modify and/or Display register L value. If nnnnnn > octal Ø377, modify HL register pair.

nnnnnno Set origin bias to nnnnnn and select memory space, thus deselecting I/O space. This command displays relative address RRRRRR above the CURADR display line. (RRRRRR = CURADR - origin bias) If no nnnnnn is given, display CURADR and relative address using last origin bias. Note that if not in origin mode, the origin bias used and the memory or I/O space selection are not defined. See Section 12.7.3 for origin mode and origin bias.

nnnnnnp Alter the saved base register to upper 8 bits of nnnnnn-Ø1ØØØØØ and display it. If no nnnnnn, just display saved value.

12345q Load the sector table selected by the control register. This is similar to '12345Q'. However, the entries following the first entry are all 2 byte word entries in order to support the expanded sector tables. These entries are in format shown in Table 12-3. The note for '12345Q' also applies here.

nnr Pop stack nn times. Error if nn > octal Ø37. This command removes nn entries from top of the stack and displays the new top entry of the stack. If nn is not given, one entry is removed. If nn=Ø, no entry is removed.

nnnnnns Push value nnnnnn onto stack. If no nnnnnn is given, push zero.

nnnt	Display the 16 bit sector table entry nnn. This is similar to 'nnnT'. However, the sector table entry is 2 bytes. To accommodate the 2 byte entry the entry is displayed on the line right above the command line. The sector table entry is in the format shown in Table 12-3.
12345t	Start keyboard and display invokable diagnostic.
12345u	Start PIO board invokable diagnostic.
nnnx	Modify and/or display register X.
nnnnnn<ENTER>	Set relative address RRRRRR in memory or I/O space to nnnnnn. If no nnnnnn, set to current address. CURADR = RRRRRR + origin bias. See Section 12.7.3 for origin mode and origin bias.
<CANCEL>	Cancel the command input line.
<BACKSPACE>	Backspace one space on input line.
(nnn)nnn.	Modify memory or I/O space contents, then increment the current address. If nnnnnn < Ø4ØØ modify one byte and increment CURADR by 1. If nnnnnn > Ø377 modify two bytes (LSB at CURADR and MSB at next higher memory location) and increment CURADR by two. No nnnnnn is treated as ØØØ.
(nnn)nnn^	Same as '.' but save nnnnnn. If no nnnnnn use the last nnnnnn saved. This command is useful when trying to fill a buffer with the same value. First (nnn)nnn^ is entered to set the first entry (byte or word) to nnnnnn, then ^ is entered to fill the next entry with that same value.
#	Clear all active debugger set breakpoints. (Maximum of ten breakpoints.)

?   Display the processor information data. The following information is displayed on the bottom of the screen:

```
86Øx M: v.r   P:ØØ5:nnn
```

Where 86Øx is the Processor Type (86Ø3 or 86Ø5), M: v.r indicates System Firmware version=v, revision=r, and P:ØØ5:nnn means processor type code = ØØ5 and microcode level = nnn octal.

Error entries beep and return to command interpreter. (Command line is not erased.) Note that the number of digits entered for a command must not exceed the number of n's shown for that command. Debugger does not check the number of digits for all commands.

Table 12-2: Default Character Font

Octal 7-bit ASCII	Character	
010	<---	left arrow
030	--->	right arrow
040		space
041	!	
042	"	
043	#	
044	$	
045	%	
046	&	
047	'	quotation
050	(	
051	)	
052	*	
053	+	
054	,	comma
055	-	dash
056	.	period
057	/	slash
060	0	
061	1	
062	2	
063	3	
064	4	
065	5	
066	6	
067	7	
070	8	
071	9	
072	:	
073	;	
074	<	
075	=	
076	>	
077	?	
0100	@	
0101	A	
0102	B	
0103	C	
0104	D	
0105	E	
0106	F	
0107	G	
0110	H	
0111	I	
0112	J	
0113	K	
0114	L	
0115	M	
0116	N	
0117	O	
0120	P	
0121	Q	
0122	R	
0123	S	
0124	T	
0125	U	
0126	V	
0127	W	
0130	X	
0131	Y	
0132	Z	
0133	[	
0134		backslash
0135	]	
0136	^	caret
0137	_	underline
0140		accent grave
0141 to 0172		A to Z (same as 0101 to 0132)
0173	{	
0174	|	
0175	}	
0176	~	tilde
0177		contraction

All other 7-bit ASCII codes are set to display a triangle. If bit 7 of an 8-bit character is set to 1, the hardware displays the character in inverse video.

Table 12-3: Debugger Sector Table Format

1 - The 8-bit sector table entry format (used by '12345Q' and 'nnnT' debugger command) is:

7	6	5	4	3	2	1	Ø	Bit
A15	A14	A13	A12	WE	AE	A17	A16	

2 - The 16-bit sector table entry format (used by '12345q' and 'nnnt' debugger command) is:

15	14	13	12	11	1Ø	9	8	7	6	5	4	3	2	1	Ø	Bit
WE	AE	Ø	Ø	Ø	Ø	A21	A2Ø	A19	A18	A17	A16	A15	A14	A13	A12	

Where: A21 through A12 are the address bits for the 4K block of memory. AE=1, access enables the 4K block, and WE = 1, write enables it.

Bits 1Ø through 13 in the second format should be Ø on write operations but are indeterminate on readback.

Note: The formats shown above, apply only to 86ØØ system firmware debugger.

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 components 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 is installed 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

A.2.1 Protection Features

The power supply has five protection features that prevent damage from certain conditions. These 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.

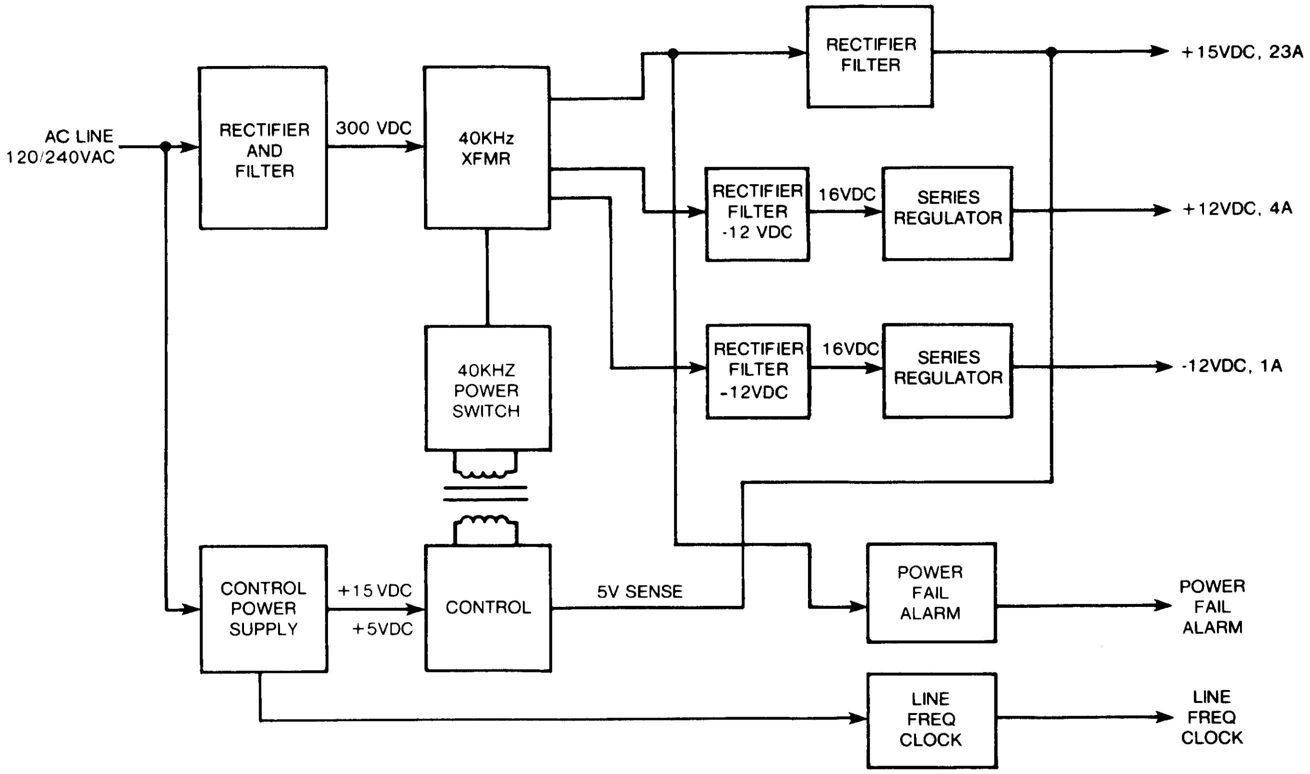


Figure A-1: Power Supply

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 checked circuits detect an undesirable operating condition, they 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 four-layered 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 by way of feed-throughs at three points, 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. Selected signals have pull-up/pull-down resistor terminators.

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. 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 tied in common between all the 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 CBPRII and CBPRIO together (if DMA is not required). The highest-priority slot is slot one; the lowest-priority slot is slot nine.

APPENDIX B
INSTRUCTION TIMINGS

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

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
OR(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
OR(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
SBM	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
SBM(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

Instruction	8600 Timing	6600 Timing
AC data	1.5Ø	1.6Ø
SU data	1.5Ø	1.6Ø
SB data	1.5Ø	1.6Ø
ND data	1.5Ø	1.6Ø
XR data	1.5Ø	1.6Ø
OR data	1.5Ø	1.6Ø
CP data	1.5Ø	1.45
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	2.3Ø
SLC	1.ØØ	1.15
SRC	1.ØØ	1.15
SRE	Ø.75	1.15
SLC(r)	1.75	2.ØØ
SRC(r)	1.75	2.ØØ
SRE(r)	1.5Ø	2.ØØ
JMP loc	2.5Ø	2.Ø5
Jcc loc	2.5Ø	2.25
Jcc loc (fall thru)	1.75	1.1Ø
EJMP loc	3.5Ø	3.4Ø
NOJ loc	1.75	1.ØØ
NOP	Ø.75	Ø.7Ø
CALL loc	3.75	2.2Ø
Ccc loc	3.75	2.45
Ccc loc (fall thru)	1.75	1.2Ø
RET	3.25	1.3Ø
Rcc	3.25	1.5Ø
Rcc (fall thru)	1.ØØ	Ø.8Ø
UR	4.ØØ	2.45
EUR	4.ØØ	3.15
IN	9.25	5.ØØ
IN(r)	1Ø.ØØ	5.85
PIN	9.25	5.ØØ
PIN(r)	1Ø.ØØ	5.85
EX(exp)	9.5Ø	7.ØØ
EX(r)(exp)	1Ø.25	7.85
EX BEEP	16.75	7.ØØ
EX CLICK	16.75	7.ØØ
MIN	Ø.75 + 9.75N	2.95 + 8.3ØN
MOUT	Ø.75 + 1Ø.ØØN	2.95 + 8.3ØN

Instruction	**86ØØ Timing**	**66ØØ Timing**
BETA	8.75	1.2Ø
ALPHA	8.75	1.2Ø
DI	1.ØØ	1.2Ø
EI	1.ØØ	1.2Ø
POP	2.25	1.45
POP(rp)	3.ØØ	2.3Ø
PUSH	2.25	1.15
PUSH(rp)	3.25	2.ØØ
PUSH loc	3.75	2.Ø5

(NOTE: The slash, /, here stands for the not equal sign, an equal sign with a slash through it.)

BT(A=B=Ø)	1.25 + 2.25N	5.7Ø + 1.6ØN
BT(A,B/Ø)	1.5Ø + 2.25N	6.ØØ + 2.35N
BTR(A=B=Ø)	2.ØØ + 2.25N	7.85 + 1.6ØN
BTR(A,B/Ø)	2.25 + 2.25N	6.95 + 2.35N
BCV(A=B=Ø)	1.5Ø + 3.ØØN	7.55 + 2.5ØN
BCV(A,B/Ø)	1.75 + 3.ØØN	6.85 + 3.25N
BCP (if match)	Ø.75 + 2.5ØN	5.35 + 1.95N
BCP (mismatch)	1.ØØ + 2.5ØN	4.85 + 1.95N
BFAC	1.5Ø + 2.25N	5.35 + 2.15N
BFSB	1.5Ø + 2.25N	5.35 + 2.15N
DFAC	1.5Ø + 3.25N	6.2Ø + 3.45N
DFSB	1.5Ø + 3.ØØN	6.3Ø + 2.9ØN
BFSL	Ø.75 + 2.ØØN	3.ØØ + 1.7ØN
BFSR	1.5Ø + 2.ØØN	3.4Ø + 1.55N
STKS	Ø.75 + 3.25N	1.55 + 1.7ØN
STKL	1.5Ø + 3.25N	3.6Ø + 1.7ØN
REGS	1Ø.ØØ	9.1Ø
REGL	7.5Ø	9.85
CCS	1.ØØ	1.65 to 2.45
CCS (r)	1.75	2.5Ø to 3.3Ø
INCP HL	1.25	1.4Ø or 1.95
INCP HL,A	1.25	1.55 or 2.1Ø
INCP (rp)	2.25	2.45 or 3.ØØ
INCP (rp),2	2.5Ø	2.5Ø or 3.Ø5
INCP (rp),A	2.25	2.4Ø or 2.95
DECP HL	1.25	1.4Ø or 1.95
DECP HL,A	1.25	1.55 or 2.1Ø
DECP (rp)	2.25	2.45 or 3.ØØ
DECP (rp),2	2.5Ø	2.5Ø or 3.Ø5
DECP (rp),A	2.25	2.4Ø or 2.95
DL DE,HL	2.25	2.5Ø
DL BC,HL	3.ØØ	3.95
DL BC,BC	3.ØØ	3.75
DL BC,DE	3.ØØ	3.9Ø

Instruction	86ØØ Timing	66ØØ Timing
DL DE,BC	3.ØØ	3.75
DL DE,DE	3.ØØ	3.9Ø
DL HL,BC	3.ØØ	3.75
DL HL,DE	3.ØØ	3.9Ø
DL HL,HL	2.25	2.5Ø
DS DE,HL	2.25	2.5Ø
DS BC,HL	3.ØØ	3.95
DS BC,DE	3.ØØ	3.9Ø
DS DE,BC	3.ØØ	3.75
DS HL,BC	3.ØØ	3.75
DS HL,DE	3.ØØ	3.9Ø
PL (r),loc	2.25	2.2Ø
PS (r),loc	2.25	2.2Ø
DPL (rp),loc	3.75	3.8Ø
DPS (rp),loc	3.75	3.8Ø
INCI(dsp),(idx)	5.5Ø	3.65 or 5.1Ø
DECI(dsp),(idx)	5.5Ø	3.65 or 5.1Ø
INCI *(dsp),(idx)	7.ØØ	7.25
DECI *(dsp),(idx)	7.ØØ	7.45
LFII(rp),(dsp),(idx)	5.25	5.6Ø
LFID(rp),(dsp),(idx)	5.25	5.8Ø
LFII(rp),*(dsp),(idx)	6.ØØ	6.25
LFID(rp),*(dsp),(idx)	6.ØØ	6.45
BRL	1.5Ø	1.ØØ
BRL(r)	2.25	1.85
STL	1.5Ø + 1.5ØN	2.7Ø + 1.25N
SC	5.ØØ	1.8Ø
BP	5.ØØ	2.ØØ
HALT	5.25	
IMULT	17.ØØ-28.ØØ	26.2Ø-77.55
DIDIV	34.5Ø-39.ØØ	57.75-82.55
IDIV	35.ØØ-39.5Ø	57.75-82.55
DPLR	3.75	3.8Ø
DPSR	3.75	3.8Ø
STLO	2.25 + 1.5ØN	3.7Ø + 1.25N
INFO	2.25	2.5Ø
BFLR	1.5Ø + 2.25N	6.8Ø + 2.15N
D(op)M(rp)	3.ØØ	4.6Ø to 5.65

Instruction	86ØØ Timing	66ØØ Timing
D(op)P(rp),loc	3.75 or 4.ØØ	5.15 to 6.2Ø
D(op)I(rp),datal	3.ØØ or 3.25	4.ØØ to 5.Ø5
DM(op)(rp)	4.5Ø	5.3Ø to 6.35
P(op)(r),loc	3.ØØ	3.4Ø (3.25 for CP)
LLDEL	7.5Ø	9.4Ø
LLINS	9.ØØ	1Ø.8Ø
COMP(rp)	2.25	3.7Ø or 4.75
COMPS(rp)	2.5Ø	4.15 or 5.2Ø
LKA	1.25	
LAS	1.25	
CBOUT	1.75	
CBOUT(r)	2.5Ø	
CBIN	1.75	
CBIN(r)	2.5Ø	
CBSOUT	2.5Ø	
CBSOUT(r)	3.25	
CBSIN	2.5Ø	
CBSIN(r)	3.25	
STKMV	3.ØØ	
IRET	5.25 or 11.5Ø	
STR	2.25	
BLKIN	Ø.75 + 2.25N	
BLKOUT	Ø.75 + 2.25N	
MBLKIN	1.5Ø + 2.25N	
MBLKOUT	1.5Ø + 2.25N	
MS INTR	5.ØØ	1.8Ø
VECTORED INTR	12.25	
RESTART INTR	5.25	1.8Ø
ERROR INTR	7.5Ø	1.8Ø
BRS	8.25	
LAK	7.25	

APPENDIX C
86ØØ 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
ØØØØ-ØØØ1	Interrupt Controller
ØØØ2	Auxiliary ROM Enable/Disable
ØØØ	PR/RIM Indicator Port (Output)
ØØØ4	CP/RIM Indicator Port (Input)
ØØØ5	CP Sector Table MSB Select (Ø=LSB ENABLE, 1=MSB ENABLE)
ØØ1Ø-ØØ17	PIO
ØØ3Ø-ØØ31	KDS Data, Command Status
ØØ34-ØØ37	Printer Port
ØØ4Ø	RIM #Ø Enable (CP/RIM Board)
ØØ42-ØØ43	RIM Ø Status - Command
ØØ56	Parallel Bus I/O Status Register
ØØ6Ø-ØØ62	ECC Memory Controller Registers
ØØ7Ø-ØØ73	MPCA
Ø2ØØ-Ø2Ø7	MFCA #Ø
Ø21Ø-Ø217	MFCA #1
Ø22Ø-Ø227	MFCA #2
Ø23Ø-Ø237	MFCA #3
Ø24Ø-Ø247	MFCA #4
Ø25Ø-Ø257	MFCA #5
Ø26Ø-Ø267	MFCA #6
Ø27Ø-Ø277	MFCA #7
Ø2ØØØ-Ø3777	Microbus
Ø4ØØØ-Ø7777	Parallel Bus I/O Buffer
Ø72ØØØ-Ø73777	MPCA Buffer Memory (all cards)
Ø74ØØØ-Ø77777	RIM Buffer
Ø14ØØØ-Ø143777	Screen Buffer (screen mode) Font Buffer (character font load mode)
Ø144ØØØ-Ø147777	Attribute Buffer
Ø177776-Ø177777	Test Card

DATAPOINT