MARK 12

Installation/ Technical Document

POINT 4 DATA CORPORATION

POINT 4 DATA CORPORATION

MARK 12

INSTALLATION/ TECHNICAL DOCUMENT

June 10, 1986

MARK 12 INSTALLATION/TECHNICAL DOCUMENT

PREFACE

This document contains interim information pertaining to the MARK 12 Computer. The material is drawn from technical memoranda, the MARK 8 Computer Reference Manual, the MARK 12 CPU Product Specification and engineering drawings.

This interim documentation will be replaced by formal documents for the MARK 12 which are now in process. They will include:

MARK 12 User's Guide
MARK 12 Technical Reference Manual

MARK 12 INTERIM DOCUMENTATION

CONTENTS

APPENDICES

(excerpts from MARK 8 Reference Manual)

(from Engineering documents)

POINT 4 Data Corporation

```
       4444
    44444      4
  44444        444
 4444     4    4444
444     444    44444
4     44444    44444

 4444444444    4444
  444444444    444
    4444444    4
       4444
```

TECHNICAL MEMORANDUM

TO: Installers of POINT 4 MARK 12 SYSTEMS

FROM: Tech Support

DATE: June 10, 1986

SUBJ: MARK 12 COMPUTER SYSTEM UNPACKING AND INSTALLATION

This technical memo describes the POINT 4 MARK 12 Computer System and provides unpacking and installation procedures.

CONTENTS

FIGURES

APPENDICES

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 Class A computing devices 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. Only shielded cables with the shield terminated to the metal hood of the connector should be used.

I. INTRODUCTION

A fully-assembled POINT 4 MARK 12 Computer System is shipped in either a highboy cabinet (22 inches wide x 42 inches high x 30 inches deep) or a lowboy cabinet (22 inches wide x 31 inches high x 30 inches deep). The cabinet has four casters which allow it to be moved with ease, four preinstalled leveler feet, and a stabilizer bar to prevent the cabinet from tipping over (see Figure 1).

The system components are completely accessible by virtue of removable panels at the front and sides. The rear panel is hinged and can also be removed.

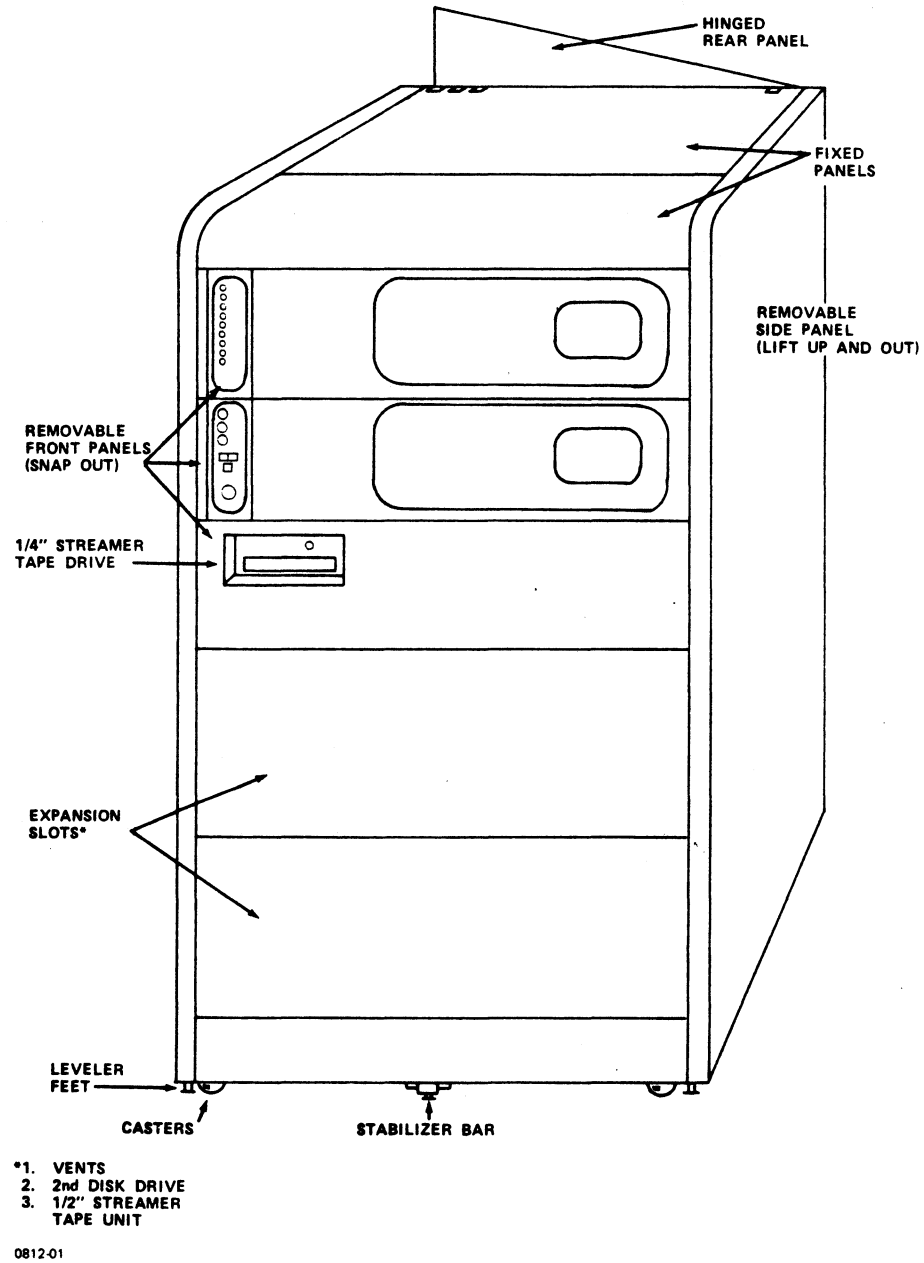


Figure 1. MARK 12 Cabinet

II. MARK 12 COMPUTER SYSTEM COMPONENTS

A full-configuration MARK 12 Computer System includes the following:

A. Main power supply

B. Computer chassis with the following boards:

1. MARK 12 CPU
2. Extended Memory (1-4 boards)*
3. LOTUS 730 Controller
4. 1/2-inch Streamer Tape Controller**
5. POINT 4 310 Multiplexer
6. POINT 4 301 Expansion Board
7. LOTUS Cache Memory (LCM)*

C. Rack-Mounted Peripheral Subsystem (RPS) with the following:

1. 8-inch Disk Drive
2. Peripheral Power Supply Unit
3. QIC-02 1/4-inch Streamer Tape Drive**

D. Optional Second Disk Drive

E. 1/2-inch Streamer Tape Drive**

*The extended memory boards and LCM are optional. A system may not include both types of boards.

**Each system is supplied with one streamer tape drive, either for 1/4-inch or 1/2-inch tape. The 1/4-inch tape drive is included in the RPS and is controlled by the LOTUS 730. The 1/2-inch tape drive is on a separate panel and requires a controller in the main chassis.

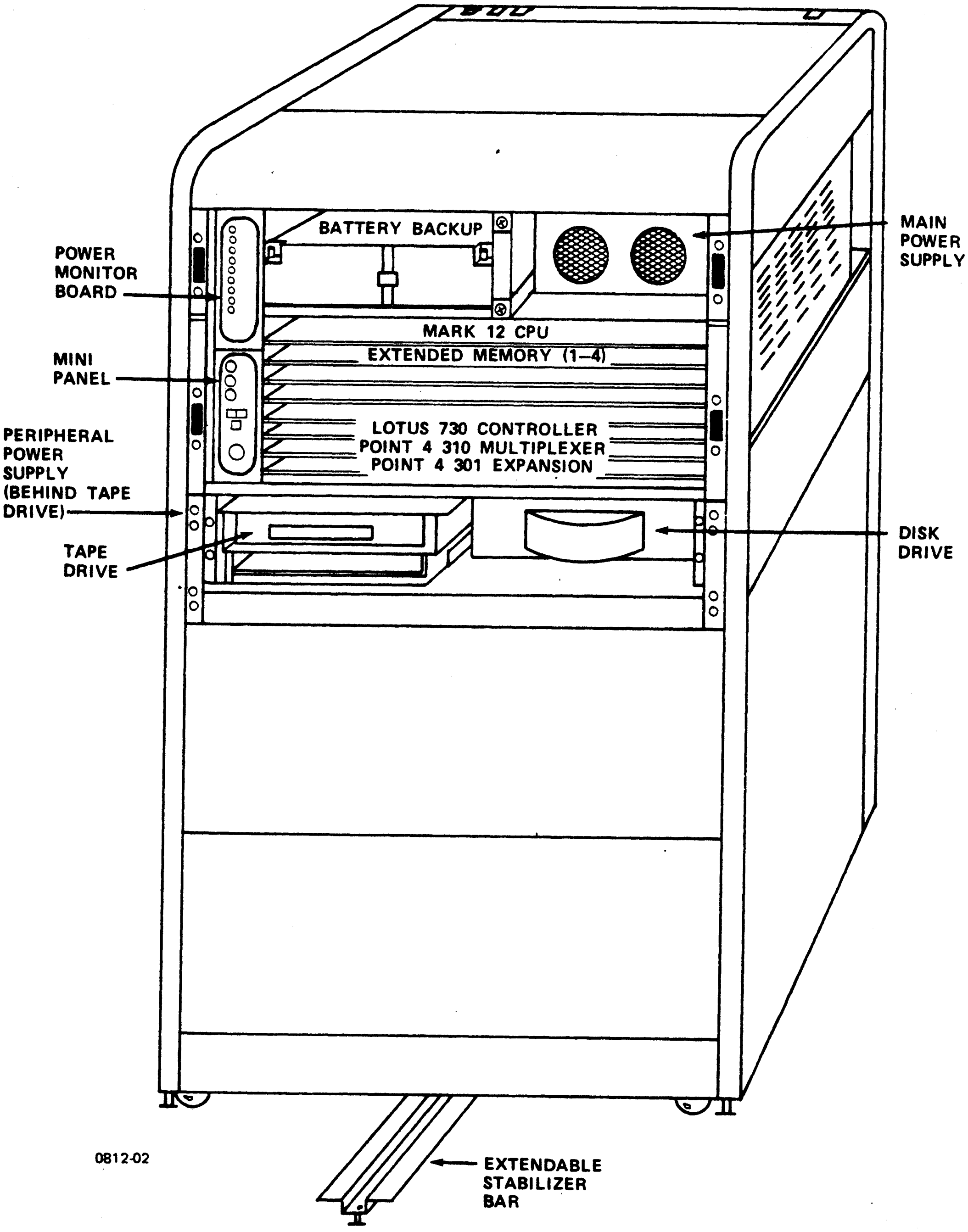


Figure 2. MARK 12 Components

III. UNPACKING THE MARK 12

The MARK 12 is shipped on a wooden pallet and is held firmly in place by four 2x4 retainer boards (three fixed and one removable). The unit is protected by styrofoam angles and a plywood board that doubles as a ramp. The assembly is covered by a cardboard carton (or a wooden crate for delivery outside the United States) held in place by straps. An Allen key is taped to the rear panel.

The recommended procedure for unpacking the MARK 12 is as follows:

1. Remove the straps.
2. Lift off the cardboard cover (or wooden crate).
3. Unbolt the removable 2x4 retainer board (see Figure 3). One of the bolts will be used to secure the ramp (step 5).

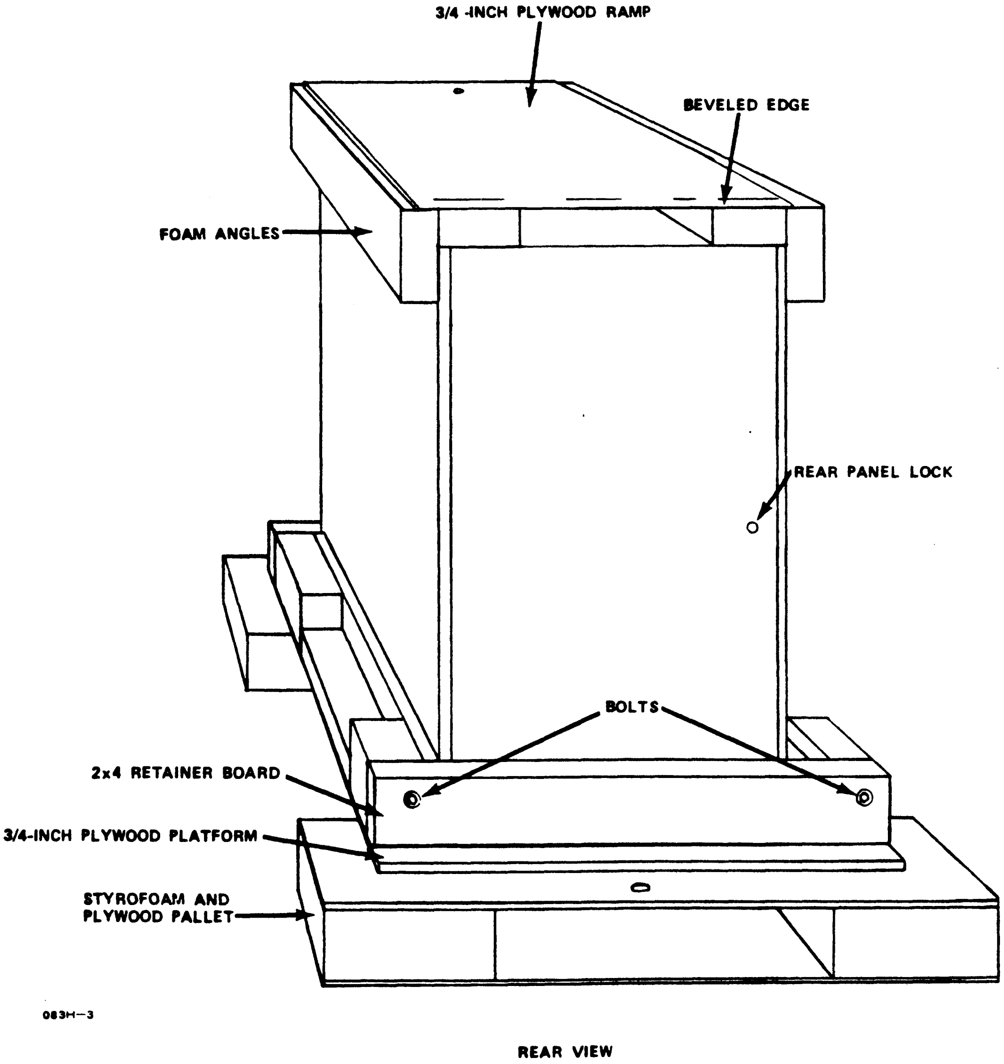


Figure 3. MARK 12 Packaging

4. Remove the plywood ramp and styrofoam protectors from the top of the unit.

5. Place the ramp on the pallet so that the hole in the ramp lines up with the hole in the pallet. Insert one of the bolts taken from the retainer board in step 3 through both holes (see Figure 4).

6. Make sure that the leveler feet are clear of the pallet and cannot impede the rollers. Roll the unit down the ramp.

7. Move the unit close to its permanent place which should allow access to the sides and rear of the unit.

8. Store the pallet, ramp, and other packing material in a safe place.

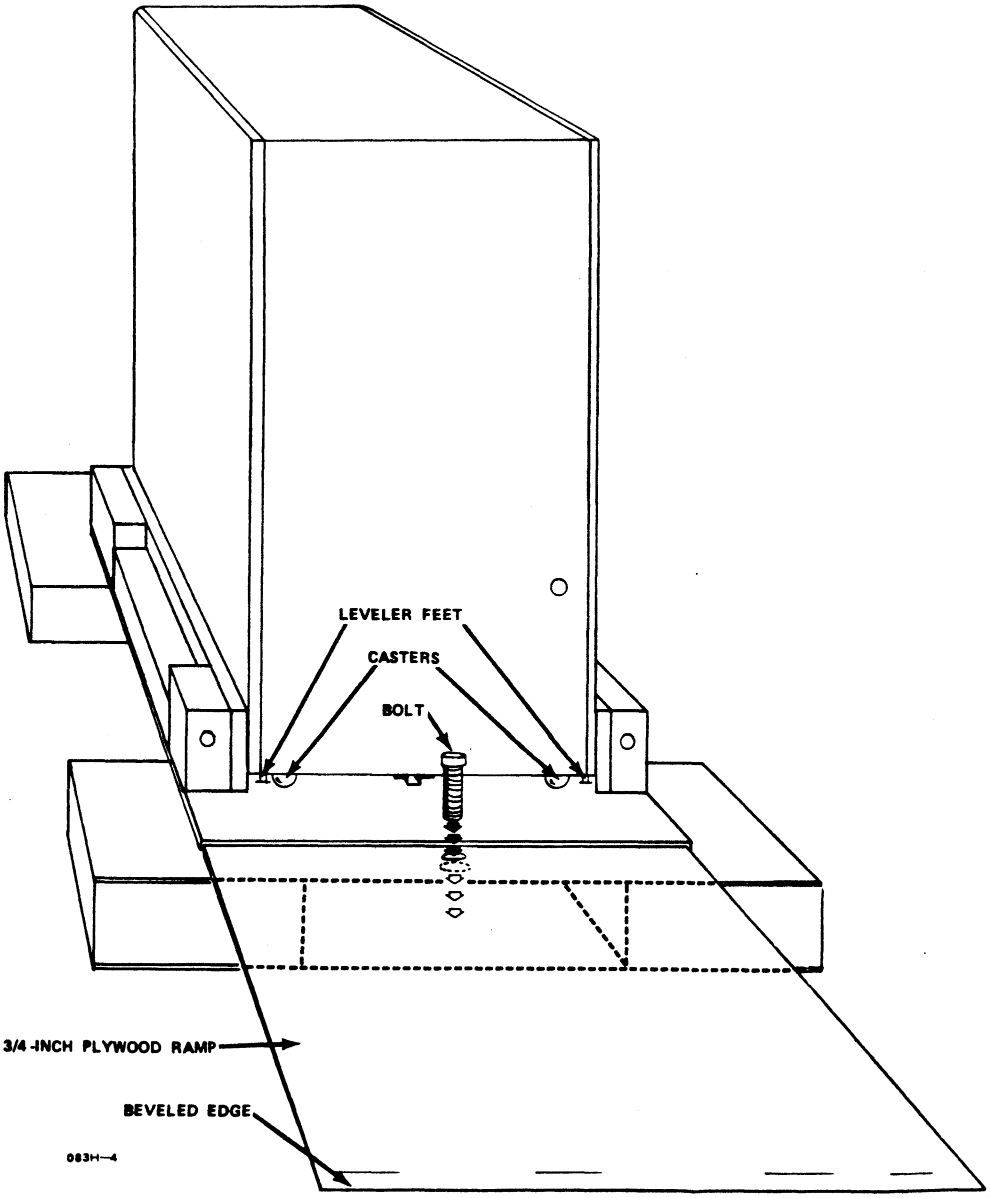


Figure 4. Ramp Installation

IV. SYSTEM INSTALLATION

The POINT 4 MARK 12 Computer System is shipped fully assembled and no additional cabling is required. The recommended installation procedure is as follows:

1. Pull out the stabilizer bar at the front of the unit to prevent it from tipping over.

2. Snap out the removable front panels (i.e., the panels covering the main power supply, the computer chassis, and the RPS).

3. Check each board in the computer chassis as follows:

   a. Slide the board out by pulling out on the metal tabs.

   b. Remove the packing from around the board.

   c. Check that the board components are firmly seated.

   d. Using the metal tabs, slide the board back into place.

4. Make sure all cables are connected.

5. Snap the front panels back into place.

6. Detach the Allen key from the rear panel. Unlock the panel by inserting the key and turning it counterclockwise. (The rear panel may be removed entirely by first detaching the green ground wire and then pulling down on the spring hinge. When replacing the rear panel, make sure that the ground wire is reconnected.)

7. Check the plug and fuse on the main power supply; make sure that the switch above them is turned to the ON position.

8. Check the connectors on the main power supply unit; make sure they are firmly seated.

9. If the Battery Backup (BBU) option is installed, remove the battery tray and plug the battery cable into connector J5 on the BBU board.

10. Check the fuse on the rear of the RPS; make sure the plugs are firmly seated and the power switch is turned to the ON position.

11. IRIS users - make sure the Pico-N is installed on the backplane.

12. Plug CRT and printer cables into the multiplexer connector panel at the back of the unit (the first connector is the leftmost one).

13. Plug the unit into the main AC power (3-pronged, grounded wall outlet). General power is controlled by a lighted on/off button at the rear of the unit (see Figure 5). If it does not light up when the unit has been connected to the main power outlet, press the button; the light should come on.

14. Close the rear panel (make sure the cables extend freely through the bottom of the door; the ground wire must be reconnected if the panel was removed entirely).

15. Push the stabilizer bar back so that it does not extend beyond the cabinet.

16. Move the unit to its permanent place.

17. Lower the leveler feet by turning them counterclockwise until the rollers are lifted off the floor.

18. Adjust the leveler feet so that the unit is level.

19. To install software, see the IRIS R8 Installation and Configuration Manual and the IRIS R8 Release Notes.

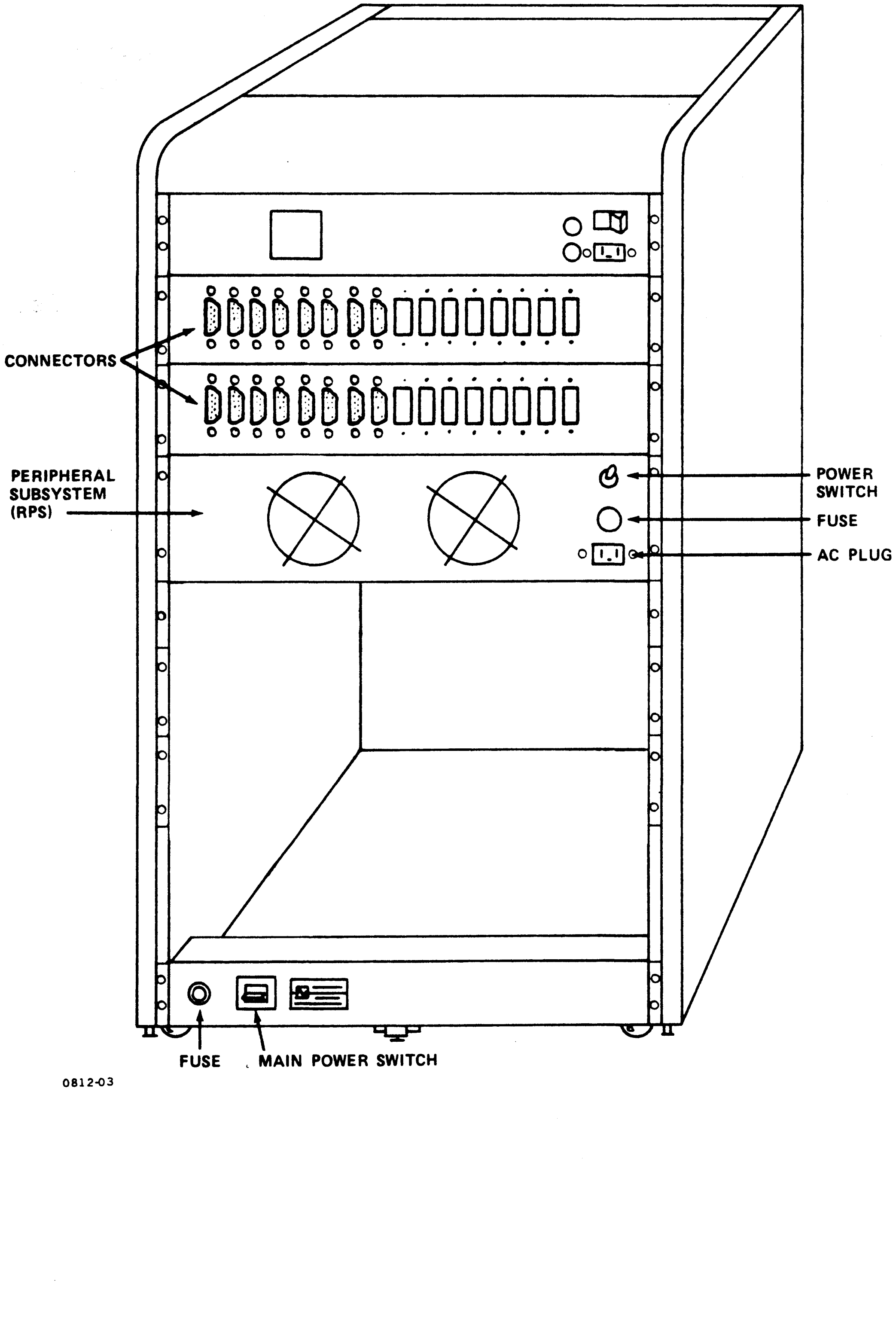


Figure 5. Rear View of Assembly

Appendix A

MARK 12 CPU AND POWER SUPPLY INSTALLATION

The following are mechanical and electrical hook-up instructions for installing a MARK 12 CPU card cage and power supply in a non-POINT 4 cabinet. See Table 1 for a list of the parts contained in the MARK 12 shipping kit.

A. Mechanical Installation

1. Install (2) chassis; 88012 power supply on top of 88013 card cage per Figure 6.

2. Install per Figure 6:

   (2) 142052 bracket supports
   (4) 707FSP 8-32 x 3/8 flat head screws
   (1) 712202 #6 washers
   (1) 707D6P #6 1/4" flat head screw
   (2) 142061 bracket

3. Install per Figure 6 **(required for UL conformity):**

   (1) 721027 cable clamp
   (1) 707FSP screw

4. (12) 709HSP 10-32 x 1/2 phillips head sems screws are available for MARK 12 assembly-to-console installation.

5. Install 082020 bezel assembly.

B. Electrical Installation - MARK 12 Cable Interconnects

Connect the cables between the MARK 12 card cage and power supply according to the following and Figure 6:

	FROM	TO
1. Loose cables in P/N 099001 MK12 shipping kit:		
(1) 088040 AC jumper	P/S J21	P/S J 5
(1) 088045 BBU jumper	C/C J 6	C/C J 6
(1) 505006 AC line cord	P/S J12	115V or 220V power source
(1) 088052 cable assy. mux pwr.	P/S J24	P/S TB1
(2) 032005 mux power supply dist. cable assembly	P/S TB1	vendors I/O pnl. (P/N 32200)
2. Cables on end to connect:		
(1) 88026 DC harness	C/C	P/S J22 pwr. sen.
	C/C	P/S P23 main pwr.
(2) 88033 display control	C/C J8	P/S J25
(3) 82015 side panel assy.	C/C	P/S J13

where
C/C = card cage assembly 82012
P/S = power supply asembly 82013

TABLE 1. MARK 12 SHIPPING KIT, P/N 099001

PART NO.	DESCRIPTION	QUANTITY				REMARKS
		01	02	03	04	
707 FSP	SCREW 6-32 x 3/8 PHIL.	1	1	1	1	
721027	CLAMP CABLE 3/8 DIA.	1	1	1	1	
082020	BEZEL ASSEMBLY	1	1	1	1	
141052	BRACKET SUPPORT	2	2	2	2	
707 EGP	SCREW 6-32 x FLAT HEAD	5	5	5	5	
709 HSP	SCREW 10-32 x 1/2 PHIL HD	12	12	12	12	
142061	BRACKET	2	2	2	2	
708 FSP	8-32 x 3/8 SEMS PHIL HD	4	4	4	4	
712240	#10 FLAT WASHER	1	1	1	1	
088040	CA. ASSY. MAIN A/C JUMPER	1	1	1	1	
088041	CA. ASSY. BBU A/C JUMPER	-	-	1	1	
088035	CA. ASSY. BBU CONTROL	-	-	1	1	
088036	CA. ASSY. BBU D/C PWR. SUP.	-	-	1	1	
505006	A/C LINE CORD	1	1	1	1	
088045	CA. ASSY. BBU JUMPER	1	1	-	-	
088052	CA. ASSY. MUX POWER	1	1	1	1	
032005	CA. ASSY. MUX P/S DIST.	2	2	2	2	

LEGEND:

099001-01	115 VAC
099001-02	230 VAC
099001-03	115 VAC WITH BBU
099001-04	230 VAC WITH BBU

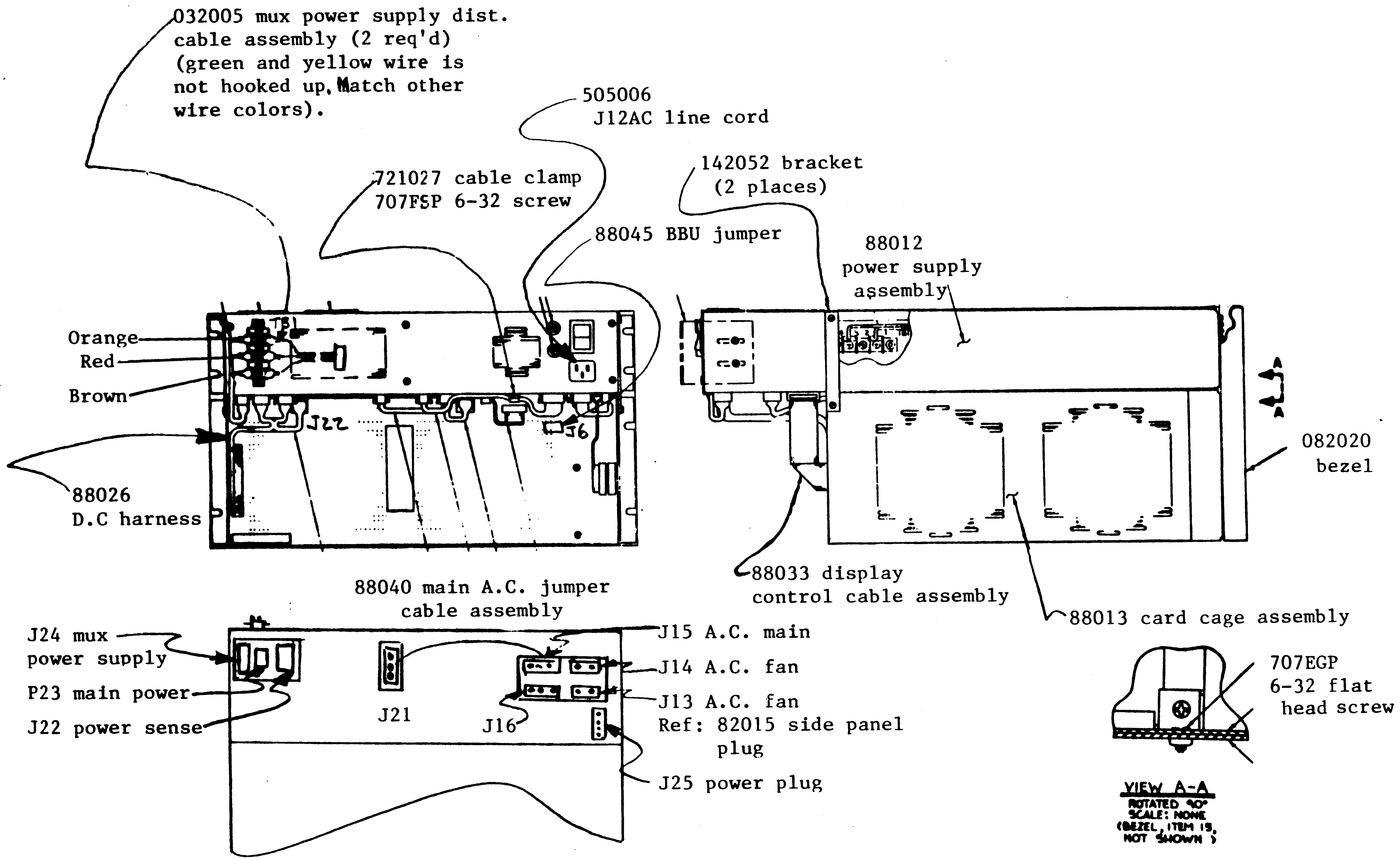


Figure 6. MARK 12 Installation Diagram

POINT 4 Data Corporation

```
       4444
    44444      4
  44444        444
 4444     4    4444
444     444    44444
4     44444    44444
```

TECHNICAL MEMORANDUM

```
 4444444444    4444
  444444444    444
    4444444    4
       4444
```

TO: MARK 12 Users

FROM: Hardware Development

DATE: June 10, 1986

SUBJ: MARK 12 COMPUTER SYSTEM (Differences from MARK 5)

This tech memo includes a description of the features of the MARK 12 computer system and describes the MARK 12 in terms of additions or deviations from the MARK 5 computer system.

CONTENTS

APPENDICES

I. INTRODUCTION

A MARK 12 Computer System consists of an 8-slot card cage, a power supply, and a MARK 12 CPU with optional extended memory. The MARK 12 is a 16-bit industry-standard CPU with 128K bytes of onboard static RAM. The MARK 12 can access up to 16MB of external memory located on up to four MARK 12 memory expansion boards, each containing from 2 to 4MB.

The MARK 12 is an extension of the POINT 4 MARK 5/8/9 product line. It is functionally equivalent to the MARK 5 except as described herein.

1.1 SYSTEM FEATURES

- Basic Unit - 8-card CPU chassis and power supply chassis (10.5-inch total rack space)
- Modular design for main power supply and battery backup (for extended memory or LCM) including Mux power supply (without current loop)
  Main P/S: +5V, +15V, -15V, -5V, +12V, -12V
  BBU P/S: +5V, +12V, -5V
- Powerfail backup capability when used with extended memory (will be available with the release of IRIS 8.4)
- Power option for 8 additional card slots
- Increased air flow for better cooling
- Designed to safety requirements of UL, CSA, VDE

1.2 CPU FEATURES

The following are additions or deviations from the MARK 5:

- 64-nanosecond effective cycle time
- Up to 16 megabytes of extended memory, packaged at up to 4 megabytes per board
- Variable size pages - any multiple of 8 bytes
- 32-bit high-speed bus accessing 16-megabyte extended memory
- Expanded Self-Test and MANIP
- Firmware-level hardware-verify and Virtual Control Function
- 3 software modes: 32KW-addressing (Nova*), 64KW-addressing (MARK 5), MARK 12

*Nova is a registered trademark of Data General Corp.

1.3 EXTENDED MEMORY FEATURES

The following are functional differences from the LCM:

- High-speed, 32-bit bus to the CPU
- Data rate - 33.3 megabytes-per-second burst transfer rate
- 2 and 4-megabyte boards with Error Detection and Correction (EDAC)
- Full battery backup option for LCM or Extended Memory only; does not backup 128KB onboard memory)

1.4 OPTIONS

System

The MARK 12 Battery Backup (BBU) Option will provide backup power to the Extended Memories and both old and new POINT 4 LCMs with the release of IRIS 8.4.

The Auxiliary Chassis provides 8 additional slots for system expansion. Power is provided by the main power supply. The combined power capabilities of the Main and BBU supplies can power most configurations.

CPU

The Universal (Switch/Indicator) option available for MARK 5/9 CPUs is standard on the MARK 12.

II. DESCRIPTION OF SYSTEM DIFFERENCES

2.1 POWER

Input Power

115V operation - 90-132V
50/60 Hz

230V operation - 180-264V
50/60 Hz

Output Power

The MARK 12 power supply provides additional current; a separate Mux power supply is no longer required.

+5V	**50 amps**
±15V	5 amps*
±12V	3 amps
-5V	5 amps

*Total of 15V and 12V currents not to exceed 5 amps

2.2 BATTERY BACKUP OPTION

The MARK 12 Battery Backup Option when available, will provide backup power to the Extended Memories and both old and new POINT 4 LCMs. The onboard memory of the MARK 12 is not directly backed up by the battery backup as with the MARK 5/8/9. With the Battery Backup Option and Extended Memory, the MARK 12 will transfer the contents of the onboard memory to and from the Extended Memory during powerfail and restart.

2.3 DATA CHANNEL

The high-speed/standard-speed data channel jumper is no longer required. The MARK 12 has a single, high-speed data channel; all standard controllers should work with the MARK 12.

2.4 DEVICE CODES 2 AND 3

The controls for Device Codes 2 and 3 do not come out on the backplane. They are reserved by the MARK 12 for internal purposes.

2.5 EXTENDED INSTRUCTION SET

The Extended Instruction Set is not available on the MARK 12.

2.6 BACKPLANE

The MARK 12 features a new backplane with improved power distribution. Improved cooling is accomplished through .7-inch spacing and improved air flow.

Slots 2 through 5 are zoned for extended memory but may be used by most devices. The extended memory bus uses backplane pins defined for memory use only.

CAUTION

> Do not install non-POINT 4 controllers, which may use the pins defined for memory use, in slots 1 through 5. Non-POINT 4 controllers can be installed in any other slot.

The Extended Memory Bus uses the following pins of slots 1-5 for access to extended memory:

A8
A-13 thru A-32
A-35 thru A-37
A-39
A-41
A-43
A-45
A-51
A-53
A-55

B-5
B-9
B-12
B-14
B-18
B-26
B-28
B-30
B-32

III. DESCRIPTION OF CPU DIFFERENCES

3.1 CPU STATUS LEDs

Sixteen small green LEDs at the front edge of the CPU board show the status of the CPU. From the left, they are:

ION - Interrupts are enabled

DME - Data Channel mapping enabled

EIS - CPU is in Extended Instruction Set mode

64K - CPU is in 64K-word addressing mode

DCR - Data Channel Request is active

11-Bit Firmware Microprogram Address

3.2 SELF-TESTING CAPABILITIES

There are two types of self-test on the MARK 12: one firmware (hardware verify), the other software (Self-Test). Both are provided in onboard ROM.

Hardware-Verify

The hardware-verify tests the full functionality of the hardware at the firmware level and ensures that the hardware is functional enough to communicate with a terminal. The CPU status LEDs give information regarding errors.

This hardware-verify capability enables the technician to obtain information about the CPU and to debug it when MANIP cannot be loaded because of a hardware problem.

An Operator Control Unit is not required for the MARK 12. Instead, this hardware-verify capability allows examining the program counter, accumulators, CPU status word and 8 words of memory when the CPU is in a Halt state. By pressing any key (except <ESCAPE> which transfers control to MANIP), the following information is displayed on the master terminal:

```
    pppppp aaaaaa bbbbbb cccccc dddddd ssssss
    vvvvvv wwwwww wwwwww wwwwww wwwwww wwwwww wwwwww wwwwww wwwwww
```

where

pppppp - value of program counter when Halt was encountered
aaaaaa - value of accumulator 0
bbbbbb - value of accumulator 1
cccccc - value of accumulator 2
dddddd - value of accumulator 3
ssssss - saved CPU status word (see Appendix A for description)
vvvvvv - value in mini-switches on front edge of CPU
wwwwww - content of 8 words of onboard memory starting at address vvvvvv

By manipulating the mini-switches, the user can read any location in onboard memory.

Software Self-Test

The software Self-Test is contained in the MANIP PROM; it can be accessed in three ways:

1. On power-on

2. Using the MANIP T command

3. By setting mini-switches located on front edge of CPU to 100200 and pressing APL switch

Power-On Sequence

On power-on, one pass of the hardware-verify function followed by four passes of the software Self-Test are performed automatically. Control then depends on the setting of the mini-switches:

- if 100200, will run Self-Test forever
- if 200 + x where x is a device code, will IPL from device x
- if other than above, control goes to MANIP

3.3 MANIP - VIRTUAL CONTROL PANEL

The MARK 12 CPU has an expanded MANIP capability. When the APL switch is depressed, a conventional MANIP is loaded into the top 1000 (octal) words of onboard memory. This MANIP in turn gives the capability of loading the remainder of the APL PROMs, or any portion thereof, to any desired location in onboard memory.

There are no separate Self-Test PROMs; instead the expanded MANIP contains Self-Test, and provides for a means of entering it.

The MARK 12 MANIP has all the MARK 5 features plus the following:

A - Displays program counter, 4 accumulators, and CPU status word (see Appendix A for description)

C - Allows modification of CPU status word, e.g.

```
C4,ssssss
```

where ssssss is the new CPU status word

J(without parameter) - Does a CONTinue function; i.e., jumps to address where last Halt occurred

P - Initializes extended memory (if any), before performing program load function

T - Loads Self-Test and begins executing it forever (must press CONTinue after initial Halt)

U - Can be used to unload DBUG from PROM into memory, e.g.

```
U1000,2000,3000
```

unloads PROM beginning at 1000, for 2000 words into onboard memory starting at location 3000

U(without parameters) - unloads DBUG and BZUD beginning at location 73000

The MARK 12 MANIP also has a signature or prompt; every line starts with

```
->
```

It differs from the DBUG signature which is

```
=>
```

IV. LCM/EXTENDED MEMORY DIFFERENCES

4.1 LCM

The new version of the LCM is fully system and software compatible with the old LCM; it has the following additional features:

- Larger capacity - 2 and 4MB
- Additional diagnostic features
- Receives +5V BBU power from the backplane; no need for additional power supply or cabling when used in the MARK 12 chassis with the MARK 12 power supply (the LCM can still be powered in the old way if used in a non-MARK 12 chassis)

4.2 EXTENDED MEMORY

The new MARK 12 Extended Memory is not compatible with the old LCM system. It can be used only with a MARK 12 CPU in a MARK 12 chassis. No extra cabling is required for battery backup; it receives +5 BBU power from the backplane. It is offered in 2 and 4MB boards.

Appendix A

CPU STATUS WORD

Bit	Symbol	Name and Significance
0*	CY	Carry flag in saved CPU status word. Always 0 in current status. If set by software to a 1, along with MSK=0, will set 64KW-addressing mode and enable EIS (if implemented), ignoring all other bits in the NIOP instruction.
1	MSK	Mask. If set to a 0 by software in a NIOP, bits 2-14 will be ignored. If set to a 1, enable bits 2-14. Always set to 1 during a read.
2	AUT	Auto Restart flag. Set by hardware to a 1, in saved CPU status word, on power up, if front panel keyswitch is in the AUTO position. General purpose flag otherwise.
3		Reserved.
4		Reserved.
5		Reserved.
6		Reserved.
7	TSC	Trap Self-modifying Code control bit. Set by software to a 1 to enable.
8	PEL	Front panel Parity Error Light control bit. Set to a 1/0 by software to turn parity error light on/off.
9	M12	MARK 12 mode control bit. If set by software to a 1, along with 64K=1, will enable MARK 12 mode.
10	-	Reserved.
11	RNL	Run flag. Set to a 1/0 by hardware to indicate Run/Halt state. Also set by hardware to a 0, in saved CPU status word, at power up.
12	ION	Interrupts Enabled flag. Set by hardware only to indicate interrupts enabled.
13	DME	Data channel Map Enable flag. Set by software to a 1 to enable data channel mapping.
14	-	Reserved for EIS enablement.
15	64K	64KW-addressing mode control bit. Must set to a 1 to enable M12 mode or TSC control bit.
*Most significant bit		

Chapter 2 - FUNCTIONAL DESCRIPTION

CONTENTS

CHAPTER 2

FUNCTIONAL DESCRIPTION

2.1 GENERAL

The MARK 12 is a 16-bit Data General Nova*-compatible CPU with 128K bytes of onboard static ram. In addition, the MARK 12 can access up to 16 megabytes of external memory located on MARK 12 memory expansion boards, each containing from 2 to 4 megabytes. The CPU uses a pipelined architecture, with a four instruction prefetch queue, and an autonomous Data Channel processor.

The MARK 12 CPU is an extension of that part of POINT 4 Data Corporation's product line that consists of the MARK 5, MARK 8, and MARK 9 computers. It uses the same size printed circuit board, and is compatible with the same peripheral controllers.

2.2 MARK 12 CPU SIGNALS

Signals common to the Data General Nova are defined in the Nova User's Manual (see list of reference documents). Signals unique to the CPU are described below. All are low-active signals, indicated by the "-" suffix.

2.2.1 32-BIT HIGH SPEED BUS TO EXPANSION MEMORY

The MARK 12 CPU features a 32-bit high speed synchronous extension memory bus used to access expansion memory boards in the MARK 12 system. Data is transferred on this "V-Bus" along 32 parallel, bidirectional, data lines. Control and status signals are carried along dedicated, unidirectional, control and status lines. This structure uses 41 pins on the 200-pin Nova backplane. The pins used are all taken from the set of pins reserved by the standard Data General Nova pinout for memory lines, that is, pins which are not permitted to be used by custom interface boards.

The 41 pins are defined as follows:

2.2.1.1 SYSTEM CLOCK (SYSCLK-)

A 12.5 MHz (nominal) signal from the CPU used to synchronize the optional expansion memory to the CPU, and to synchronize the transfer of data. The memory uses only the low-going edge of this signal.

2.2.1.2 DATA/ADDRESS LINES (V0-/V31-)

32 bi-directional lines used to transfer both addresses (from CPU, strobed by AS-) and data (from CPU, strobed by DS-, or from expansion memory, strobed by DTACK-). V0 is the most significant bit (MSB).

2.2.1.3 ADDRESS STROBE (AS-)

This signal is generated by the CPU when a new expansion memory block address is on the V1-/V15- lines. A single address strobe may be followed by a sequence of data strobes.

2.2.1.4 DATA STROBE (DS-)

This signal is generated by the CPU to indicate that valid data is present on the V0-/V31- lines during an expansion memory write cycle, or that more data is requested during a read cycle. Each data strobe causes two successive 32-bit transfers for a 64-bit quadruple word transfer; and up to 64 such pairs, separated by at least one idle state (80 nanoseconds), may be used to transfer up to a 256-word page to or from memory.

Note: When both AS- and DS- are activated at the same time, the meanings of V0- and V8-/V15- are as follows:

The V8-/V15- lines hold a word address offset into the memory page addressed by the AS-only transfer. This is used for single-word transfers (DMA) and for partial page transfers (less than 256 words).

The V0- line, if active, indicates that the EDAC logic on the expansion memory is to be bypassed, for diagnostic purposes only.

2.2.1.5 WRITE DATA (WRITE-)

A control signal generated by the CPU and indicating that data is to be written to expansion memory. When this signal is inactive (high), data is to be read from memory.

2.2.1.6 LONG WORD (LWORD-)

A signal generated by the CPU indicating that the current transfer is a part of a multi-word transfer, transferring 32 bits at a time. When this signal is inactive (high), and WRITE- is active, it means to write one word only to the specified word address. If WRITE- is inactive, it means to read the expansion memory status word onto the V0-/V15- lines (refer to Section 2.3.5.3.1).

2.2.1.7 BUS BUSY (BBSY-)

This signal is an open-collector signal generated by the expansion memory in response to the AS- signal if the memory contains the block address on the V1-/V15- lines.

2.2.1.8 DATA TRANSFER ACKNOWLEDGE (DTACK-)

Data Transfer Acknowledge; an open-collector signal generated by the memory to indicate that valid data is available on the bus during a read cycle, or that data has been accepted from the bus during a write cycle.

2.2.1.9 BUS ERROR (BERR-)

An open-collector signal generated by expansion memory to indicate that the data read from memory contains an error, either correctable or not, and that correction will be automatically attempted.

2.2.1.10 BUS ERROR CORRECTION ATTEMPT (BERC-)

An open-collector signal generated by the expansion memory to indicate that a data correction attempt is being made.

BERR-	BERC-	
H	H	No error
L	H	Some error, correction will be attempted
H	L	Correction was successful
L	L	Correction attempt was unsuccessful; i.e., a multi-bit error occurred

2.2.2 POWER FAIL/POWER GONE SIGNALS

These two signals must be generated by the power supply providing power to the MARK 12 CPU.

2.2.2.1 POWER FAIL (PWRF-)

The Power Fail signal is an early warning indication to the MARK 12 CPU of impending power failure. This signal must occur at least 5 milliseconds before DC power goes out of regulation.

When PWRF- goes active, a non-maskable program interrupt is generated which allows the software to gracefully shut down the entire system. If the system in which the CPU is installed also contains MARK 12 memory boards with battery backup, this shutdown would also include copying the contents of CPU memory to expansion memory.

2.2.2.2 POWER GONE (PWRGON-)

PWRGON- is asserted by the CPU chassis power supply to indicate to the MARK 12 CPU that DC power is now out of regulation. When received by the CPU, this signal will result in a CPU hardware reset. PWRGON- must be asserted for 1 microsecond minimum, or as long as power remains out of regulation. When power is restored, after a power failure or brownout, PWRGON- must remain asserted until DC power is within regulation.

2.3 PROGRAMING

2.3.1 OVERVIEW

The MARK 12 CPU software interface is generally compatible with the basic instruction set of the POINT 4 MARK 5 CPU, in both 32K Word and 64K Word addressing modes. The remainder of this section documents the differences from those standards.

2.3.2 SOFTWARE MODES

The MARK 12 CPU has a number of operational modes, which are controlled by a set of mode bits. The use of mode bits, combined in a 16-bit CPU Status Word, makes for a very powerful and structured way of setting, saving, restoring, and displaying the CPU's operating mode.

2.3.2.1 32K WORD ADDRESSING MODE

The 32K Word addressing mode is available in the MARK 12 CPU to allow the user to emulate the original Data General Nova CPUs that had only 64KB of available memory. Indirect addressing chains (see below) and auto-increment/decrement locations (see Section 2.3.2.1.2) are functional in this mode. Data Channel Mapping (see Section 2.3.5.3.2) may be enabled as necessary and unrestricted "self-modifying code" (refer to Section 2.3.2.1.3) is also allowed. In 32K mode there is an attendant performance loss due to the additional overhead of 32K mode emulation.

After an IORST instruction is executed, the CPU is automatically placed in 32KW mode, with Data Channel Mapping disabled. 32KW mode can also be entered (with Data Channel Mapping unaffected) by executing an NIOP CPU instruction with 0 in A0; see Section 2.3.4 below.

2.3.2.1.1 INDIRECT ADDRESSING

With the MARK 12 CPU in 32KW mode, all memory references are limited to 15 bits of address, with the most significant bit ignored in direct addressing and used for multi-level indirection in indirect addressing. The MARK 12 CPU provides no protection against infinite indirection.

2.3.2.1.2 AUTO INDEXING

If an indirect address points to a location in onboard memory in the range 20-27 (octal), that word is fetched, the contents of the word is incremented by one and written back into the location. This updated value is then used to continue the addressing chain. The locations 20-27 are referred to as the auto-increment locations. If an indirect address points to a location in the range 30-37 (auto-decrement locations), that word is fetched, the contents of the word is decremented by one and written back into the location. The updated value is then used to continue the addressing chain. The POINT 4 IRIS Operating System does not use this capability; the MARK 2/3/4 series of CPUs do not implement it.

In the MARK 12 CPU, auto-increment/decrement locations are implemented only when in 32KW or 64KW mode. In MARK 12 mode there is no Auto Indexing.

2.3.2.1.3 SELF-MODIFYING CODE

Since the MARK 12 CPU uses an instruction prefetch scheme, there is a potential problem if the software program being executed attempts to modify an instruction which is imminently to be executed. If the instruction is already in the Instruction Prefetch Queue (see Section 3.3), the modified instruction will be stored into onboard memory, but the unmodified instruction will be executed.

Of course the CPU hardware/firmware could always test for this condition in all Store and ISZ/DSZ instructions, but the performance penalty would be noticeable.

The problem is resolved in the MARK 12 CPU by allowing unrestricted self-modifying code when the CPU is running in either 32K or 64K (see Section 2.3.2.2) mode, since neither of these modes are considered as the preferred operational mode.

In normal MARK 12 mode, it is illegal to modify an instruction and then flow into it, ie., proceed to execute the modified instruction without an intervening JMP or JSR (any Jump instruction reloads the Instruction Prefetch Queue).

2.3.2.2 64K WORD ADDRESSING MODE

In this mode, all 16 bits are used as a word address, allowing access to 64K Words, or 128K Bytes of onboard memory. As a result, multi-level indirect addressing is not available in this mode. However, auto-increment/decrement locations are functional and self-modifying code is also allowed. Data Channel Mapping may be enabled or disabled as desired.

As an aid to systems software developers, there is an additional feature available to the user which is not available in either of the other two operational modes. That is, the ability to force the CPU to generate a software trap when instruction sequences are encountered which are not allowed while in the MARK 12 mode of operation; namely, self-modifying code and code which utilizes any of the sixteen auto-increment/decrement registers. By setting/clearing the TRP bit in the CPU Status Word (see Section 2.3.4), this diagnostic function can be enabled or disabled.

With the TRP bit set, a software trap will be executed whenever a STA, ISZ or DSZ instruction has an effective address which is 1-5 words in front of itself, or if a STA or LDA instruction addresses any of the sixteen auto-increment/ decrement registers. The trap is executed as follows: The two words preceding the program interrupt service routine, to which there is a pointer contained in onboard memory at location 1, are used. The contents of the updated program counter at the time the trap occurred is stored in the first of these, and the second is used as a vector to the trap handler. It is then up to the trap handler to determine if there is an intervening JMP or JSR or if the self-modifying code restriction is violated, and to take any appropriate action.

2.3.2.3 EXTENDED INSTRUCTION SET (EIS) MODE

In EIS mode the class of ALU instructions having both the No-Load bit set and the Unconditional-Skip code set, is redefined as a set of macro-instructions implementing such functions as:

Binary Multiply and Divide

Decimal (BCD) Add and Subtract

Byte Access: GETBYTE and PUTBYTE

Immediate Compare, Load, Add and Subtract

Bit Test and Modify instructions

Block Move

Table Search

Stack instructions

<u>In the MARK 12, the EIS mode is disabled. However, the hardware exists to implement it at a later date.</u>

2.3.3 CPU STATUS WORD

The MARK 12 CPU Status Word is defined as shown in table 2-1. CPU Status can be set by an NIOP CPU instruction; it is read by a DIAP ac,CPU instruction with (ac) = 2 (see Section 2.3.5.1). Further, the MARK 12 hardware saves the current CPU status in an auxiliary register ("saved status") whenever the HALT state is entered. Saved status is read by a DIAP ac,CPU instruction with (ac) = 1. Refer to Section 2.3.6 for the default condition of the status word at power up time. Bit 0 is the MSB.

TABLE 2-1 CPU STATUS WORD

Bit	Symbol	Name and Significance
0	CY	Carry flag in saved CPU status word. Always 0 in current status. If set by software to a 1, along with MSK = 0, will set 64K mode and enable EIS (if implemented), ignoring all other bits in the NIOP instruction.*
--		
1	MSK	Mask. If set to a 0 by software in a NIOP, bits 2-14 will be ignored. If set to a 1, enable bits 2-14. Always set to 1 during a read.
2	AUT	Auto Restart flag. Set by hardware to a 1, in saved CPU status word, on power up, if front panel keyswitch is in the AUTO position. General purpose flag otherwise.
3	-	Reserved.
--		
4	-	Reserved.
5	-	Reserved.
6	-	Reserved.
--		
7	TSC	Trap Self-Modifying Code control bit. Set by software to a 1 to enable. See section 2.3.2.4 for detailed explanation.
8	PEL	Front panel parity error light control bit. Set to a 1/0 by software to turn parity error light on/off.
9	M12	MARK 12 Mode control bit. If set by software to a 1, along with 64K=1, will enable MARK 12 Mode.
--		
10	-	Reserved.

*See Notes on page 2B-12.

TABLE 2-1 CPU STATUS WORD (CONT.)

Bit	Symbol	Name and Significance
11	RNL	Run flag. Set to a 1/0 by hardware to indicate Run/Halt state. Also set by hardware to a 0, in saved CPU status word, at power up.
12	ION	Interrupts Enabled Flag. Set by hardware only to indicate interrupts enabled.
--		
13	DME	Data Channel Map Enable Flag. Set by software to a 1 to enable Data Channel mapping.
14	-	Reserved for EIS enable bit.
15	64K	64K Addressing mode control bit. Must set to a 1 to enable M12 or TSC control bit.

Note: Bits 0 and 1 in the NIOP case are defined so as to allow compatibility with existing code, since the MARK 5/8/9 CPUs have used the NIOP with the following values:

000001	=	Set 64K mode
100001	=	Set EIS and 64K mode
177777	=	Set EIS and 64K mode
000000	=	Set 32K mode, no EIS

Bits 0 and 1 in the DIAP case are defined so that they do not need to be changed for a subsequent NIOP instruction in the new format.

Note: To set the CPU Status Word, the customary way is to use an NIOP CPU instruction, with the appropriate value in accumulator 0. Alternatively, a different accumulator may be used, although the assembler will then not accept the NIOP mnemonic.

2.3.5 SPECIAL CPU ACCESS INSTRUCTIONS

2.3.5.1 READ HARDWARE/FIRMWARE IDENTIFICATION, STATUS WORD, ETC.

In the standard MARK 5 CPU instruction set, a DIA ac, 77 instruction reads the 16 mini-switches on the front edge of the CPU board into the specified accumulator. If the DIA instruction is accompanied by a Start or Clear pulse, it will simultaneously enable or disable interrupts. The Pulse signal does not have any effect in I/O instructions to device 77 (CPU).

In the MARK 12 CPU, all the above works exactly the same, except that the DIAP instruction has a new definition:

```
DIAP     ac,77
```

returns a value in the specified accumulator which depends on the original content of that accumulator.

Entry	Value Returned
0	Hardware/Firmware Identification
1	CPU Status Byte saved when HALT state is entered
2	Current CPU Status Byte
>2	Same as 0

The Hardware/Firmware Identification word is defined as follows:

Left byte = Hardware ID = 12 (octal), defines
MARK 12 CPU
Right byte = Firmware ID = 1 for initial release

The CPU Status Word is defined in Section 2.3.4.

2.3.5.2 LOAD PROGRAM FROM APL PROM

The NIO CPU instruction is used to read in any program contained in the APL PROM on the CPU board and to begin executing it.

A0 = Address in the APL PROMs from which the program will be loaded

A1 = Number of words to be loaded

A2 = Onboard memory address where the program will be loaded

A3 = Starting address in onboard memory to begin program execution

All accumulators are preserved.

2.3.5.3 EXPANSION MEMORY ACCESS

2.3.5.3.1 CPU ACCESS TO EXPANSION MEMORY

The MARK 12 CPU uses four pairs of I/O instructions to access expansion memory; one for multiple word transfers to or from onboard memory, one for single "block" transfers to or from onboard memory, and two for single word transfers to or from accumulator 2. One of the two pairs of I/O instructions used to transfer single words also allows Data Channel Map translation. However, all other expansion memory access instructions have no map translation. In all of the following descriptions, the mnemonic "MEM" represents the device code, which is 2 for expansion memory access.

2.3.5.3.1.1 MULTIPLE WORD TRANSFERS

DIA and DOA are used to transfer multiple words between expansion memory and onboard memory. The specifics are detailed as follows:

DIA ac,MEM Read from expansion memory to onboard memory.

DOA ac,MEM Write from onboard memory to expansion memory.

In both of the instructions above, all four accumulators have prescribed meanings, regardless of which accumulator is coded in the instruction. They are,

A0 = Number of words to transfer (maximum 400 octal). This value must be a multiple of 4; i.e., the two least significant bits are ignored. If (A0) = 0, no transfer is done.

A1 = Block number in expansion memory, where a block is defined as 256 words. The block number may range up to 77777 (octal) for 16 MB addressing.

A2 = Onboard address; i.e., address in onboard memory from which data is to be transferred. Must be a multiple of 4; i.e., the two least significant bits are ignored and taken as 00.

A3 = Bits 8-15: Word address offset into the block in expansion memory addressed by A1, where the transfer is to begin. Must be a multiple of 4; i.e., the two least significant bits are ignored and taken as 00.

Bit 0: If the msb of A3 is 1, the Error Detection and Correction circuitry is disabled. On a read, the data is read regardless of any error indication and no attempt at error correction is made. On a write, only the new data is written to the addressed quadruple-word in expansion memory, preserving the previously existing Error Correction bits.

Note: Transfer may not overlap from one block into the next. In other words, the sum (A0) + (A3) must not exceed 400 octal.

2.3.5.3.1.2 BLOCK TRANSFERS

DIB and DOB are used to transfer entire blocks (256 words) between expansion memory and onboard memory. The specifics are detailed as follows:

DIB ac,MEM Read a block from expansion memory to onboard memory.

DOB ac,MEM Write a block from onboard memory to expansion memory.

In both the above, A1 and A2 have prescribed meanings, regardless of which accumulator is coded in the instruction.

A1 = Block number in expansion memory

A2 = Onboard memory address

Note: Error Detection and Correction cannot be disabled as with the DOA/DIA instructions.

In all four of the above instructions, after execution the accumulators will be as follows:

A0 = Expansion memory status word (see Table 2-2)

A1 = Unchanged

A2 = Next onboard memory address, i.e., A2 is incremented by the number of words transferred

A3 = Unchanged

2.3.5.3.1.3 SINGLE WORD TRANSFERS

DIC and DOC are used to transfer single words between expansion memory and accumulator 2. The specifics are detailed as follows:

DIC ac,MEM Read one word from expansion memory into accumulator 2.

DOC ac,MEM Write one word from accumulator 2 into expansion memory.

In both of the instructions above, A1 and A3 have prescribed meanings, regardless of which accumulator is coded in the instruction. They are,

A1 = Block number containing the desired word in expansion memory

A3 = Bits 8-15: Word address offset of the desired word into the block in expansion memory addressed by A1.

Note: Error Detection and Correction cannot be disabled as with the DOA/DIA instructions.

In both of the above instructions, after execution the accumulators will be as follows:

A0 = Expansion memory status word (see Table 2-2)

Note: The status word returned is 0 if there are no error conditions to report; i.e., if the lsb = 0 then the entire status word is equal to zero.

A1 = Unchanged

A2 = The word read from expansion memory when using DIC; unchanged when using DOC.

A3 = Unchanged

2.3.5.3.1.4 SINGLE WORD TRANSFERS WITH DATA CHANNEL MAP TRANSLATION

DIC/DOC ac,MAP are used to transfer single words between expansion memory and accumulator 2 using the Data Channel Map address translation features (refer to 2.3.5.3.2). The specifics are detailed as follows:

DIC ac,MAP Read one word from expansion memory into accumulator 2 with address translation. The mnemonic "MAP" represents the device code, which is 3 for Data Channel Map access.

DOC ac,MAP Write one word from accumulator 2 into expansion memory with address translation.

In both of the instructions above, A3 has a prescribed meaning, regardless of which accumulator is coded in the instruction. It is,

A3 = Logical address of the desired word in onboard or expansion memory, depending on the contents of the Data Channel Map.

Note: Error Detection and Correction cannot be disabled as with the DOA/DIA instructions.

After execution, the accumulators will be as follows:

A0 = Expansion memory status word (see Table 2-2), or 0 if word referenced is located in onboard memory, or if there are no error conditions to report.

A1 = Block address as read from Data Channel Map (refer to Section 2.3.5.3.2).

A2 = The word read from expansion memory when using DIC; unchanged when using DOC.

A3 = Unchanged

TABLE 2-2 EXPANSION MEMORY STATUS WORD

Bit	Symbol	Name and Significance
0	--	Reserved.
1	DON	Done Flag. Set by hardware to a 1 to indicate that the previous transfer was accomplished. A 0 indicates that no memory board with the specified block number has responded. In this case, ERR is also set to a 1.
2	EDC	Error Detection and Correction (EDAC) Option Flag. Set by hardware to a 0 if the EDAC option is installed.
3-10	C0/C7	These are the 8 EDAC check bits written or read along with the 64 data bits. Used for diagnostic purposes only.
11	VIR	Virgin Flag. This bit is set to a 1 by a DOA, DOB or DOC instruction (i.e., a write to expansion memory), and cleared by hardware at expansion memory power-up time. At Power-Fail Auto-Restart time, if this bit is 0, it indicates that Battery Back-up was not successful.
12	BAK	Battery Back-up Option Flag. Set by hardware to a 1 if the expansion memory BBU option is installed.
13	SER	System Error Flag. Set by hardware to a 1 to indicate that an error has occurred on a mapped data channel input (to write 1 word, the expansion memory must first read 4 words). Cleared by any instruction that reads the expansion memory status.
14	CER	Correctable Error Flag. Set by hardware to indicates that a correctable error has occurred on the last transfer; the error was corrected both in onboard memory and in expansion memory. Cleared by an instruction that reads the expansion memory status.

(from MARK 12 CPU Product Specification)

TABLE 2-2 EXPANSION MEMORY STATUS WORD (CONT.)

Bit	Symbol	Name and Significance
15	ERR	Error Flag. Set by hardware to a 1 to indicate that there has been an error on the last transfer. If Bit 14 is 0, it indicates that the error was uncorrectable. Cleared by any instruction that reads the expansion memory status read.

If it is desired to read the expansion memory status word without doing any memory transfer, this can be accomplished by a DIA ac,MEM instruction with zero in A0.

Any other I/O instructions with device code MEM will have no effect on the expansion memory.

2.3.5.3.2 DATA CHANNEL ACCESS TO EXPANSION MEMORY - DATA CHANNEL MAP

The MARK 12 CPU features a Data Channel Map which allows arbitrary Data Channel access to onboard memory, and block-mapped Data Channel access to expansion memory.

The Data Channel Map is actually a 256 word x 16 bit memory, where each word corresponds to one 256-word block in logical addressing space, and 15 of the 16 bits define the physical "block address" corresponding to that logical block. The 16th bit indicates whether a Data Channel transfer is to go into or out of onboard memory or expansion memory.

On power-up and after an IORST instruction, Data Channel mapping is disabled, i.e., all Data Channel transfers will go into or out of onboard memory without address translation. This is done so that the standard disk bootstrap routine will bring the initial block into onboard memory, where it can be executed.

When a Data Channel request occurs and mapping has been enabled by setting the Data Channel Mapping Enable bit in the CPU Status Word (see Section 2.3.4), the CPU reads the 16-bit address furnished by the requesting device, and applies the 8 most significant bits thereof as an address into the Data Channel Map. If the content of that word in the Data Channel Map has the most significant bit set, the Data Channel transfer is done into or out of onboard memory, otherwise it goes to expansion memory for the transfer. The 15 bits from the Data Channel Map are then combined with the 8 least significant bits of the given 16-bit address to form a 23-bit address, and the Data Channel transfer is done to or from the translated address.

If the software wishes to have Data Channel Mapping enabled, it must first set up the Data Channel Map appropriately, and then set the Data Channel Map Enable mode bit in the CPU Status Word.

For each block (256 words) in logical addressing space that is to be mapped into expansion memory, the following instruction must be executed:

DOB ac,MAP

where (regardless of "ac"),

MAP = 3 (device code)

A1 = M*100000 (octal) + B

where M = 0 for transfers to or from expansion memory

M = 1 for transfers to or from onboard memory

B = block address (up to 77777 octal for expansion memory or 377 for onboard memory)

A2 = Bits 0-7 = Logical block address given by the Data Channel device, i.e., bits 0-7 of the Data Channel address. Bits 8-15 = Ignored.

For any block that is not to be mapped, or to disable mapping after it has been enabled, use the same instruction to vector the block to itself; i.e., right half of A1 = left half of A2.

No warning is given here if expansion memory with the specified block number does not exist on the system. If Data Channel access to non-existent memory is later attempted, input will be discarded, and output will be zero-filled.

To read the content of the Data Channel Map, use a DIB ac,MAP with (A2) as above, and it will return the Map content in A1.

2.3.6 POWER ON INITIALIZATION

When power is first applied to the CPU, the firmware goes through the following initialization sequence:

It performs a firmware self-test routine.

It sets all accumulators and the carry bit to zero.

It sets the program counter to zero.

It sets the 64K mode bit and clears the ION, DME, and RNL bits in the CPU status word. It generates an IORST pulse on the I/O bus to initialize all peripheral controllers.

If the key switch on the front panel is in the Auto position, it sets the AUT bit in the CPU status word to 1, otherwise it clears it.

It then loads MANIP (see Section 2.4) into the top 1000 words (octal) of onboard memory and begins to execute it.

From this point on, the MANIP (software) program controls the operation of the CPU. MANIP functions relevant to power-up are as follows:

Read saved CPU status word and isolate these 2 bits:

RNL bit -- indicates power-up if 0, else HALT or STOP.

AUT bit -- indicates keyswitch in Auto position, on power-up.

Then perform the following:

```
IF (RNL bit = 0)                          {power-up}
THEN Run four passes of software self-test -- if
     error, HALT,

     IF (AUT bit = 1)            {keyswitch = Auto}
     THEN Read expansion memory status word,

          IF (status = Memory_content_valid)
          THEN Copy block 0 from expansion memory
               to onboard memory page 0, and
          JMP 377,                    {Auto restart}
          ELSE Type a Bell, Backslash, expansion
          memory status word, followed by a CR/LF,
          and await input

     ELSE (AUT bit = 0)            {keyswitch = On}
     Read mini-switches (see 2.5.2) --> X,

          IF (200 < X < 277)
          THEN do IPL bootstrap using device code
               X-200.

          IF (X = 100200)
          THEN load software self-test and execute
          it forever,

          ELSE Type a CR/LF, and await input.
```

Note that expansion memory is not zeroed out at power-up time. This is deferred to some other MANIP function, in order to allow examining expansion memory in case there is some doubt as to the validity of the memory status byte.

Section 3
OPERATING PROCEDURES

3.1 INTRODUCTION

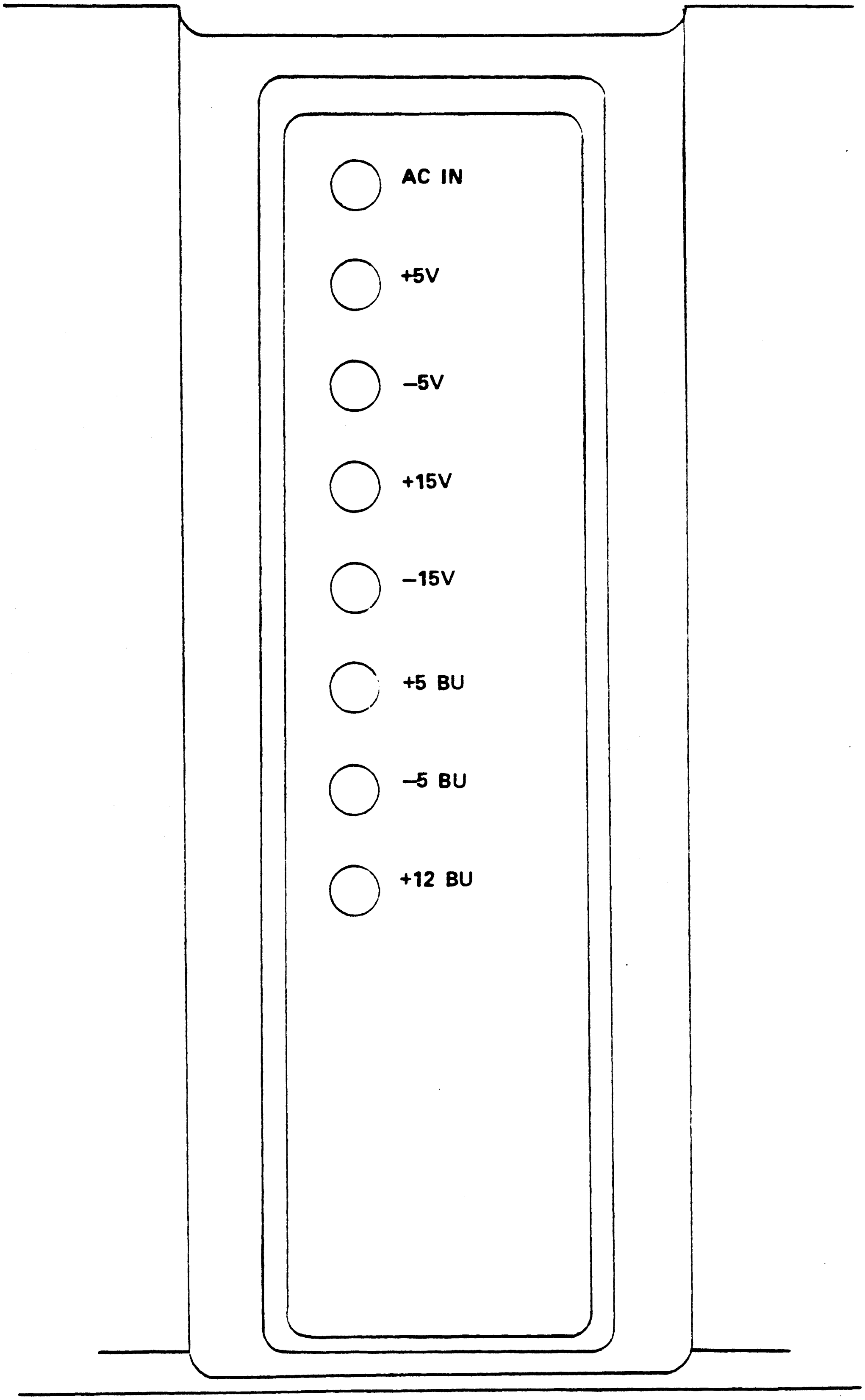
This section describes the capabilities and operating procedures for the power supply indicator panel and operator control units. Three types of control units exist for the POINT 4 MARK 8:

- Processor Mini-Panel
- Detachable Operator Control Unit (optional)
- Virtual Control Panel

Specific procedures for performing common types of operations are provided. The controls and indicators are also described.

3.2 POWER SUPPLY INDICATOR PANEL

The POINT 4 MARK 8 power supply chassis houses a set of light-emitting-diode indicators. These indicators monitor power supply voltages and Battery Backup voltages. The power supply indicator panel is located on the left-hand side of the POINT 4 MARK 8 power supply chassis. Figure 3-1 is an illustration of the power supply panel indicators.



082–12

Figure 3-1. Power Supply Indicator Panel

3.2.1 PANEL INDICATORS

Table 3-1 explains the meaning of illuminated power supply panel indicators.

TABLE 3-1. POWER SUPPLY PANEL INDICATORS

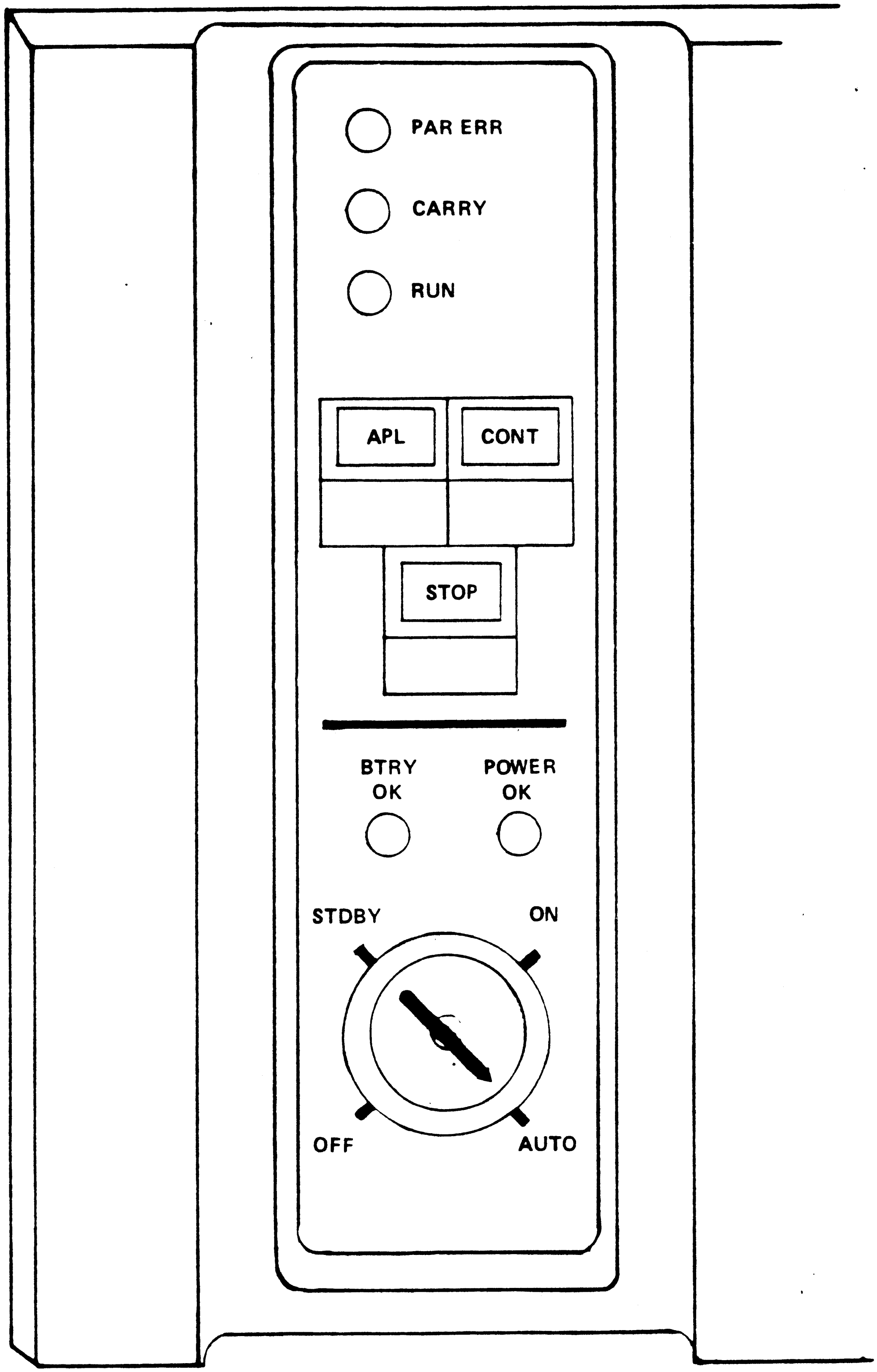
Indicator	Meaning
AC IN	Indicates that AC power has been applied to the power supply unit.
+5V	The +5V output voltage is in tolerance. This output voltage is available for user applications.
-5V	The -5V output voltage is in tolerance. This output voltage is available for user applications.
+15V	The +15V output voltage is in tolerance. This output voltage is available for user applications.
-15V	The -15V output voltage is in tolerance. This output voltage is available for user applications.
+5 BU	The +5V battery backup supply is in tolerance. This output voltage is available only to the POINT 4 MARK 8 CPU/memory board. If the battery backup unit is not installed, this light has the same meaning as the +5V light above.
-5 BU	The -5V battery backup supply is in tolerance. This output voltage is available only to the POINT 4 MARK 8 CPU/memory board. If the battery backup unit is not installed, this light has the same meaning as the -5V light above.
+12 BU	The +12V battery backup supply is in tolerance. This output voltage is only available to the POINT 4 MARK 8 CPU/memory board. The +12V supply is operational with or without the battery backup unit.

3.3 PROCESSOR MINI-PANEL

The POINT 4 MARK 8 processor chassis houses essential controls and indicators for basic processor control functions. The processor Mini-panel is located on the left-hand side of the POINT 4 MARK 8 chassis (see Figure 1-1).

3.3.1 MINI-PANEL OPERATING FUNCTIONS

There are three types of operating functions on the Mini-panel. These functions are: power controls and indicators, processor monitoring indicators, and program executions controls. Figure 3-2 illustrates the processor Mini-panel controls and indicators.



082-13

Figure 3-2. POINT 4 MARK 8 Processor Mini-panel

3.3.1.1 Power Controls and Indicators

The lower half of the Mini-panel is devoted to power control and monitoring.

3.3.1.1.1 POWER CONTROLS

The power switch is controlled by a four-position key-operated rotary switch. This switch controls the four functions described in Table 3-2.

TABLE 3-2. PROCESSOR MINI-PANEL POWER CONTROL SWITCH SETTINGS

Switch Setting	Function
OFF	Disables the battery backup unit. This switch position is used for storage or extended power off conditions. The battery charger is on as long as the power supply is plugged in.
STDBY	Turns the processor off, leaving the battery backup unit operational (if installed). Voltages for +5V and +15V are removed from the chassis but +5 BU, -5 BU and +12 BU are maintained for CPU slot use only.
ON	Turns on power to the processor and places the Mini-panel in the Panel-On Mode. In this mode all controls and indicators on the Mini-panel are enabled. The power-fail auto-restart feature (available with the battery backup option) is disabled, and the processor must be started manually whenever AC power is turned ON.
AUTO	Maintains processor power on and places the Mini-panel in the Panel-Off Mode. In this mode all controls on the Mini-panel are disabled, and all indicators remain active. Disabling the controls prevents interference with the operation of the CPU. The power-fail auto-restart feature is enabled if the battery backup option has been installed. This feature causes the processor to resume operation automatically upon AC power restoration following power failure.

3.3.1.1.2 POWER INDICATOR INTERPRETATIONS WITHOUT BATTERY BACKUP

Light-emitting-diode (LED) indicators illuminate to indicate an active state for AC power and Battery Backup power. Table 3-3 gives interpretations for the POWER OK LED for systems without the Battery Backup option.

TABLE 3-3. POWER INDICATOR INTERPRETATIONS WITHOUT BATTERY BACKUP

Power Supply AC IN	Processor Chassis POWER OK	Interpretation
OFF	OFF	Power supply not connected to AC.
ON	OFF	Power supply operational but power not available to processor chassis. If keyswitch is in ON or AUTO position, this indicates that one of the power supply voltages is out of tolerance (see Power Supply Indicator panel) or that there is a problem in the cabling between the processor and the power supply.
ON	ON	All power supply voltages are in tolerance and available to the processor chassis.

3.3.1.1.3 POWER INDICATOR INTERPRETATIONS WITH BATTERY BACKUP

The BTRY OK LED is operational if the Battery Backup option has been installed in the power supply chassis. The meaning of the BTRY OK LED depends on the state of the AC IN LED. These interpretations are shown in Table 3-4.

Figure 3-2 illustrates the positions on the key-operated rotary switch, the POWER OK indicator, and the BTRY OK indicator.

TABLE 3-4. POWER INDICATOR INTERPRETATIONS WITH BATTERY BACKUP

Power Supply AC IN	Processor Chassis BTRY OK	Interpretation
OFF	OFF	AC power source is off and the Battery Backup Unit has been fully discharged.
OFF	ON	The Battery Backup Unit is supplying +5V BU, -5V BU and +12V BU to the CPU/memory board to maintain the contents of memory.
ON	OFF	The Battery Backup Unit is being recharged but is not yet fully charged. The LED indicator blinks as the batteries are being charged.
ON	ON	The Battery Backup Unit batteries are fully charged.

3.3.2 PROCESSOR OPERATION MONITORING INDICATORS

The Mini-panel has three LED indicators in addition to the power monitoring indicators. These indicators perform the functions shown in Table 3-5. (See Figure 3-2 for the locations of the operation monitoring LED indicators.)

TABLE 3-5. PROCESSOR OPERATION MONITORING INDICATORS

Indicator	Function
PAR ERR	Indicates that a parity error has occurred during a memory read operation. The LED indicates that the processor has come to a halt pending operator action. Pressing the CONTinue or APL switch causes the processor to resume operation and turns off the parity error light. This LED is operational only if the parity error option is installed.
CARRY	Indicates the current state of the processor carry flag. The LED illuminates when the carry flag is set to a 1.
RUN	Indicates that the processor is in normal operation (executing one instruction after another). When the processor is stopped, the light goes off.

3.3.3 PROGRAM EXECUTION CONTROLS

Three momentary contact switches are available to control program execution in the processor. These switches are enabled in the Panel-On Mode (key switch set to the ON position). They are disabled in the Panel-Off Mode (key switch set to the AUTO position). Table 3-6 describes the switches and their functions. (See Figure 3-2 for the locations of the program execution controls.)

TABLE 3-6. PROGRAM EXECUTION CONTROLS

Switch	Function
STOP	Stops processor operation before executing the next instruction. The processor finishes current instruction, fetches the next instruction and then stops. The Program Counter then points to the next instruction to be executed. **NOTE** If the processor is caught in an infinite indirect addressing loop which prevents completion of the instruction, the STOP control will not work. Press APL to stop processor and load MANIP for debugging (see Section 3.5 for use of MANIP).
CONT	Causes program execution to resume, starting at address contained in Program Counter.
APL	The Automatic Program Load (APL) switch performs these functions: 1. Loads contents of an octal debugger/manipulator PROM (MANIP) into the top 1000 (octal) words of memory. The debugger/manipulator is used for access to accumulators and memory. Allows examination and deposit of data for operation monitoring and control. Optionally allows loading of system software from disc or other DMA devices. See Section 3.5 for debugger/manipulator program commands. Note that if CPU Board switches are set to 2XX (octal), the CPU performs an automatic APL from device XX when the APL switch is pressed. 2. May also be used in combination with the Self-Test switch (located on front edge of CPU circuit board) to load contents of Self-Test PROM into memory. The Self-Test program performs a complete check of hardware functions and executes a worst-case memory test. It can be used either as a hardware verifier or as a continuous reliability test. See Section 2.5 for description of self-test diagnostics. If the CPU is running, press STOP before pressing APL. However, if the processor is performing a multi-level indirect addressing instruction, pressing APL causes the processor to stop without use of STOP switch.

3.4 (not applicable)

3.5 VIRTUAL CONTROL PANEL

The POINT 4 MARK 8 has the ability to perform many front panel operations plus many extra system monitoring functions from a master terminal. This feature is designed for use by programmers to debug system problems and to manipulate the contents of registers and memory. The feature is implemented in a stand-alone program called MANIP which is loaded into RAM from a PROM when the APL switch is pressed.

MANIP is a simple but powerful position-independent memory manipulator and debug package. MANIP occupies only 1000 (octal) words of memory.* All operations are executed by typing one letter followed by octal parameters as required (except ":" which is also preceded by an octal parameter) and ending with a RETURN.

Table 3-11 shows the functions provided by MANIP (the number in parentheses indicates the number of parameters required for that particular function). The MANIP functions are described in detail in the following subsections.

MANIP is initially loaded into locations 177000 through 177777.

Location 177000 is reserved for saving the initial value of the program counter (PC), that is, the value of PC where the CPU had halted before MANIP was started. MANIP may be moved at any time by use of its MOVE (M) instruction.

The carry light flashes while MANIP is waiting for an input character to be entered (except in I mode). This is a signal that MANIP is active and will respond to input.

If an error is made while entering control information, it may be corrected in two ways.

1. Press ESC (or any other control character except RETURN) to delete the type-in and enable a new type-in.

2. If the error was made in entering an octal value, type a number of zeros followed by the correct octal number, as MANIP uses only the last six octal digits typed in for the octal word.

*For those who are familiar with POINT 4 Data Corporation's IRIS Operating System, MANIP is comparable to DBUG. The main differences are that MANIP does not have (1) symbolic capability, (2) breakpoints or trace, (3) disc read or write, and (4) Ctrl H/Ctrl A (backspace) capability. MANIP occupies only 1000 (octal) words of memory, while DBUG occupies 3000 (octal) words of memory.

TABLE 3-11. MANIP COMMAND FUNCTIONS AND PARAMETERS

Code	Function	Parameters Required
A	Type initial PC, accumulators and carry flip-flop	(0)
C	Change accumulator or carry flip-flop	(2)
D	Dump (octal, word or byte)	(1 or 2)
E	Enter octal into sequential locations	(1 or 2)
F	Set up an address offset	(0, 1 or 2)
J	Jump with accumulators and carry restored	(1 or 2)
K	Store a constant in a block of memory	(3)
M	Move a block in memory	(3)
N	Search memory for not-equal (with mask)	(3 or 4)
O	Output ASCII string on master terminal	(1 or 2)
P	Program load from disc	(0 or 1)
S	Search memory for constant using a mask	(3 or 4)
X	Calculate a checksum for a block of memory	(2)
Y	Set up an output delay after each RETURN (for proper scrolling)	(1)
:	Examine or deposit into a specified location	(2)
CTRL	Control characters are used to access CTU	See 3.6.1

3.5.1 ADDRESSING MODES

For many commands, MANIP allows either word or byte addressing, using either real memory addresses or offset (virtual) memory addresses based on an offset that has been previously entered (via F command). MANIP also is designed to allow addressing up to 64K words of memory. This is accomplished by having two word addressing modes (real and virtual), and three byte addressing modes (one virtual plus two real modes: lower 32K and upper 32K).

These modes are invoked by the optional second parameter "a" shown for commands D, E, J and O.

NOTE

> The J command does not permit byte addresses.
>
> When no "a" parameter is given, the addressing mode is "word address, including offset, if any." (If there is no "a" parameter, the preceding comma is optional.)

The "a" parameter definitions are as follows:

a Parameter	Definition
omitted	word address, including offset, if any
0	word address, absolute
1	byte address, using offset, if any
2	byte address, lower 32K
3	byte address, upper 32K

3.5.2 COMMAND DESCRIPTIONS

A MANIP command consists of a single letter which is the command identifier and parameters which specify addressing modes, memory addresses and data input. All parameters must be entered in octal form. The letters x, y, z, a, m, and n are used on the following pages to represent octal parameters. Press the RETURN key after entering any command.

Table 3-12 shows MANIP commands and descriptions.

TABLE 3-12. MANIP COMMAND DESCRIPTIONS

Command & Parameters	Definition
A	Type out initial value of program counter (PC) saved in first location of MANIP, contents of accumulators A0, A1, A2, A3, and carry flip-flop as they were at the time MANIP was entered.
Cx,y	Change accumulator or carry flip-flop: ● If x is 0, 1, 2, or 3, then y is stored as saved value for accumulator x (A0, A1, A2, A3, respectively). ● If x is 4, then saved value of the carry flip-flop is set to 0 if y=0; saved value is set to 1 if y=1. ● If x is greater than 4 and an address offset has been established (see F command), x is interpreted as a real address using the offset previously established, and typed out on the master terminal. The y parameter is not used in this case. ● Parameter Description x - 1 octal digit 0-7 y - 1 word octal

TABLE 3-12. MANIP COMMAND DESCRIPTIONS (Cont)

Command & Parameters	Definition
Dx,a	Dump memory in octal, beginning at location x, using addressing mode a. Eight words (or bytes if a byte address mode is used) are typed per line, with the address of the first word (byte) at the beginning of each line. ● Parameter Description x - an octal number representing a 16-bit memory address a - one digit (0-3 or omitted) representing an addressing mode
Ex,a	Enable entry at address x, using address mode a. The address (changed to a word address if it was a byte address) is printed, followed by a colon; an octal value may then be entered into the memory location, followed by RETURN. The next address (x+1) will then be printed and opened for entry. Entry may be thus continued into sequential address locations until ESC is pressed (after RETURN) to terminate entry. ● If no entry is typed in before RETURN, the present contents of the opened location is typed out in octal form, allowing examination of value before entering a new one. If another RETURN is entered without an entry, the current value is preserved, and the next address is printed and opened for entry. ● If a space character is typed instead of RETURN, the contents of the previous address is typed and opened. This feature is convenient for confirming an entry just typed in. ● Parameter Description x - octal number representing 16-bit memory address a - one digit (0-3 or omitted) representing a memory addressing mode

TABLE 3-12. MANIP COMMAND DESCRIPTIONS (Cont)

Command & Parameters	Definition
Fx,y	Establish an address offset (i.e., a fixed difference between real memory address on the one hand and addresses as entered and listed in MANIP on the other). The difference x-y is added to addresses entered, and subtracted from memory addresses before listing. If y is not entered, it is assumed to be zero. Whenever a nonzero offset is established, an F is typed at the beginning of each line. To revert to real memory addressing, type F0. ● Parameter Description x - from 1 to 6 digits (octal) representing a real memory address y - from 1 to 6 digits (octal) representing listing address which is equivalent to address specified in x
F	Save current offset value, and reinstate previous offset in effect before current one was established. Types offset being reinstated. Makes it convenient to toggle back and forth between two different offsets (or one offset and real memory addressing).
Jx,a	Jump to location x (using addressing mode a) with accumulators and carry stored. ● Parameter Description x - an octal number representing 16-bit memory address a - one digit (0-3 or omitted) representing a memory addressing mode
Kx,y,z	Store the octal constant z in locations x through y, inclusive. ● Parameter Description x - octal number representing 16-bit beginning memory address y - octal number representing 16-bit ending memory address z - octal number representing constant

TABLE 3-12. MANIP COMMAND DESCRIPTIONS (Cont)

Command & Parameters	Definition
Mx,y,z	Move block in core. Locations x through y, inclusive, are moved to area starting at location z. ● Source and destination areas may overlap in either direction without bad effects. ● May be used to move MANIP itself as long as destination area does not overlap source area. ● Parameter Description x - octal number representing 16-bit beginning memory address y - octal number representing 16-bit ending memory address z - octal number representing 16-bit beginning memory address of new location
Nx,y,z,m	Search for not-equal (see "S" command). Search locations x through y, inclusive, for values not equal to constant z. Each word is first ANDed with mask m before comparison with z. ● If m is not entered it is assumed to be 177777. ● Use of the mask is best explained by an example: The command Nx,y,0,170000 will search locations x through y for any value whose four MSBs are set greater than 7777. When such a value is found, its address and contents are typed in octal form on the CRT. ● Parameter Description x - octal number representing 16-bit beginning memory address y - octal number representing 16-bit ending memory address z - octal number representing constant m - octal number representing mask; or a blank which defaults to the value 0.

TABLE 3-12. MANIP COMMAND DESCRIPTIONS (Cont)

Command & Parameters	Definition
Ox,a	Output ASCII. Contents of memory starting at location x (using address mode a) are typed out as text. Output is terminated if a zero byte is encountered. ● Parameter Description x - octal number representing 16-bit memory address a - one digit (0-3 or omitted) representing a memory addressing mode
Px	Program load from disc or other DMA devices. Performs standard bootstrap APL function (i.e., gives an NIOS instruction with device code x and then idles at location 377 waiting for the disc to overwrite that location). If x is omitted, reads mini-switches at front edge of CPU board and uses their contents as the device code. **NOTE** If switch representing the 200 bit is set in addition to the device-code switches, MANIP cannot be accessed. Pressing APL causes MANIP to try to boot from the disc. ● Parameter Description x - 2-digit octal number (01 through 76) representing the disc device code from which the program is to be loaded

TABLE 3-12. MANIP COMMAND DESCRIPTIONS (Cont)

Command & Parameters	Definition
Sx,y,z,m	Search locations x through y, inclusive, for constant z. Each word is first ANDed with mask m before comparison with z. • If m is omitted, it is assumed to be 177777. • Use of mask is best explained by an example: The command Sx,y,60025,160077 will search locations x through y for any I/O instruction for device 25. When a comparison is found, its address and contents are typed in octal form on CRT. • Parameter Description x - octal number representing 16-bit beginning memory address y - octal number representing 16-bit ending memory address z - 1 to 6 digits representing octal constant m - 1 to 6 digits representing octal mask; or a blank which defaults the value to 177777
Xx,y	Calculate and type checksum over memory locations x through y. Utilizes a revolving checksum (using a SUBL 0,1 instruction, with A0 = each word from x through y, and A1 = accumulating checksum; initially 0). This insures that if two words in memory are swapped, the swap will be detected by the checksum. Useful for determining if any word in memory has changed. • Parameter Description x - an octal number representing 16-bit beginning memory address y - an octal number representing a 16-bit ending memory address

TABLE 3-12. MANIP COMMAND DESCRIPTIONS (Cont)

Command & Parameters	Definition
Yx	Set up a RETURN delay, required on some CRTs for proper scrolling. After each subsequent carriage return/line feed, MANIP counts up an accumulator from x to 0 before proceeding. For maximum delay set x=0, for no delay set x=177777. ● Parameter Description x - 1 to 6 octal digits representing RETURN delay value
x:y	Octal value y is stored at location x, and next cell is opened. See E command for more information. ● Parameter Description x - octal number representing 16-bit memory address y - 1 to 6 digits representing an octal value
CTRL ( )	Any Control Character (except <CTRL-W>) is recognized as a Cassette Tape Unit (CTU) Command. MANIP will pass the command to the CTU, with the exception of <CTRL-R>, which it uses to read the CPU. See Section 3.6.1 for CTU commands and command functions.

3.6 PROCESSOR/CASSETTE TAPE UNIT INTERFACE

This section describes commands used to transfer stand-alone programs such as diagnostics between the Cassette Tape Unit (CTU) and the POINT 4 MARK 8. The CTU is used in conjunction with the POINT 4 MIGHTY MUX 310 board. The master port (device code 10/11) is port 0, and the CTU is port 1 of the 310 MUX. There are sixteen basic CTU commands which can be enabled from the master terminal. DBUG (POINT 4's stand-alone debug program) can perform all sixteen commands; MANIP (the Virtual Control Panel program built into POINT 4 MARK 8) can perform only a subset of the CTU commands. Section 3.6.2 describes CTU commands enabled in MANIP; Section 3.6.3 describes those enabled in DBUG.

3.6.1 CTU COMMANDS

A CTU command consists of a single character control code (CTRL and an ASCII character), a block number for the starting block, an additional block count and a <RETURN>. There are sixteen functions which can be specified by control codes from an ASCII keyboard.

3.6.1.1 Command Functions

The control code of a CTU command is a single nonprinting character entered while holding down the CTRL key on the keyboard. The CTU will echo two printable characters, a caret for the control key and the ASCII letter representing the command for ease of command verification. The CTU command functions are listed in Table 3-13.

TABLE 3-13. CTU COMMAND FUNCTIONS

Function	Control Code	CTU ASCII Echo
Read Blocks	<CTRL-R>	^R
Write Blocks	<CTRL-W>	^W
Seek Block	<CTRL-S>	^S
Enquiry	<CTRL-E>	^E
Verify Block	<CTRL-V>	^V
Write Buffer	<CTRL-B>	^B
Access Buffer	<CTRL-A>	^A
Fill Buffer	<CTRL-F>	^F
Put in Buffer	<CTRL-P>	^P
List Directory	<CTRL-D>	^D
Open/Create File	<CTRL-O>	^O
Kill File	<CTRL-K>	^K
Rewind Drive	<CTRL-Z>	^Z
Select Track	<CTRL-T>	^T
Initialize Track	<CTRL-I>	^I
Cancel Command	<CTRL-X>	^X

3.6.1.2 Command Format

All CTU commands are structured as follows:

```
COMMAND [BLOCK NO.], [ADD'L BLOCK COUNT] <RETURN>
```

NOTE

> A field enclosed in brackets is an optional field.

Field functions can be described as follows:

COMMAND
This is a one-character control code specifying the function to be performed. See Subsection 3.6.1.1 for a complete listing of control codes and their functions.

[BLOCK NO.]
This is an optional DECIMAL block address specification. Zero (0) is the first block address. The maximum block address depends upon tape length (typically 999).

NOTE

> CTU blocks contain only 128 words; IRIS Operating System blocks contain 256 words.

[ADDITIONAL BLOCK COUNT]
This is an optional DECIMAL field which specifies the number of blocks to be operated upon in addition to the block specified in the [BLOCK NO.] field. A specification of zero (0) for this field instructs the CTU to operate only on the block specified in the [BLOCK NO.] field. The maximum count is 255.

<RETURN>
An ASCII carriage return character is the execute instruction for the CTU. If the CTU receives a <CTRL-X> before a <RETURN>, the CTU cancels (does not execute) the preceding command string specified. A new command can follow the <RETURN>.

3.6.1.3 CTU Error Conditions

The CTU reports error conditions by presenting an error code. An error condition is given in the following format:

BELL, Error Code, BELL, <RETURN>, Line Feed

The error codes and the errors they represent are shown in Table 3-14.

TABLE 3-14. CTU ERROR CODES

Error	Description
P	A write or erase operation was attempted on a write-protected tape.
M	Tape motion failure. This error occurs either as a result of a jam or mechanical malfunction, or as a result of incorrect tape positioning due to operator handling. In case of incorrect tape positioning, the CTU does not know the tape location and thus runs into the stops.
R	Read error. The operator should retry the command. An excessive number of read errors usually indicates noise interference, faulty system ground, a defective tape or a CTU hardware malfunction.
U	Unknown name. An attempt was made to delete a filename not found in the directory.
?	Syntax error in command string.
F	Track Directory is full (126) names. An old filename must be killed before a new filename may be entered.

3.6.2 CTU COMMANDS IN MANIP

MANIP allows reading from CTU into RAM (CPU main memory), but does not allow writing onto tape. To write to the CTU, obtain a cassette which contains DBUG, read it into memory by means of the MANIP <CTRL-R> command, then Jump into DBUG and use its CTU write capabilities (see Section 3.6.3).

All MANIP commands consist of a control character (CTRL and an ASCII character), followed optionally by one or more parameters, and terminated by a <RETURN>. The only exception is <CTRL-X> which cancels any partially entered command immediately. Data is stored on tape in blocks of 256 bytes (128 words) each. Table 3-15 lists the CTU commands used in MANIP. All numeric parameters (x,y) are in DECIMAL, origin 0.

The Read command transfers data into memory starting at address 0. To start the transfer at some other address, precede the CTU command with:

```
    Memory address (octal) : <RETURN>
```

MANIP then displays the content of the chosen location, followed by a colon. This allows examination of the word before starting the tape transfer. To proceed, type <CTRL-R>, followed by its parameters (if any) and a <RETURN>.

TABLE 3-15. CTU COMMANDS IN MANIP

Control Character/ Parameters	Description
<CTRL-D>	List directory (index) from tape, if tape is so formatted.
<CTRL-E>	Enquire (error status).
<CTRL-O>file	Open the named file, if it is in the directory.
<CTRL-O>file,x,y	Create a directory entry for the named file starting at block x and containing y+1 blocks of 128 words each.
<CTRL-R>	Read the open file from tape into memory.
<CTRL-R>x,y	Read from tape into memory; read y+1 blocks starting at block x.
<CTRL-S>x	Seek to block x on tape. <CTRL-S>999 will wind the tape all the way forward.
<CTRL-T>n	Select track n (0 or 1).
<CTRL-X>	Cancel partially entered command.
<CTRL-Z>	Rewind tape to starting position.
<CTRL-K>file	Kill the named file, i.e., erase its name from the directory.

NOTE

ESC exits CTU mode and reverts to normal MANIP commands, but does not cancel any partial command that may already have been transmitted to the CTU. Use <CTRL-X> to cancel a partial command.

3.6.3 CTU COMMANDS IN DBUG

DBUG is a position-independent debug package of the IRIS Operating System. When using the Virtual Control Panel on the POINT 4 MARK 8 it is necessary to use DBUG commands to write to the CTU. The write procedure from MANIP requires a cassette containing DBUG which must be read into memory using MANIP. A jump into DBUG allows use of DBUG CTU commands to write to the CTU. CTU commands in DBUG may also be used in other CTU transfer procedures.

All CTU commands consist of a control character, followed optionally by one or more parameters, and terminated by a <RETURN>. The only exception is <CTRL-X> which cancels any partially entered command immediately. Data is stored on tape in blocks of 256 bytes (1216 words) each. Table 3-16 lists the CTU commands used in DBUG. All numeric parameters (x,y) are in DECIMAL, origin 0.

TABLE 3-16. CTU COMMANDS IN DBUG

Control Character/ Parameters	Description
<CTRL-A>x,y	Access CTU buffer, i.e., transfer buffer into memory. Transfers y bytes, starting at byte x. Default = 256 bytes starting at byte 0.
<CTRL-B>x	Write CTU buffer onto tape, at block x.
<CTRL-D>	List directory (index) from tape, if tape is so formatted.
<CTRL-E>	Enquire (error status).
<CTRL-F>	Fill CTU buffer from memory (128 words).
<CTRL-I>x	Initialize (format) selected track to x+1 blocks of 128 words each. Maximum = I999 for 1000 blocks.
<CTRL-K>file	Kill the named file, i.e., erase its name from the directory.
<CTRL-O>file	Open the named file, if it is in the directory.

TABLE 3-16. CTU COMMANDS IN DBUG (Cont)

Control Character/ Parameters	Description
<CTRL-O>file,x,y	Create a directory entry for the named file (max. 5 char.), starting at block x and containing y+1 blocks of 128 words each.
<CTRL-P>x,y	Put into CTU buffer from memory, transferring y bytes beginning at byte x in the buffer. Default = 256 bytes starting at byte 0.
<CTRL-R>	Read the open file from tape into memory.
<CTRL-R>x,y	Read from tape into memory; read y+1 blocks starting at block x.
<CTRL-S>x	Seek to block x on tape.
<CTRL-T>n	Select track n (0 or 1).
<CTRL-V>	Verify; i.e., read from tape into CTU buffer, checking checksum.
<CTRL-W>	Write from memory onto tape into the open file, if any.
<CTRL-W>x,y	Write from memory onto tape, writing y+1 blocks starting at block x.
<CTRL-X>	Cancel partially entered command.
<CTRL-Z>	Rewind tape to starting position.

NOTE

ESC exits CTU mode and reverts to normal DBUG commands, but does not cancel any partial command that may already have been transmitted to the CTU. Use <CTRL-X> to cancel a partial command.

All commands that transfer data into or out of main memory default to an initial address of 0. To start the transfer at some other address, precede the CTU command with:

```
    Memory address (octal) :  <RETURN>
```

DBUG then displays the content of the chosen location, followed by a colon. This allows examination of the word before starting the tape transfer. To proceed, type the CTU control character (e.g., <CTRL-R> or <CTRL-W>), followed by its parameters and a <RETURN>.

Table 3-17 is a quick-reference guide to the commands used for data transfer from a source to a destination.

TABLE 3-17. SUMMARY AND OVERVIEW OF DATA TRANSFER COMMANDS

Source	Destination	Command
Tape	Memory	<CTRL-R>
Memory	Tape	<CTRL-W>
Tape	Buffer	<CTRL-V>
Buffer	Tape	<CTRL-B>
Buffer	Memory	<CTRL-A>
Memory	Buffer	<CTRL-F> complete buffer <CTRL-P> selected byte(s) only

Section 4
INPUT/OUTPUT INTERFACES

4.1 INTRODUCTION

This section provides information regarding the basic operating principles and programming methods for the input/output devices which are Compatible with the POINT 4 MARK 8 Computer. Two types of I/O devices are available:

- Those transferring data via I/O programmed instructions only
- Those using the data channel for input/output transfers

The following subsections outline the interrupt handling and priority scheme, conventions for handling the master Teletype or CRT, programmed transfer handling and data channel transfer handling. Also provided are I/O bus signal descriptions and I/O transfer timing diagrams.

4.2 PROGRAM INTERRUPT AND PRIORITY SCHEME

Many input/output devices require service within a short time after they request it, but they need service infrequently relative to the processor speed and require only a small amount of time for servicing. Failure to service within the specified time (which varies among devices) causes operation of the device below its maximum speed and can result in loss of information.

The use of interrupts in the current program sequence facilitates concurrent operation of the main program and a number of peripheral devices. The program interrupt scheme allows an I/O device to gain control of the processor. When an interrupt occurs, the processor suspends normal program execution and starts a device service routine. When the routine is completed, processing of the interrupted program may resume.

4.2.1 INTERRUPT SEQUENCE

When a device needs service, it sets its Interrupt Request flag. The processor begins servicing interrupts if all four of the following conditions exist:

- The processor has just completed an instruction fetch or a data channel transfer
- At least one device has a pending Interrupt Request
- Interrupts are enabled (i.e., ION is set)
- No device is waiting for a data channel transfer

The processor responds to the interrupt request by storing the value of the program counter into memory location 0 and jumping to the instruction addressed by memory location 1. Location 1 must contain the address of the interrupt handling routine. Interrupts are disabled at the start of the interrupt service cycle and must be re-enabled by the software at the end of the interrupt service.

4.2.2 DEVICE PRIORITY

The processor features a special Interrupt Acknowledge instruction that eliminates the need for lengthy device polling. This instruction inputs the device code of the interrupting device into an accumulator register permitting the interrupt service routine to identify the device requesting service. The computer uses a three-way priority system to determine which, if any, device may interrupt the processor at a given moment. First, the processor contains a programmable Interrupt ON (ION) flag. The processor recognizes interrupt requests only when this flag is set. Second, the processor can selectively disable the interrupt capability of each device with the device Interrupt Mask flags (see Section 5.6.2, MSKO instruction). Third, if two or more devices request interrupts simultaneously, the priority resolution is made by the Jumper-Saver logic on the POINT 4 chassis backplane. A device whose controller board is physically closer to the processor is given priority over a device that is further away. See Appendix E for an example of interrupt programming.

4.3 PROGRAMMED TRANSFERS

For programmed input/output the program directly controls the data transfer between the CPU and the I/O device. As discussed in Section 5.6, on input/output instructions, each data word is transferred between an accumulator specified in the instruction and an I/O device buffer (A, B, or C) specified in the instruction, as shown in Figure 4-1.

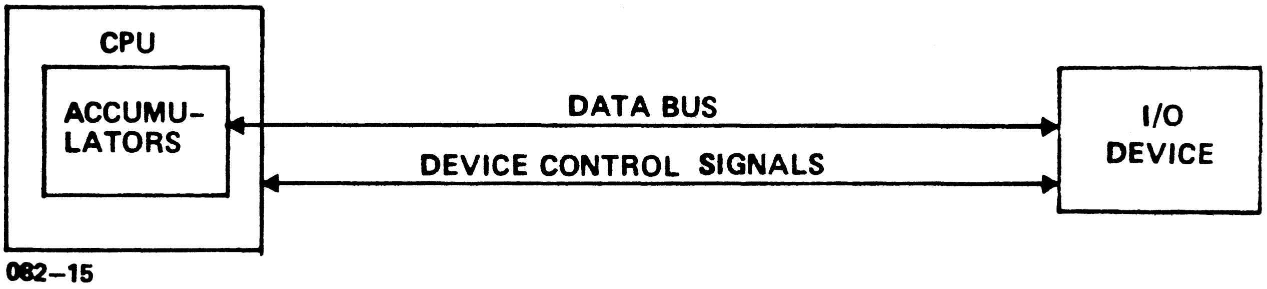


Figure 4-1. Data Word Transfer Between Specified Accumulator and I/O Device Buffer

4.3.1 MASTER TERMINAL INTERFACE

The standard Teletype or CRT I/O controller has separate interface logic for input and output. Input and output device codes are also separate. The input logic interfaces to the keyboard and, in some Teletypes, a paper tape reader. The output logic interfaces to the printer or video display and, in some Teletypes, the paper tape punch. When the CRT or Teletype is connected to the computer, a character entered on the keyboard for input to the computer must be "echoed" back to the output interface logic on the terminal in order to appear on the screen or paper.

All alphanumeric and control characters are represented by standard ASCII codes (see Appendix C) consisting of eight bits, the most significant of which is usually an even parity bit.

4.3.2 PROGRAMMING CONVENTIONS

Programming conventions for handling CRT and Teletype terminals are described in the following subsections.

4.3.2.1 Instruction Formats

Terminal output and input use separate device codes as specified in bits 10-15 of the instruction. The S or C pulse may be sent to clear Busy and/or Done flags and control the starting of the transfer between the interface and the device. The data transfer output instructions transmit bits 8-15 from the specified accumulator to the output interface register. The input instruction loads the input interface register into bits 8-15 and resets bits 0-7 of the specified accumulator.

4.3.2.2 Terminal Output

Transmission to the terminal takes place when an S pulse is sent, which sets the output Busy flag. When the character has been printed, the interface clears the Busy flag, sets the Done flag and requests an interrupt, if interrupts are enabled. Terminal output uses the device code 11, the mnemonic TTO, and uses bit 15 to control the Interrupt Mask flag. To transfer a character from an accumulator to the terminal output buffer, use the instruction

```
    DOA ac,TTO
```

where ac may be any of the four accumulators.

If the NIOS instruction is used to send the S pulse, the instructions must appear in the following sequence

```
    DOA ac,TTO
    NIOS TTO
```

Normally this operation is done in one instruction

```
    DOAS ac,TTO
```

4.3.2.3 Terminal Input Via CRT Keyboard

Terminal input uses octal device code 10, the mnemonic TTI, and uses bit 14 to control the Interrupt Mask flag. When a key is pressed on the keyboard, the character is placed in the input buffer, and Done is set. If interrupts are enabled for that device, an interrupt is requested. The program then reads the character with the instruction

```
    DIA ac,TTI
```

The program then uses an S pulse to clear Done. The S pulse may either be part of the DIA instruction

```
    DIAS ac,TTI
```

or be generated with an NIO instruction

```
    DIA ac,TTI
    NIOS TTI
```

4.3.2.4 Terminal Input Via Paper Tape Reader

When paper tape is used for input, the paper tape reader must be started by the program using the instruction

```
    NIOS TTI
```

This instruction sets Busy and clears Done in the TTY controller. When the character has been read from paper tape into the controller, the device controller clears Busy and sets Done, producing an interrupt if interrupts are enabled. The program then reads the character with the instruction

```
    DIA ac,TTI
```

Sequential characters can be read by using the instruction

```
    DIAS ac,TTI
```

This results in the reading of one character and the restarting of the paper tape reader for reading of the next character.

4.4 DATA CHANNEL TRANSFERS

Mass storage devices such as tape drives, discs and mass storage units can transfer blocks of data at high speeds directly into memory, without requiring programmed I/O instructions for each word transferred, by using the Direct Memory Access (DMA) data channel. Data channel device interface logic contains both conventional device registers and flags, and special data channel logic.

The program initiates a data channel transfer by supplying certain parameters to the device registers and starting the device. The device automatically transfers one or more data words to or from memory. When finished with the DMA transfer, the device generates an interrupt if so enabled. At the start of each instruction cycle, the processor checks to see if a device is requesting data channel service. If a device is requesting data channel service, the data channel transfer is performed before going on with the instruction. Several data channel devices can be active at the same time, with devices closest to the processor having channel priority over devices further away.

The POINT 4 MARK 8 has two jumper-selectable data channel speed options:

```
Standard Data Channel
    (Jumper CPU board Pin A93 to ground)
    Input  - 1100 nanoseconds
    Output - 1433 nanoseconds

High-Speed Data Channel
    (Do not jumper CPU board Pin A93 to ground)
    Input  -  800 nanoseconds
    Output -  933 nanoseconds
```

4.4.1 SELECTION OF DATA CHANNEL SPEED

CPU logic tests input to Pin A93 (top slot) to determine which speed has been enabled (see Figure 4-2, for backplane pin assignments). If Pin A93 is jumpered to ground, the data channel will operate at Standard Data Channel speeds. If Pin A93 is left open, the data channel will operate at High-speed Data Channel speeds.

4.4.2 CRITERION FOR DATA CHANNEL SPEED SELECTION

The High-Speed Data Channel on the POINT 4 MARK 8 does not have stringent requirements for controller timing. The controller is given about 200 nanoseconds from the start of DCHA to put its address on the I/O bus. The POINT 4 MARK 8 also allows 200 nanoseconds from the start of DCHI before it requires the input data on the I/O bus. (See Section 4.6.3 on Data Channel transfer approximate timing.)

The result of this relatively long access time to the I/O bus is that many DMA controllers which cannot operate on a high-speed data channel with other computers can operate on the POINT 4 MARK 8 High-speed Data Channel. Therefore the High-speed Data Channel should be used unless the system includes a controller with specifications which indicate otherwise. If operation is inconsistent at high speeds, the user can switch to the Standard Data Channel by jumpering Pin A93 to ground. If operation is then consistent at the Standard Data Channel speed, this speed should be maintained.

4.4.3 DATA CHANNEL ACCESS PRIORITIES

The amount of time a device must wait for data channel access depends on when its request is made within an instruction and how many devices of higher priority are also requesting access. Once the processor reaches a point at which it can pause to handle transfers, a given device must wait until all devices closer to the processor on the bus have been serviced. Under normal conditions, a device can preempt all processor time if it requests access at the maximum rate. An exception is made if Power-Fail has been sensed, in which case the data channel is allowed only every other cycle. The alternate cycle is used for Power-Fail Interrupt processing. At less than the maximum rate the closest device never waits longer than the time required for the processor to finish the instruction that is being performed when the request is synchronized. However, indirect addressing can extend this beyond the normal instruction execution time.

4.5 INPUT/OUTPUT BUS INTERFACE SIGNALS

Input/Output Bus signals connect the processor logic to peripheral device logic. The logic for programmed I/O transfers and data channel transfers forms the interface between the processor Main Data Bus and the peripheral device controller logic. Logic to implement both I/O transfer and I/O skip instructions is present in all device controllers. Data channel transfer logic is present only in those controllers that control devices using the data channel. Device-end control logic for these functions may vary widely, depending on the requirements of the particular device. This subsection describes the POINT 4 MARK 8 I/O bus and control signals.

4.5.1 INPUT/OUTPUT INTERFACE SIGNALS

Signals on the Input/Output Bus can be grouped into the signal classifications shown in Table 4-1.

Figure 4-2 is a diagram of I/O signals across the I/O Bus. Table 4-2 lists these signals by classification group and signal name, indicating direction and describing each signal function.

TABLE 4-1. INPUT/OUTPUT BUS SIGNAL CLASSIFICATIONS

Signal Classification	Number of Lines	Definition
Bidirectional Data Bus	16	Used for transfer of all data and address words between the CPU and a peripheral device, for both programmed I/O and data channel transfers.
Device Codes	6	Codes generated by CPU to specify the address of a peripheral device or a controller used for an input/output instruction.
Programmed Data Transfer Signals	6	These signals, generated by the CPU in response to input/output instructions for data transfers, are used to control data transfers for programmed input/output devices.
Device Control Signals	4	These signals are generated by the CPU in response to input/output instructions, and are used to initialize and control I/O devices. The signals affect only the device whose device code is in the instruction, except in the case of the IORST instruction which resets all devices.
Skip Testing Flags	2	Flags supplied to the CPU when skip-testing is required.
Interrupt Control Signals	5	Signals used to initialize and control the interrupt sequence.
Data Channel Transfer Signals	6	Signals used to control data channel transfers between memory and a peripheral device.

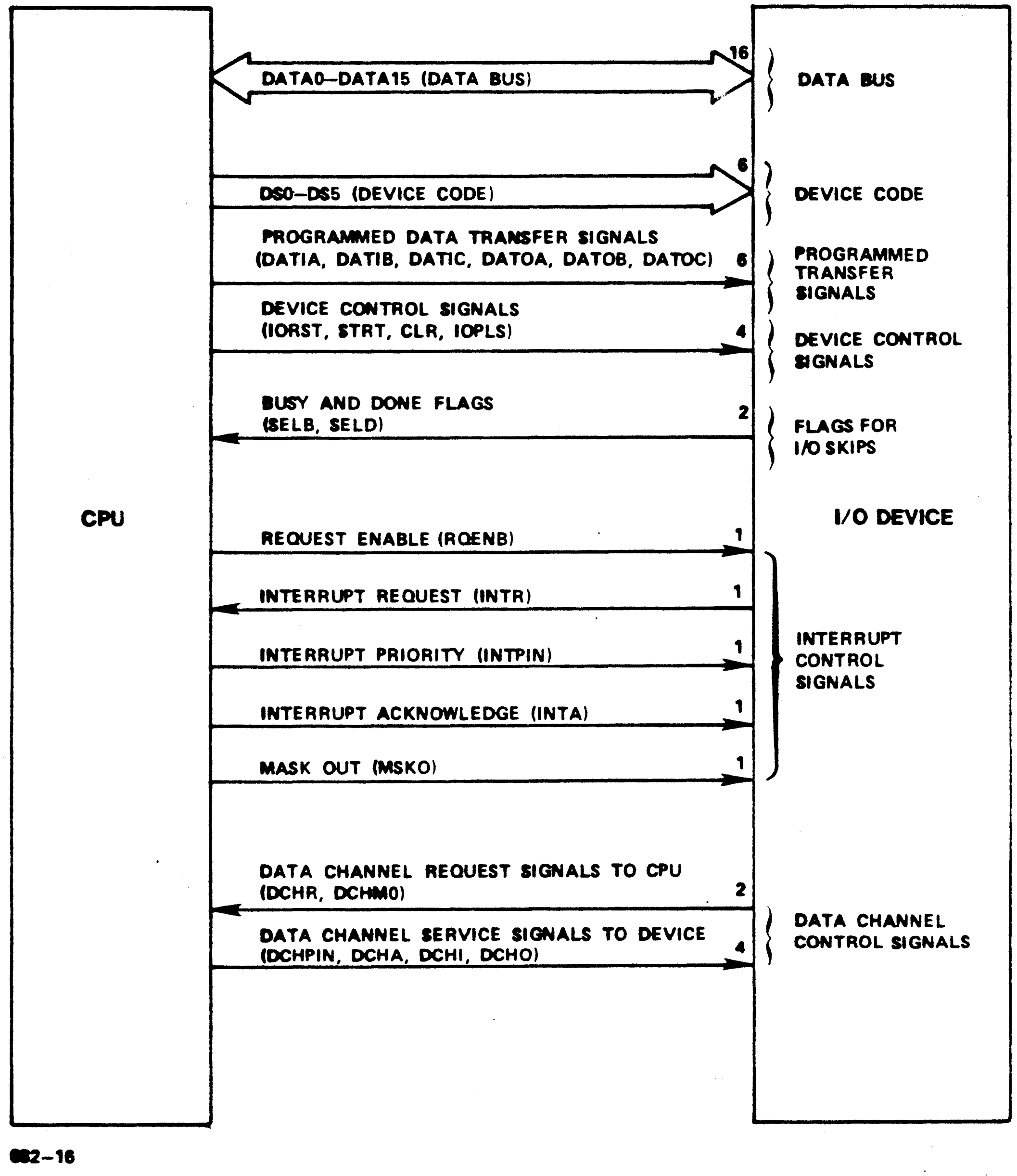

DATA0–DATA15 (DATA BUS)
16
DATA BUS
DS0–DS5 (DEVICE CODE)
6
DEVICE CODE
PROGRAMMED DATA TRANSFER SIGNALS
(DATIA, DATIB, DATIC, DATOA, DATOB, DATOC)
6
PROGRAMMED TRANSFER SIGNALS
DEVICE CONTROL SIGNALS
(IORST, STRT, CLR, IOPLS)
4
DEVICE CONTROL SIGNALS
BUSY AND DONE FLAGS
(SELB, SELD)
2
FLAGS FOR I/O SKIPS
CPU
I/O DEVICE
REQUEST ENABLE (RQENB)
1
INTERRUPT REQUEST (INTR)
1
INTERRUPT PRIORITY (INTPIN)
1
INTERRUPT ACKNOWLEDGE (INTA)
1
MASK OUT (MSKO)
1
INTERRUPT CONTROL SIGNALS
DATA CHANNEL REQUEST SIGNALS TO CPU
(DCHR, DCHMO)
2
DATA CHANNEL SERVICE SIGNALS TO DEVICE
(DCHPIN, DCHA, DCHI, DCHO)
4
DATA CHANNEL CONTROL SIGNALS
082–16

Figure 4-2. Input/Output Signals

TABLE 4-2. INPUT/OUTPUT SIGNALS

Signal Group	Signal Name*	Direction	Description
Data Bus	DATA0- to DATA15-	Bidirectional	All data and addresses are supplied to and from the device via these lines. DATA0- is the MSB.
Device Code	DS0- to DS5-	From CPU	The CPU places the device code (bits 10-15 of the instruction word) on these lines during the execution of an input/output instruction. DS0- is the MSB.
Programmed Data Transfer Signals	DATIA+	From CPU	Data In A. Generated by a DIA instruction. Causes the A buffer of the device whose device code is on the lines to be placed on the Data Bus for entry into the accumulator specified by the instruction.
	DATOA+	From CPU	Data Out A. Generated by a DOA instruction. Causes the accumulator specified in the DOA instruction to be placed on the Data Bus for entry into the A buffer of the device whose device code is on the lines.
	DATIB+	From CPU	Data In B. Generated by a DIB instruction. Functions like Data In A, except uses buffer B.
	DATOB+	From CPU	Data Out B. Generated by a DOB instruction. Functions like Data Out A, except uses buffer B.
	DATIC+	From CPU	Data In C. Generated by a DIC instruction. Functions like Data In A, except uses buffer C.
	DATOC+	From CPU	Data Out C. Generated by a DOC instruction. Functions like Data Out A, except uses buffer C.

*Signal names ending with "+" are active high; those ending with "-" are active low.

TABLE 4-2. INPUT/OUTPUT SIGNALS (Cont)

Signal Group	Signal Name*	Direction	Description
Device Control Signals	IORST+	From CPU	Input/Output Reset. Generated when APL is pressed on the Mini-panel, when RESET is pressed on the Operator Control Unit, when an IORST instruction is being executed, and during power turn-on.
	STRT+	From CPU	Start. Generated when the CTRL field of an input/output transfer instruction contains code 01. It usually clears the Done flag and Interrupt Request, and sets the Busy flag in the device whose device code is on the lines.
	CLR+	From CPU	Clear. Generated when the CTRL field of an input/output transfer instruction contains code 10. It usually clears the Busy and Done flags and the Interrupt Request in the device whose device code is on the lines.
	IOPLS+	From CPU	I/O Pulse. Generated when the CTRL field of an input/output transfer instruction contains code 11. The effect, if any, depends on the device.
Skip Testing Flags	SELB-	From Device	Selected Device Busy. Supplied to CPU by the device whose device code is on the lines when the Busy flag is set. Indicates that the device is busy.
	SELD-	From Device	Selected Device Done. Supplied to the CPU by the device whose device code is on the lines when the Done flag is set. Indicates that the device is done.

TABLE 4-2. INPUT/OUTPUT SIGNALS (Cont)

Signal Group	Signal Name*	Direction	Description
Interrupt Control Signal	RQENB-	From CPU	Request Enable. Generated during each memory read/write cycle to synchronize INTR- and DCHR-. In any device, changes in INTR- or DCHR- may only occur following the leading edge (high-to-low transition) of RQENB-.
	INTR-	From Device	Interrupt Request. This signal goes low (following the leading edge of RQENB-) if the device wants to request an interrupt.
	INTPIN-	From CPU	Interrupt Priority Input. On POINT 4 chassis, produced by Jumper-Saver logic on the backplane for the highest-priority device requesting an interrupt. If using universal CPU in non-POINT 4 chassis, produced by a jumper to ground on lowest I/O board.
	INTA+	From CPU	Interrupt Acknowledge. Generated by an INTA instruction. Causes the device whose INTPIN- line is low to place its device code in bits 10-15 of the Data Bus for entry into accumulator specified in the instruction.
	MSKO-	From CPU	Mask Out. Generated by a MSKO instruction. Commands all I/O devices to set their Interrupt Disable flags according to the state of the associated Mask Bit in the word on the Data Bus.

TABLE 4-2. INPUT/OUTPUT SIGNALS (Cont)

Signal Group	Signal Name*	Direction	Description
Data Channel Transfer Signals	DCHR-	From Device	Data Channel Request. This signal goes low (following the leading edge of RQENB-) if the device wants to request a data channel transfer.
	DCHPIN-	From CPU	Data Channel Priority. On POINT 4 chassis, produced by Jumper-Saver logic on the backplane for the highest-priority device requesting an interrupt. If using universal CPU in non-POINT 4 chassis, produced by a jumper to ground on lowest I/O board.
	DCHA-	From CPU	Data Channel Acknowledge. Generated by CPU in response to a Data Channel Request. Initiates a data channel cycle in the device whose DCHPIN- is low. The device places the memory address for data-channel access on the Data Bus.
	DCHMO-	From Device	Data Channel Mode. Generated by a device connected to the data channel while DCHA- is low. Indicates the type of data channel cycle being requested as follows: DCHMO — Type of Cycle 0 (high) — Data Out (from CPU) 1 (low) — Data In (to CPU)
	DCHI+	From CPU	Data Channel In. When the mode is Data In, DCHI+ is generated during the time the device is placing a data word on the Data Bus.
	DCHO+	From CPU	Data Channel Out. When the mode is Data Out, DCHO+ is generated during the time that the word accessed from memory is on the Data Bus.

4.5.2 BACKPLANE PIN SIGNAL CONNECTORS

All signal connections between the processor and each controller take place via two 100-pin backplane connectors. Figure 4-3 shows the connector pin layout for all I/O signals. The labelled pins refer to the I/O control signals, data transfer signals and the power lines used by peripheral controllers.

The POINT 4 MARK 8 has one special signal, STSEL-, which goes through backplane connector A pin 91. It is a remote Self-Test select signal that allows the CPU Self-Test program to be bootstrapped to memory when the signal is at an active low during the APL sequence.

SEE APPENDIX I

Figure 4-3. Backplane I/O Signals (1 of 2)

SEE APPENDIX I

Figure 4-3. Backplane I/O Signals (2 of 2)

4.6 INPUT/OUTPUT TIMING

Three classes of operations take place on the I/O Bus: operations associated with programmed I/O instructions, operations associated with interrupt handling, and operations associated with data channel transfers. Timing diagrams in this section represent each signal or group of signals by a horizontal line with a raised section representing the active state. Control signals generated at a specific time to control a particular function show the raised line for the time that the signal is active. For signals carrying binary information, the raised line indicates the amount of time during which that information remains on the bus. Raised lines may represent either high or low voltage levels, depending on whether the signal is active when low or high. (See Table 4-1 for active voltages on all signals.) Times on timing diagrams are given in nanoseconds.

4.6.1 PROGRAMMED I/O INSTRUCTION TIMING

Figure 4-4 shows the timing for Data In (DIx) and Data Out (DOx) instructions both with and without a Control Pulse (S, C or P) specified. In all cases, the processor first places the appropriate device code on the device select lines DS0-DS5. The selected device then responds to the signals which follow.

4.6.1.1 Data In

During Data In transfers, the processor generates a DATIA, DATIB, or DATIC signal. The device selected then places the contents of its appropriate buffer onto the data transfer lines. At the end of the DATIx active signal the processor strobes the value on the data lines into the appropriate processor accumulator. Following the transfer, the processor generates the pulse for a START (S), CLEAR (C), or PULSE (P), if called for by the instruction.

4.6.1.2 Data Out

During Data Out transfers, the processor places the contents of the accumulator selected by the instruction onto the data transfer lines. The processor then generates a DATOA, DATOB, or DATOC signal, which causes the device to strobe in the data to the buffer specified. When the data has been loaded into the buffer, the processor generates the pulse for START (S), CLEAR (C), or PULSE (P), if called for by the instruction.

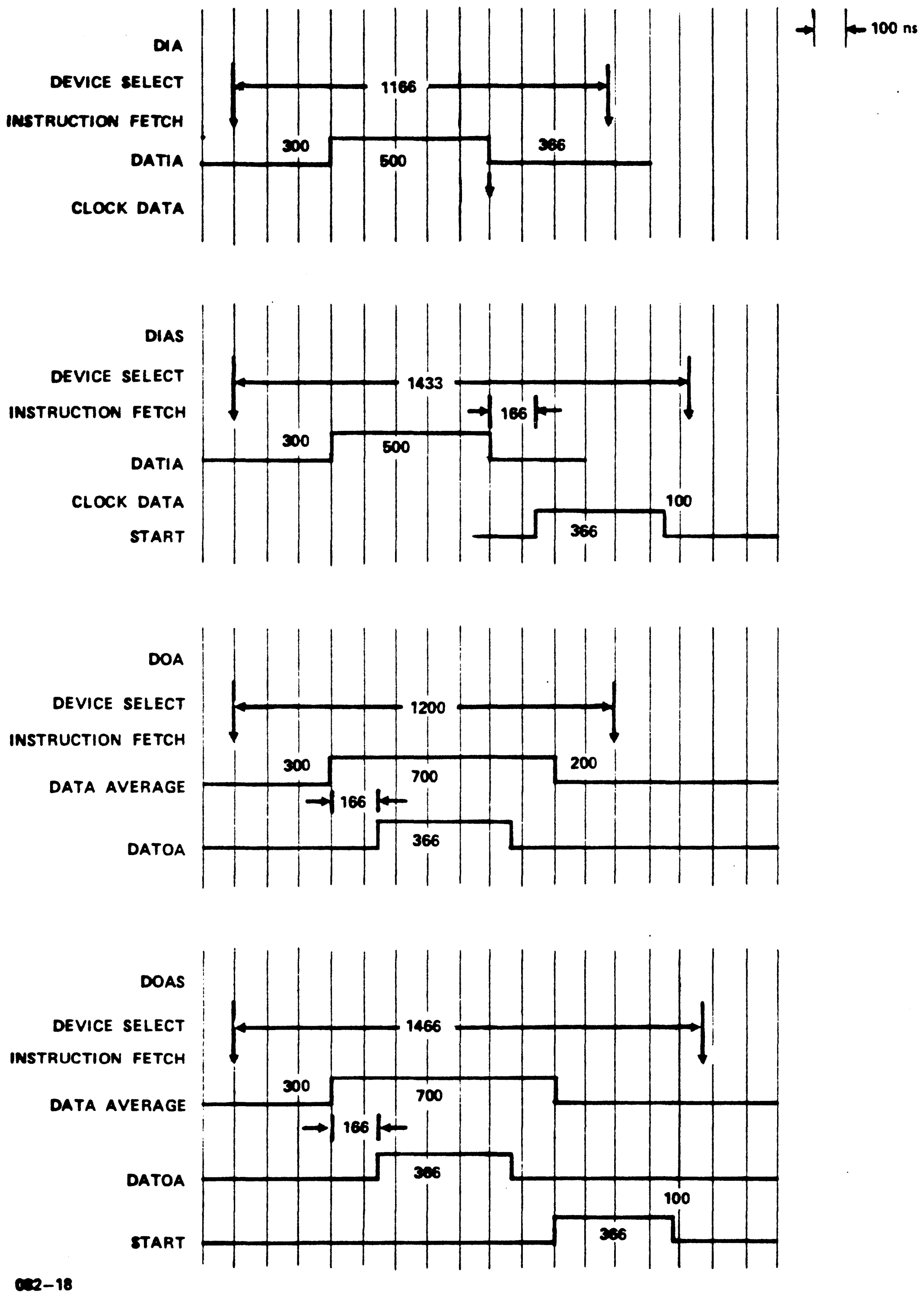
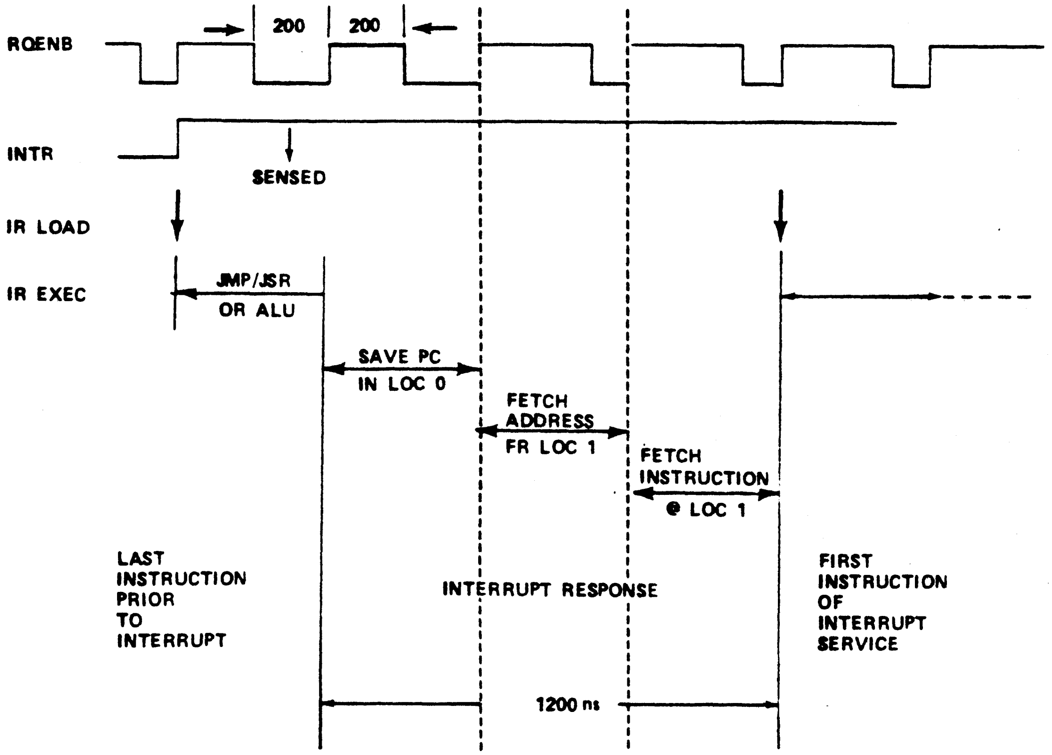


Figure 4-4. Programmed I/O Instruction Timing

4.6.2 PROGRAM INTERRUPT TIMING

At the end of every memory cycle the processor generates the signal RQENB and places it on the I/O Bus. All devices receive the RQENB signal and each responds according to its need for service. Any device requiring interrupt servicing pulls the signal INTR- low.

Figure 4-5 is a timing diagram of interrupt handling.



RQENB= REQUEST ENABLE: PROVIDES A CLOCK SIGNAL TO SYNCHRONIZE INTERRUPT REQUESTS (AS WELL AS DATA CHANNEL REQUESTS) FROM PERIPHERAL CONTROLLERS. ITS NOMINAL PERIOD IS 400ns (200ns OFF, 200ns ON). CONTROLLERS MAY CHANGE INTR (INTERRUPT REQUEST) OR DCHR (DATA CHANNEL REQUEST) ONLY AT THE LEADING EDGE OF RQENB.

INTR= INTERRUPT REQUEST: SENSED BY CPU AT THE TRAILING EDGE OF RQENB WHICH OCCURS 100ns BEFORE THE END OF EACH INSTRUCTION. IF INTR IS PRESENT, THE NEXT THREE MEMORY CYCLES (1200ns) ARE TAKEN TO SAVE PC IN LOCATION 0 AND TO JUMP TO THE INTERRUPT SERVICE ROUTINE WHOSE ADDRESS IS IN LOCATION 1.

082–19

Figure 4-5. Interrupt Timing

4.6.3 DATA CHANNEL TRANSFER TIMING

Data channel transfers are in either the input or output direction: Data Channel Input being a write into memory and Data Channel Output being a read from memory. In either case, the device first requests use of the I/O Bus. When the processor acknowledges the request, it stops program execution long enough to conduct the transfer between the device and memory.

Operations in data channel requests are similar to those of an interrupt request. At the end of every memory cycle the processor generates the signal RQENB and places it on the I/O Bus. All devices receive the RQENB signal and each responds according to its need for service. Any device requiring data channel service pulls the DCHR- line low. The Jumper Saver logic on the POINT 4 chassis backplane then determines which is the highest priority device requesting data channel service, and sends DCHP- (Data Channel Priority) to that device. Devices whose DCHP- line is inactive (high) ignore subsequent data channel control signals.

When the processor is ready to process the data channel request, it activates the signal DCHA (Data Channel Address). The device whose DCHP- line is active (low) places the address for the DMA transfer on the I/O bus during DCHA. At the same time the device also activates or negates DCHMO to specify whether an input or output transfer is to take place.

When DCHA terminates, the processor strobes the address into its memory address register. From this point on the operation depends on the direction of the data channel transfer.

Data Channel Input - When a data input transfer is required, the processor transmits DCHI immediately following the trailing edge of DCHA. The device then places the data word onto lines DATA0 through 15. Near the trailing edge of DCHI, the processor stores the data word into memory, and the device removes the data word from lines DATA0 through 15.

Data Channel Output - In an output transfer, the processor starts a Read memory cycle at the trailing edge of DCHA. When the data has been fetched from memory, the processor places the word on lines DATA0 through 15 and activates a DCHO signal. The device then fetches the data from the data lines.

When the transfer required is a single-word transfer, the device clears DCHR the next time it receives RQENB. If the transfer required is several words in consecutive data channel cycles, the DCHR flag should remain active until the leading edge of RQENB following the DCHA of the last transfer desired.

Figure 4-5 is a timing diagram of standard data channel operations and Figure 4-6 is a timing diagram of high-speed data channel operations.

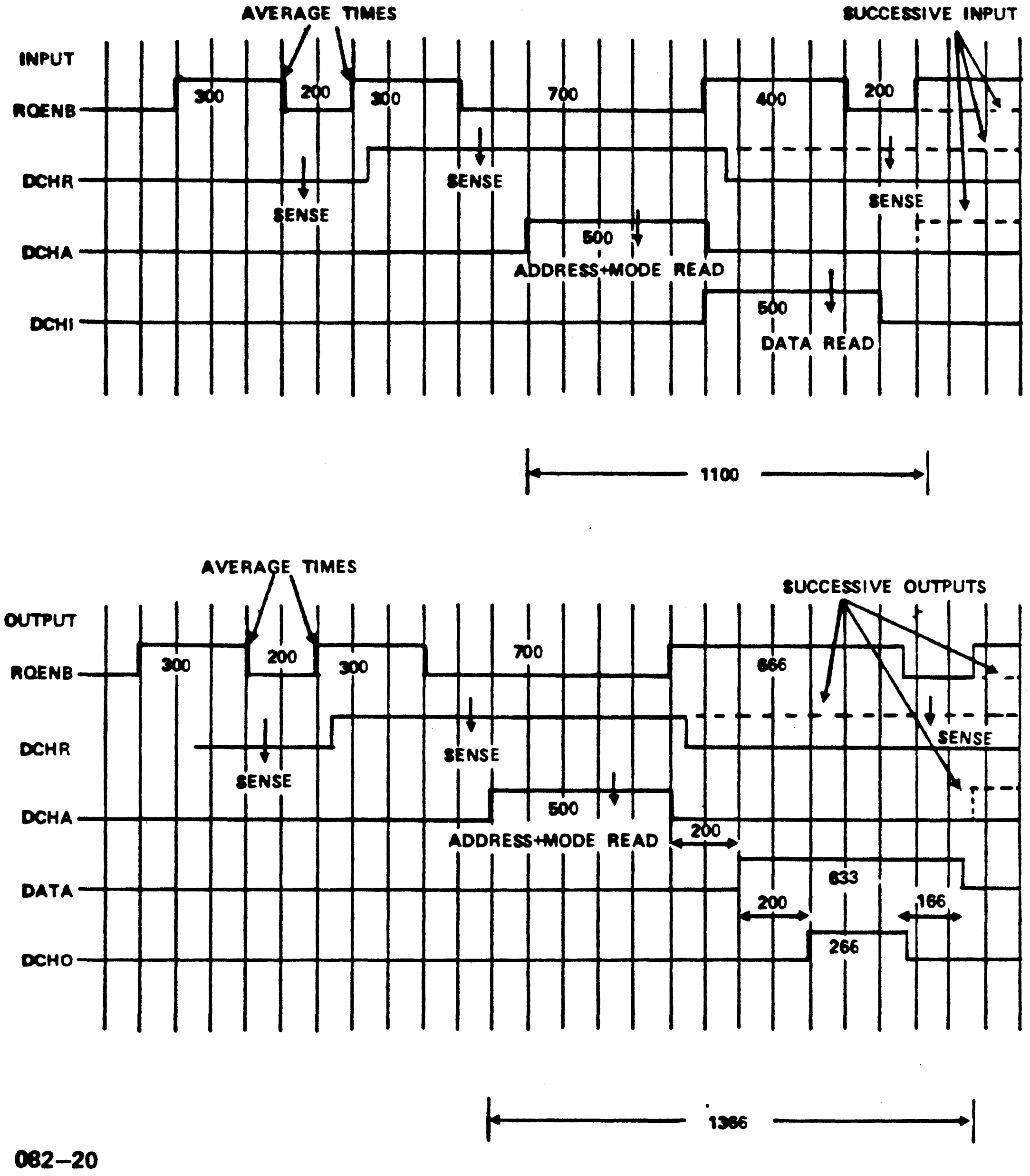


Figure 4-6. Standard Data Channel Timing

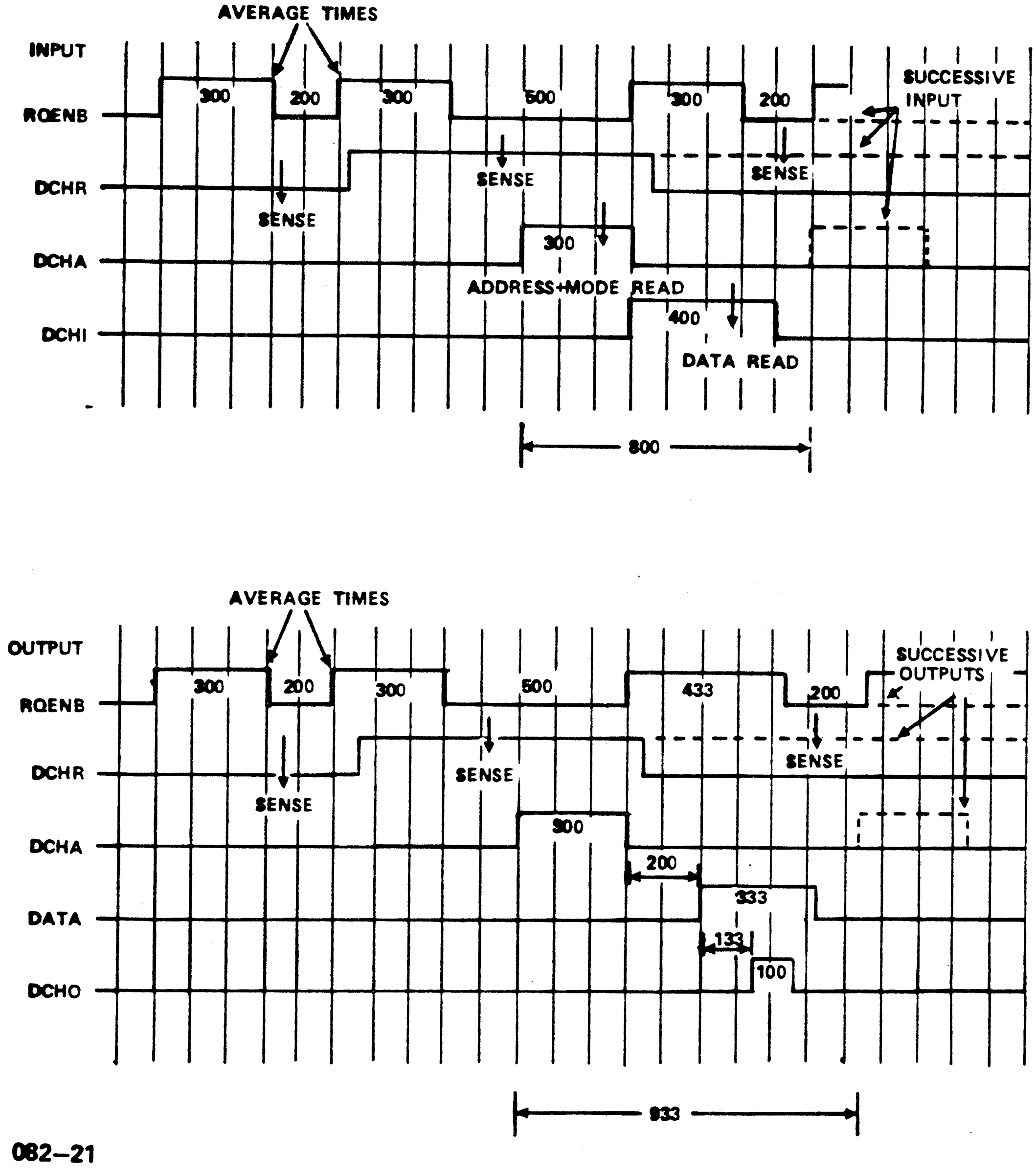


Figure 4-7. High-Speed Data Channel Timing

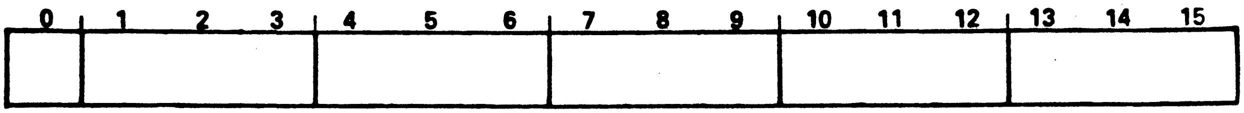
Section 5
STANDARD INSTRUCTION SET

5.1 INTRODUCTION

This section explains the function and use of POINT 4 MARK 8 instructions. Included is a discussion of two's-complement notation, addressing modes, and the individual instructions in the memory reference, arithmetic/logical, and input/output instruction groups. Input/output instructions and interrupt handling instructions are presented, with details given for special code-77 (CPU) instructions.

5.2 OCTAL REPRESENTATION AND TWO'S COMPLEMENT NOTATION

The computer uses 16-bit binary words for program instructions and data. The bits are numbered 0 through 15 with bit 0 the most significant bit (MSB) and bit 15 the least significant bit (LSB). For convenience, binary words are represented in 6-digit octal form. Each octal digit represents three bits and can have values between 0 and 7, except the most significant digit which represents a single bit and has a maximum value of 1. The POINT 4 MARK 8 16-bit binary word format is shown in Figure 5-1.



082-23

Figure 5-1. POINT 4 MARK 8 16-bit Binary Word Format

The reader is presumed to be familiar with binary and octal notations. For a simple review, the following example shows the correspondence between decimal, binary and octal representation:

Decimal	Binary	Octal	
0	0000000000000000	000000	
1	0000000000000001	000001	
2	0000000000000010	000002	
8	0000000000001000	000010	
64	0000000001000000	000100	
5407	0001010100011111	012437	
32,767	0111111111111111	077777	(15 bit max.)
65,535	1111111111111111	177777	(16 bit max.)

The computer represents negative numbers in two's-complement form. Signed positive and negative numbers are used both as 16-bit operands and as 8-bit address displacements in memory reference instructions. A review of two's complement arithmetic follows.

In two's-complement arithmetic, positive and negative values are distinguished by a 0 or 1 in the leftmost bit position (sign bit). Positive numbers have a sign bit of 0, with the numerical value expressed in ordinary binary form by the remaining bits. Negative numbers have a sign bit value of 1 and the numerical value expressed in two's-complement form. The two's complement is found by taking the one's complement or logical complement of the number including the sign bit (changing all 0's to 1's and all 1's to 0's) and adding 1.

The number zero is represented by 0's in all bit positions. There is only one representation for zero, since the two's complement of zero is also zero. Zero is a nonnegative value. For this reason also, there is one more negative number than there are nonzero positive numbers.

The range of signed, 8-bit fields is as follows:

	Binary	Octal
	Representation	Value
Largest positive	01 111 111	+177
	01 111 110	+176
	•	
	•	
	•	
	00 000 001	+1
	00 000 000	0
	11 111 111	-1
	11 111 110	-2
	•	
	•	
	•	
	10 000 001	-177
Most negative	10 000 000	-200

5.3 INSTRUCTION TYPES

From the programmer's point of view, the POINT 4 MARK 8 Computer is comprised of four accumulators, 64K words of memory and an input/output bus. The instructions control and manipulate the data flowing between these elements.

Instruction words can be classified into one of the following three categories:

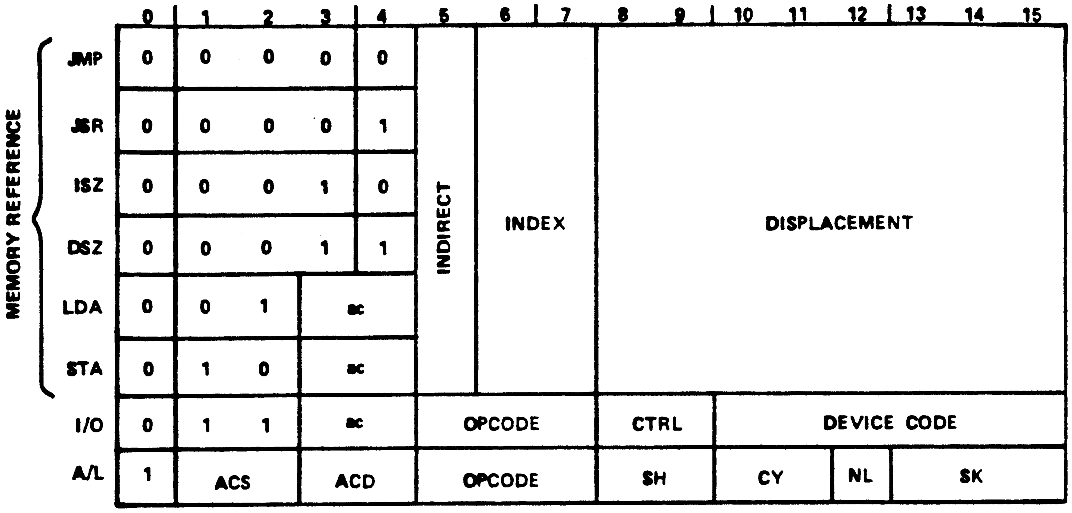
1. Memory Reference Instructions are instructions that reference a memory location. These include:

   LDA - Load an accumulator from memory
   STA - Store an accumulator into memory
   JMP - Jump to another location in memory
   JSR - Jump to a subroutine in memory
   ISZ - Increment memory and skip if zero
   DSZ - Decrement memory and skip if zero

2. Arithmetic/Logic Instructions are instructions that specify a particular arithmetic or logical operation to be performed on one or two operands stored in the accumulators, and allow for testing the result for skip conditions.

3. Input/Output Instructions are instructions for input/output operations with a specific peripheral device.

Figure 5-2 is an overview of the formats for each type of instruction. Each of these three classes is discussed in detail in the succeeding subsections.



ac	=	Accumulator
CTRL	=	Control pulse
ACS	=	Source accumulator
ACD	=	Destination accumulator
SH	=	Shift control
CY	=	Carry preselection
NL	=	No-load
SK	=	Skip condition

082–24

Figure 5-2. POINT 4 MARK 8 Instruction Format Summary

5.4 MEMORY REFERENCE INSTRUCTIONS

Six memory reference instructions are used to move data between memory locations and accumulators, to transfer program control to a new location, and to modify and test memory words. The memory reference instructions fall into three general categories, as follows:

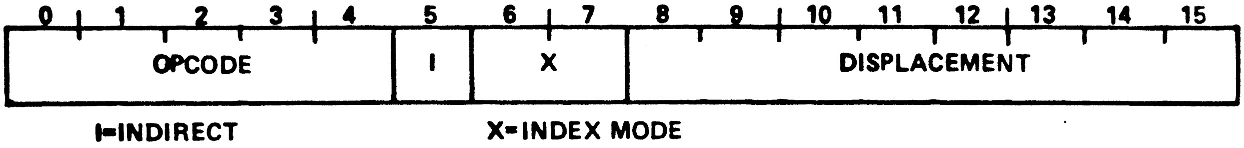
1. Move Data Instructions: LDA, STA
2. Jump Instructions: JMP, JSR
3. Modify Memory Instructions: ISZ, DSZ

Before describing the function of each instruction in this group it is necessary to describe the way in which they address memory.

5.4.1 MEMORY ADDRESSING

Each memory reference instruction uses one of several addressing modes to determine an effective memory address, E. The processor accesses the location specified by the effective memory address and uses the contents as the operand of the instruction.

The Jump instructions (JMP, JSR) and the Modify Memory instructions (ISZ, DSZ) both use the binary format shown in Figure 5-3.

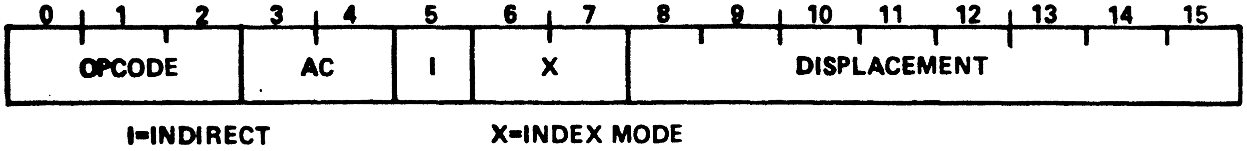


082-25

Figure 5-3. Jump and Modify Memory Instruction Binary Word Format

Bits 0-4 of the instruction word are the OPCODE field. Bit 5 is the indirect or I field; bits 6 and 7 are the Index Mode or X field; and bits 8-15 are the displacement or D field.

The Move Data instructions (LDA, STA) use the binary format shown in Figure 5-4.



082-26

Figure 5-4. Move Data Instruction Binary Word Format

Table 5-1 defines the memory reference instructions, indicating bits, their fields and field definitions.

All addresses, both direct and indirect, are entered into the Effective Address Register. When this register contains the effective address E, the instruction specified in bits 0 through 4 of the command is executed.

TABLE 5-1. MEMORY REFERENCE INSTRUCTIONS

Bits	Field	Definition
0-2	OPCODE	Determines which instruction is performed. For the Move Data instructions (LDA, STA), bits 0, 1 and 2 make up the OPCODE field. For the Jump instructions (JMP, JSR), and Modify Memory instructions (ISZ, DSZ), bits 0-4 make up the OPCODE field.
3,4	Accumulator (ac)	The ac field for the Move Data instructions (LDA, STA) specifies one of the four general accumulators. The specified accumulator will either have data stored in it or data transferred from it.
5	Indirect (I)	Determines whether the X and D fields specify the effective address (E) directly or whether indirect addressing is to be used.
6,7	Index Mode (X)	Defines one of the four addressing modes. Each addressing mode may be thought of as a page of 256 words which the instruction can address directly.
8-15	Displacement (D)	Specifies the word addressed on the selected page.

5.4.1.1 Indexing Mode

The X field selects one of the indexing modes shown in Table 5-2.

TABLE 5-2. INDEXING MODES

Bits 6-7	Definition
00	Page Zero - Page zero is defined as the first 256 memory locations (addresses in the range from 000000 to 000377 octal). The effective memory address in page zero addressing is equal to the value of the D field which is an unsigned binary integer that can have values from 000 octal to 377 octal.
01	Relative Addressing - In the relative addressing mode the address placed in the Effective Address Register is equal to the address in the Program Counter (PC) plus the value of the displacement in the D field. In this case the displacement D is a signed binary integer. Bit 8 is the sign (0 = positive 1 = negative) and the integer may have any value in the range from -200 to +177 octal (decimal -128 to +127). The address in PC can be visualized as the center of a 256-word page and any address between the bottom (128 words below the PC) and top (127 words above PC) of the page can be specified by the displacement D.
10 or 11	Base Register Addressing - In the base register addressing mode the address placed in the Effective Address Register is equal to the address in accumulator register A2 (code 10) or A3 (code 11) plus the value of the displacement in the D field. In this case the displacement D is a signed binary integer. Bit 8 is the sign (0 = positive 1 = negative) and the integer may have any value in the range from -200 octal to +177 octal (decimal -128 to +127). The address in A2 or A3 can be visualized as the center of a 256-word page and any address between the bottom (128 words below A2 or A3) and top (127 words above A2 or A3) of the page can be specified by the displacement D.

5.4.1.2 Indirect Addressing Operations

When the I field (bit 5) of the Memory Reference Instruction contains a 1, an indirect addressing sequence is required. In this case, the address in the Effective Address Register (determined by the X and D fields) is the memory address from which a second address word is to be fetched.

This address word can be interpreted as follows:

- If the most significant bit of the address word equals 0, this address word is the effective address, E.

- If the most significant bit of the address word equals 1, the action taken depends on whether the processor is set in 32K or 64K addressing mode, as described below.

When the processor is in 32K (normal) mode, a second level of indirect addressing is allowed. In this case, if the most significant bit (bit 0) of the address word fetched from memory contains a 1, another level of indirect addressing is required, and the address word in the Effective Address Register specifies the address word to be fetched from memory. The process continues until an address word with bit 0=0 is found. Through programming error it is possible to become caught in an infinite loop of indirect addressing. Caution should be taken when using indirect addressing to avoid this problem.

When 64K addressing is enabled, a second level of indirect addressing is not permitted. In this case all 16 bits of the word fetched from memory are used as the effective address. A 1 in bit 0 of the address word simply indicates an address in the upper 32K words (100000-177777) of memory.

5.4.1.3 Automatic Incrementing and Decrementing of Locations

If at any time during the indirect addressing sequence, the Effective Address Register contains an address in the range from 000020 octal to 000037 octal, the following auto-indexing action is performed:

1. The contents of the memory location specified by the Effective Address Register are fetched, and the contents are either incremented or decremented by one, as follows:

    - If the address is in the range 000020 octal through 000027 octal, the contents are incremented.

    - If the address is in the range 000030 octal through 000037 octal, the contents are decremented.

2. The incremented or decremented value is written back into the same memory location from which the value was fetched in step 1.

3. The incremented or decremented value produced in step 1 is stored in the Effective Address Register and used for the next level of indirect addressing (if bit 0=1 and the processor is set to 32K addressing mode), or for the effective addresss (if bit 0=0 or the processor is in 64K addressing mode).

NOTE

> The value of bit 0 following incrementing or decrementing controls continuation of indirect addressing.

5.4.2 TYPES OF MEMORY REFERENCE INSTRUCTIONS

When the Effective Address Register contains the effective address E, one of two groups of memory reference instructions is performed as determined by the operation codes. Refer to Section 5.4.1 for basic memory reference instruction formats and field definitions. Refer to Appendix A (Von Neumann Map of the POINT 4 MARK 8 Command Structure) for octal formats of each instruction, and to Appendix B (POINT 4 MARK 8 Instruction Reference Chart) for octal-to-symbolic conversion of memory reference instructions.

5.4.2.1 Move Data Instructions

When the code in bits 1 and 2 of the OPCODE field is not 00, and the effective address (E) is in the Effective Address Register, two operations are performed, as shown in Table 5-3.

TABLE 5-3. MOVE DATA INSTRUCTIONS

Bits 1-2	OPCODE	Definition
01	LDA	Load Accumulator Instruction - The contents of memory location E are stored in the accumulator specified by the ac field (bits 3 and 4). The contents of E are unaffected; the original contents of the accumulator are lost.
10	STA	Store Accumulator Instruction - The data in the accumulator specified by the ac field is transferred to memory location E. The contents of the accumulator are unaffected; the original contents of E are lost.

5.4.2.2 Jump and Modify Memory Instructions

When bits 0, 1 and 2 are all zero, and the effective address (E) is in the Effective Address Register, one of four operations is performed. The operation is specified by the code in bits 3 and 4 of the OPCODE extension field. These jump and modify memory instructions are shown in Table 5-4.

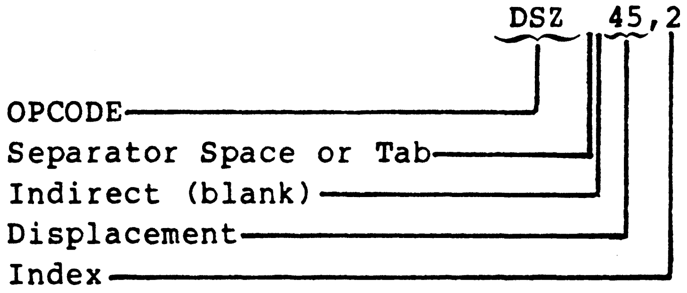
TABLE 5-4. JUMP AND MODIFY MEMORY INSTRUCTIONS

Bits 1-4	OPCODE	Definition
0000	JMP	Jump Instruction - The effective address E is transferred from the Effective Address Register to the Program Counter (PC). The next instruction is then fetched from jump address E and sequential execution is continued from there.
0001	JSR	Jump to Subroutine Instruction - After the effective address E has been calculated the address in PC is incremented and the incremented value is stored in accumulator A3. Then the effective address E is transferred from the Effective Address Register to the Program Counter (PC). The next instruction is then fetched from jump address E. Execution of another JMP or JSR instruction that specifies A3 will cause the program to return to the address in A3 plus or minus any desired displacement D.
0010	ISZ	Increment and Skip if Zero - The contents of effective address E are fetched, incremented and written back into address E. If the incremented value is equal to zero, PC is incremented by one to skip the next instruction.
0011	DSZ	Decrement and Skip if Zero - The contents of the location specified by effective address E are decremented and written back into address E. If the decremented value is equal to zero, PC is incremented by one to skip the next instruction.

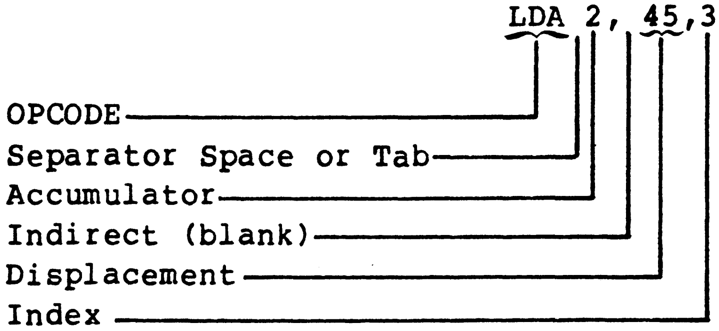
5.4.2.3 Assembler Language Conventions and Addressing Examples

The assembler language memory reference instruction consists of the instruction OPCODE mnemonic (STA, LDA, JMP, etc.) followed by symbols that specify the accumulator, the addressing mode and the memory address. The assembler program translates these statements into binary code which the processor executes. Table 5-5 shows the programming conventions for memory reference instructions.

The format for modify memory and jump instructions requires the instruction mnemonic and a memory address, including indirect addressing displacement, and indexing indicator. The assembly language instruction will be formatted as follows:



The move data instructions LDA and STA also require that an accumulator (A0-A3) be specified. For example



Fields that are not specified will be assembled containing 0s. An "@" symbol denotes indirect addressing and places a 1 in bit 5 of the instruction. An example of indirect page-zero (X Field = 00) addressing is as follows:

```
    LDA 1,@20
```

Relative addressing is formatted as follows:

```
    LDA 0,. +15
```

The symbol "." indicates X = 01 (relative addressing) and thus "." represents the current value of the program counter.

TABLE 5-5. ASSEMBLER LANGUAGE CONVENTIONS FOR MEMORY REFERENCE INSTRUCTIONS

Instruction Function	OPCODE Mnemonic	Separator Space or Tab	Accumulator Number	,	Memory Address			
					Indirect	Displcmt	,	Index
Load Accumulator	LDA		ac*	,	blank or @	Displacement	,	
Store Accumulator	STA							blank
Jump	JMP		none					1**
Jump Subroutine	JSR							2
Increment and Skip if Zero	ISZ							3
Decrement and Skip if Zero	DSZ							

*ac = 0, 1, 2, 3 representing A0, A1, A2, A3

**Instead of "displacement,1" the following sequence may be used: ".±displacement"

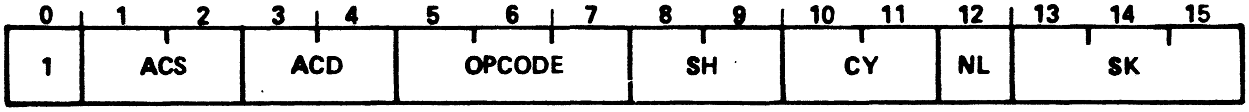
5.5 ARITHMETIC AND LOGICAL INSTRUCTION GROUP

The eight arithmetic/logical instructions perform binary addition subtraction and logical functions on 16-bit operands. These instructions are:

- Arithmetic: ADD, ADC, INC, SUB, NEG
- Logical: MOV, COM, AND

All Arithmetic and Logic instructions contain a 1 in bit 0 and have their basic Arithmetic/Logical Unit (ALU) function specified by bits 5-7 as shown in Figure 5-5. The fields of the Arithmetic/Logical instruction format are as follows:

- Source Accumulator (ACS)
- Destination Accumulator (ACD)
- OPCODE
- Shifter/Swapper (SH)
- Carry Preselect (CY)
- No-Load (NL)
- Skip Condition Tester (SK)



0	1 2	3 4	5 6 7	8 9	10 11	12	13 14 15
1	ACS	ACD	OPCODE	SH	CY	NL	SK

082-27

Figure 5-5. Arithmetic/Logical Instruction Format

5.5.1 ARITHMETIC AND LOGICAL PROCESSING

The organization of the arithmetic/logical processing unit must be described before discussion of the eight arithmetic and logical instructions and their auxiliary control fields. Subsystem organization is shown in Figure 5-6, and described in the following subsections.

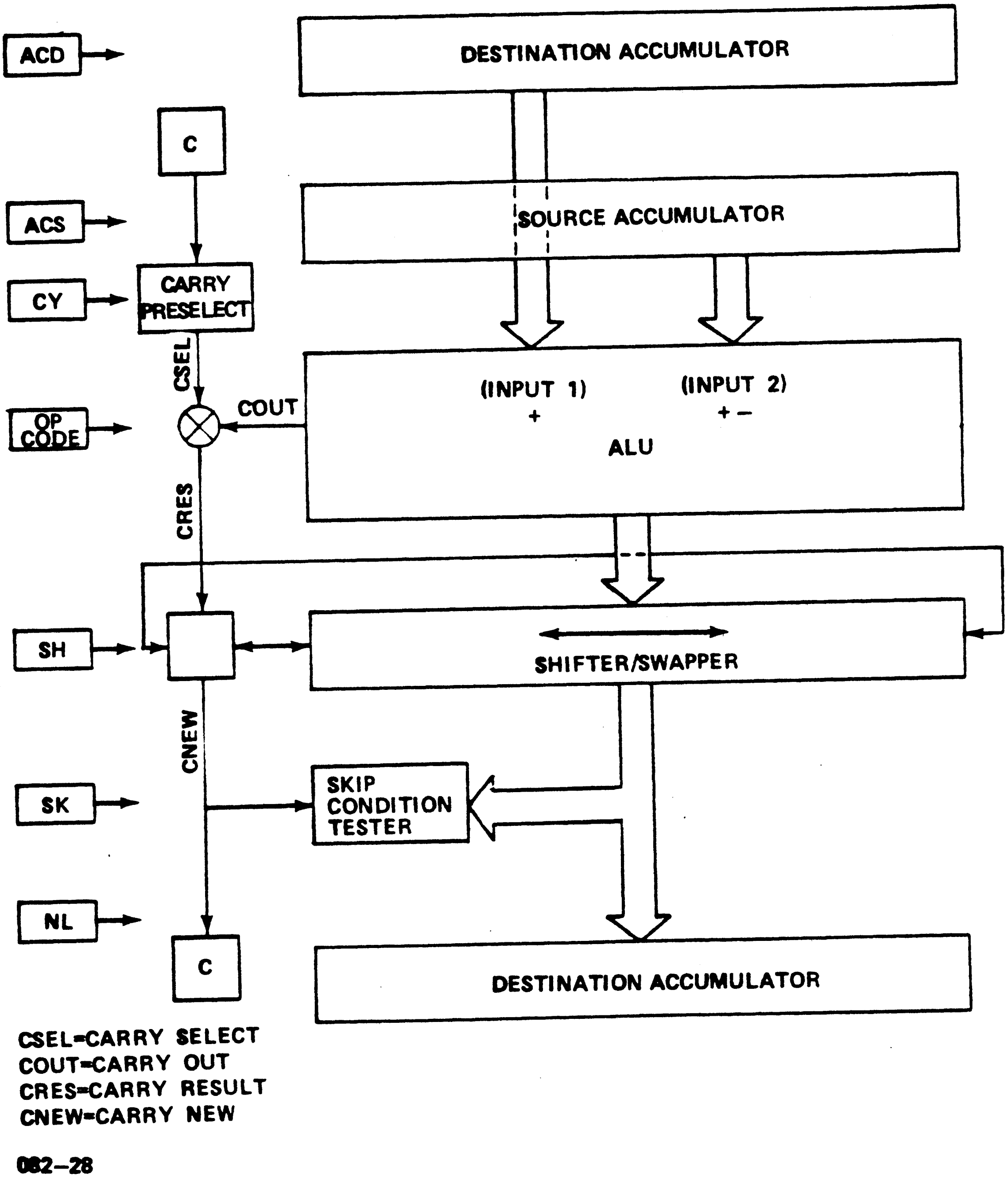


Figure 5-6. Arithmetic/Logical Operations

5.5.1.1 Arithmetic/Logical Operations

The heart of the subsystem is the Arithmetic/Logic Unit (ALU) which performs the actual addition, subtraction or logical operation. It has provision for two inputs:

Input 1: Comes from the accumulator selected by the ACD field and is used only in the operations which require two operands (ADD, SUB, ADC and AND).

Input 2: Comes from the accumulator selected by the ACS field and is used in all operations.

The ALU performs the arithmetic or logical operation specified by the OPCODE field (bits 5-7). The result of this operation may cause a carry-out to occur from the most significant bit of the ALU. In the case of an operation which adds unsigned integers, a carry-out is equivalent to overflow; however this is not always true. See Section 5.5.1.2 for a more complete discussion of carry and overflow operation.

If the result of the arithmetic or logical operation involves a carry-out (COUT) the carry preselected by the CY field of the instruction (CSEL) is complemented. The resulting carry (CRES) together with the 16-bit operation result generated by the ALU is applied as a 17-bit operand to the Shifter where a shift-left, shift-right or swap may occur as determined by the SH field of the instruction. After shifting, the carry (CNEW) and the 16-bit operation result are loaded into the Carry Flag (C) and the destination accumulator (ACD) unless this is prevented by a 1 in the No-Load (NL) field. In either case they are tested for a skip condition (i.e., to determine if the next instruction should be skipped) as specified in the SK field of the instruction.

5.5.1.2 Overflow and Carry-Out Operations

The 16-bit numbers processed by the ALU may be thought of as unsigned integers between 0 and 64K or as signed integers between -32K and +32K.

Binary Number	Unsigned Interpretation		Signed Interpretation	
(in ALU)	Octal	Decimal	Octal	Decimal
1111111111111111	177777	64K-1	-00001	-1
1111111111111110	177776	64K-2	-00002	-2
•				
•				
•				
1000000000000001	100001	32K+1	-77777	-32K+1
1000000000000000	100000	32K	-100000	-32K
0111111111111111	077777	32K-1	+77777	32K-1
0111111111111110	077776	32K-2	+77776	32K-2
•				
•				
•				
0000000000000001	000001	1	+00001	1
0000000000000000	000000	0	00000	0

When working with either interpretation there is the possibility of an overflow (answer greater than the maximum number that can be represented) or underflow (less than the minimum). In general, the ALU will produce the correct result if no overflow or underflow occurs, and will produce 64K more than or less than the correct result if there is underflow or overflow, respectively.

The relationship between underflow/overflow and the carry-out from the ALU MSB is shown in the following paragraphs.

1. Unsigned integers:

   Decimal: 0 <= x < 64K
   Octal: 0 <= x <= 177777

   When ADDing two numbers, if the true result is less than 64K, the ALU will produce the correct result and no carry-out will result. If the true result is greater than or equal to 64K, the ALU will produce 64K less than the true result (i.e., the true result truncated to 16 bits) and a carry-out will result. Note that in these cases, a carry-out is synonymous with overflow and indicates that the ALU output is not the true result.

   SUBtraction is accomplished in the ALU by complementing the subtrahend and adding it to the minuend with a carry-in. Therefore, when SUBtracting one unsigned integer from another, if the true result is positive or zero, the ALU will produce the true result and will also produce a carry-out.

If the true result is negative, the ALU will produce the true result plus 64K (since all numbers are interpreted as positive), and no carry-out will result. Note that in these cases a carry-out is the opposite of underflow and indicates that the ALU output is the true result.

2. Signed Integers:

```
    Decimal:  -32K <= x < 32K
    Octal:    -100000 <= x <=77777
```

When ADDing two positive integers (or SUBtracting a negative integer from a positive one), if the true result is less than 32K, the ALU will produce the true result and no carry-out. If the true result is greater than or equal to 32K, the ALU output will appear negative (since the MSB = 1), will be 64K less than the true result, and no carry-out will occur. Note that in this case an overflow is not signalled by a carry-out.

When ADDing two integers with opposite signs (or SUBtracting two numbers having the same sign) the ALU will always produce the true result since the true result must be between -32K and +32K. A carry-out will occur if the result is positive and not if it is negative.

When ADDing two negative numbers (or SUBtracting a positive number from a negative one) if the true result is greater than or equal to -32K, the ALU will produce the true result. If the true result is less than -32K the ALU output will appear positive (MSB=0) and will be 64K greater than the true result. In either case, a carry-out will always occur.

These relationships are illustrated in Figure 5-7.

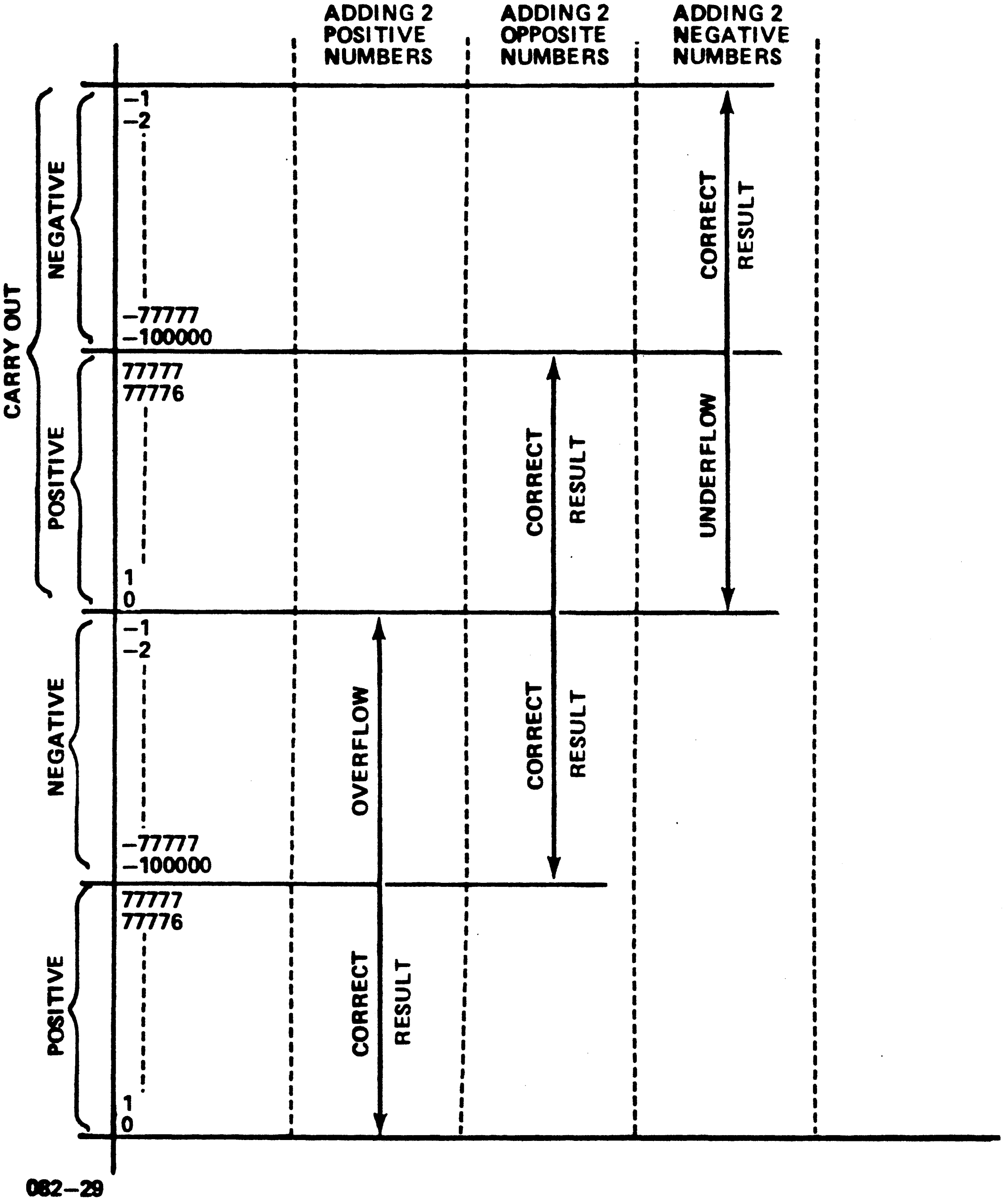


Figure 5-7. Overflow and Carry Operations Analysis for Signed Integers

5.5.2 ARITHMETIC/LOGIC FUNCTIONS

The OPCODE field (bits 5 through 7) defines one of eight arithmetic/logic operations to be performed by the 16-bit ALU as shown in Table 5-6.

TABLE 5-6. ARITHMETIC/LOGIC FUNCTIONS

Bits 5-7	OPCODE	Definition
000	COM	Complement - Complement the contents of ACS. Do not modify the preselected carry bit.
001	NEG	Negate - Produce the two's complement of the contents of ACS. If ACS=0, complement the preselected carry bit.
010	MOV	Move - Supply the unmodified contents of ACS. Do not modify the preselected carry bit.
011	INC	Increment - Add 1 to the contents of ACS. If the result is 0, complement the preselected carry bit.
100	ADC	Add Complement - Add the complement of ACS to ACD. Complement the preselected carry bit if ACS is less than ACD.*
101	SUB	Subtract - Subtract ACS from ACD. Complement the preselected carry bit if ACS is less than or equal to ACD.*
110	ADD	Add - Add the contents of ACS to the contents of ACD. If the unsigned sum is greater than or equal to two to the sixteenth power, complement the preselected carry bit.
111	AND	And - Logically AND the contents of ACS with the contents of ACD. Do not modify the preselected carry bit.
*Using a 16-bit unsigned integer interpretation.		

5.5.3 SECONDARY FUNCTIONS

The Shift (SH), Carry (CY), No-load (NL), and Skip (SK) fields specify secondary operations performed on the ALU result produced by the OPCODE field. These fields are discussed in the sections that follow.

5.5.3.1 Shift Field (SH)

The SH field (bits 8 and 9) determines the shifting action (if any) produced by the Shifter on the result of the calculation produced by the ALU, as shown in Table 5-7.

TABLE 5-7. SHIFT FIELD DEFINITIONS

Bits 8-9	Mnemonic	Definition
00	-	No Shift - Do not modify the ALU result. The carry resulting from the ALU operation is unaffected.
01	L	Left Rotate - Shift the result one place to the left, and insert the state of the carry resulting from the ALU (CRES) in the LSB (bit 15) position. Insert the out-shifted MSB (bit 0) into the carry bit (CNEW).
10	R	Right Rotate - Shift the result one place to the right, and insert the state of the carry resulting from the ALU (CRES) into the MSB (bit 0) position. Insert the out-shifted LSB (bit 15) into the carry bit (CNEW).
11	S	Swap - Swap the 8 MSBs of the result with the eight LSBs. The carry resulting from the ALU is unaffected.

5.5.3.2 Carry Control Field (CY)

The CY field (bits 10 and 11) specifies the base to be supplied to the ALU for carry calculation, as shown in Table 5-8.

TABLE 5-8. CARRY CONTROL FIELD

Bits 10-11	Mnemonic	Definition
00	-	No change - The current state of the carry flag is supplied to the ALU as a base for carry calculation.
01	Z	Zero - The value 0 is supplied to the ALU as a base for carry calculation.
10	O	One - The value 1 is supplied to the ALU as a base for carry calculation.
11	C	Complement - The complement of the current state of the carry flag is supplied to the ALU as a base for carry calculation.

The three logical functions (MOV, COM, AND) supply the values listed above as the carry bit to the Shifter. The five arithmetic functions (ADD, ADC, INC, SUB, NEG) supply the complement of the base value if the ALU operation produces a carry-out of bit 0; otherwise they supply the value listed above.

5.5.3.3 No-Load Field (NL)

The NL field (bit 12) determines whether or not the output of the Shifter is stored in ACD and in Carry. If bit 12=0, the Shifter output is stored in ACD and in Carry. If bit 12=1, no storage action occurs.

5.5.3.4 Skip Control Field (SK)

The SK field determines the type of skip test to be performed on the Shifter output. If the selected skip test is affirmative, the next instruction is skipped. The skip tests that can be selected by the SK field (bits 13-15) are shown in Table 5-9.

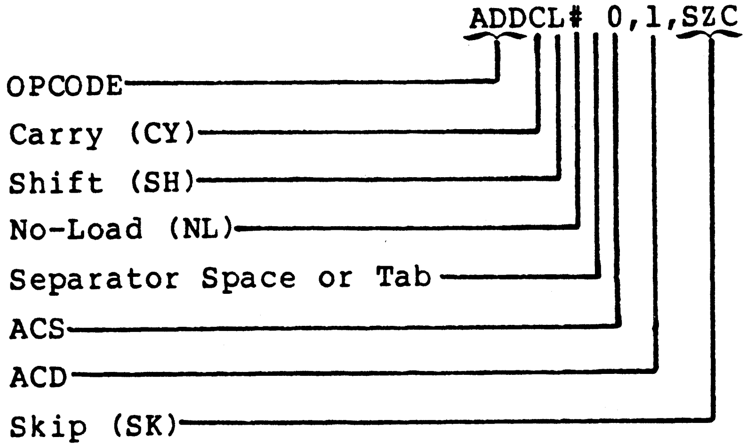
TABLE 5-9. SKIP CONTROL FIELD

Bits 13-15	Mnemonic	Definition
000	-	No skip test (never skip)
001	SKP	Skip unconditionally (no skip test required)
010	SZC	Skip if carry bit is zero
011	SNC	Skip if carry bit is nonzero
100	SZR	Skip if result is zero
101	SNR	Skip if result is nonzero
110	SEZ	Skip if either carry bit or result is zero
111	SBN	Skip if both carry bit and result are nonzero

5.5.4 ASSEMBLER LANGUAGE CONVENTIONS AND EXAMPLES

The assembler language arithmetic or logical instruction consists of the instruction OPCODE mnemonic (ADD, NEG, COM, etc.) followed by symbols that specify the carry indicator, the shift indicator, the load/no-load indicator, a source and a destination accumulator and the skip conditions. Table 5-10 shows the programming conventions for arithmetic and logical instructions.

The format is as follows:



The CY, SH, NL, and SK fields are specified by adding the appropriate mnemonic symbols. None of these four fields has to be specified, but their symbols must appear in the proper order and place if they are included. Those fields not specified will be assembled containing 0s. For example

```
    ADDCL 0,1
```

performs the following operation: Add A0 to A1 and supply the complement of the Carry flag to the ALU. Shift the 17-bit output to the left, and store it into A1 and the Carry flag.

TABLE 5-10. ASSEMBLER LANGUAGE CONVENTIONS FOR ARTHMETIC AND LOGICAL INSTRUCTIONS

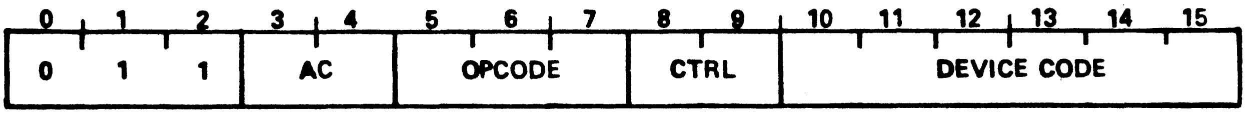
Instruction Function	OPCODE Mnemonic	Optional Secondary Functions			Separator Space or Tab	Accumulators				Optnl Skip SK*
		CY*	SH*	NL*		ACS	,	ACD	,	
Add	ADD									blank
Subtract	SUB									SKP
Move	MOV	none	none			0		0		SZC
Increment	INC	Z	L	none		1		1		SNC
Negate	NEG	O	R	#		2	,	2	,	SZR
Complement	COM	C	S			3		3		SNR
Add Complement	ADC									SEZ
Logical And	AND									SBN

*Elimination of a mnemonic symbol for these fields will cause the field to be assembled as all zeros.

5.6 INPUT/OUTPUT INSTRUCTION GROUP

The input/output instructions enable the processor to communicate with the peripheral devices on the system and also perform various operations within the processor. I/O instructions transfer data between accumulators and devices, start or reset device operation, or check the status of each device. Each I/O instruction contains a 6-bit device code field, which specifies the particular device for this data transfer. The system allows up to 63 peripheral devices, with each device assigned a unique code from 00 through 76 octal. The 77 octal code denotes a special class of instructions that controls certain CPU functions such as interrupt handling. Use of the 00 code is not recommended, since a device with that code would give a default response to an Interrupt Acknowledge instruction.

All instruction words in this category have the format shown in Figure 5-8.



082-30

Figure 5-8. Input/Output Instruction Format

An instruction in this class is designated by 011 in bits 0-2. The OPCODE and Control (CTRL) fields define the I/O operation to be performed. If a data transfer operation is involved, the ac field (bits 3 and 4) specifies the accumulator involved in the data transfer (otherwise it has no effect). Bits 10-15 select the device that is to respond to the instruction.

5.6.1 REGULAR I/O INSTRUCTIONS

Regular I/O instructions apply to all device codes except code 77. These instructions fall into two basic categories, depending on whether they transfer data or test the state of the device:

1. I/O Transfer: NIO, DIA, DOA, DIB, DOB, DIC, DOC

2. I/O Skip: SKPBN, SKPBZ, SKPDN, SKPDZ

The POINT 4 MARK 8 input/output system provides for specification of the following functions:

- Full 16-bit data transfer:
  - Three input channels (DIA, DIB, DIC)
  - Three output channels (DOA, DOB, DOC)

- Control Functions:

    - Three outgoing control pulses (START, CLEAR, or IOPULSE - designated by S, C or P, respectively).

    - Two device flags (Busy and Done) that can be sensed by I/O Skip instructions

These input/output functions are illustrated in Figure 5-9.

Each device interface contains a 6-bit address decoder (bits 10-15). When the processor executes an I/O instruction, it places the specified device code onto the Device Select lines of the I/O Bus. The appropriate device will recognize its own code and thus respond to the I/O instruction. All other devices ignore the instruction.

The Control (CTRL) field can have two different functions, depending on the category into which the instruction falls:

- In conjunction with a Data Transfer Instruction, one of the three different control pulses may be sent to the device - START, CLEAR, or IOPULSE

- In conjunction with I/O Skip Instructions, the control field determines which of the two flags in the I/O device will be tested - Busy or Done

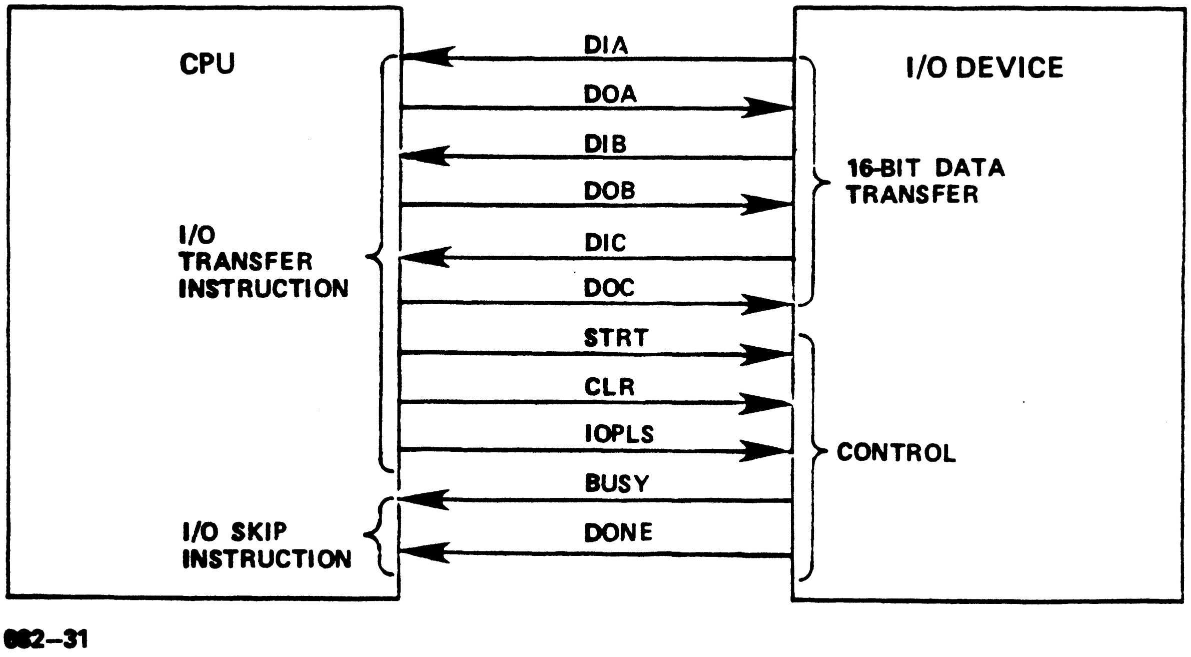


Figure 5-9. CPU-I/O Device Input/Output Functions

5.6.1.1 I/O Transfer and Device Control Instructions

I/O Transfer instructions move data between the processor and the device interface. There are six possible device buffers, labeled A, B and C. Each label may refer to two separate buffers - an input (read) and an output (write) buffer.

The OPCODE field (bits 5-7) of the instruction specifies the type of transfer to take place (Data In, Data Out, No Transfer, etc.). Bits 3 and 4 specify the accumulator that supplies or receives the data and bits 8 and 12 specify a control function (if any). The type of transfer is determined by the code in the OPCODE field as shown in Table 5-11.

TABLE 5-11. I/O TRANSFER TYPE SPECIFICATION

Bits 5-7	OPCODE	Definition
000	NIO	No data transfer involved. Device control only.
001	DIA	Move the contents of the A buffer in device D to the accumulator specified in the ac field, and send the control pulse specified by the Control field to the selected device.
010	DOA	Move the contents of the accumulator specified in the ac field to the A buffer in device D, and send the control pulse specified by the Control field to the selected device. The original contents of ac are unaffected.
011	DIB	Move the contents of the B buffer in device D to the accumulator specified in the ac field, and send the control pulse specified by the Control field to the selected device.
100	DOB	Move the contents of the accumulator specified in the ac field to the B buffer in device D, and send the control pulse specified by the Control field to device D. The original contents of ac are unaffected.
101	DIC	Move the contents of the C buffer in device D to the accumulator ac, and send the control pulse specified by the Control field to device D.
110	DOC	Move the contents of the accumulator specified in the ac field to the C buffer in device D, and send the control pulse specified by the Control field to device D. The original contents of ac are unaffected.

The Control field (bits 8 and 9) for I/O Transfer Instructions determines which control pulse should be transmitted, if any. The processor first performs the data transfer and then outputs the pulse. The Control field is defined for regular I/O transfer instructions as shown in Table 5-12.

TABLE 5-12. CONTROL FIELD SPECIFICATION

OPCODE Bits 5-7	Control Bits 8-9	Control Mnemonic	Definition
000-110	00	None	None
000-110	01	S	Produce the STRT pulse. Typically this starts the device by clearing its Done flag, setting its Busy flag, and clearing its interrupt request flag.
000-110	10	C	Produce the CLR pulse. Typically this clears both the Busy and Done flags, and the interrupt request flag, idling the device.
000-110	11	P	Pulse the special I/O bus control line (IOPLS). The effect, if any, depends upon the device.

5.6.1.2 I/O Skip Instructions

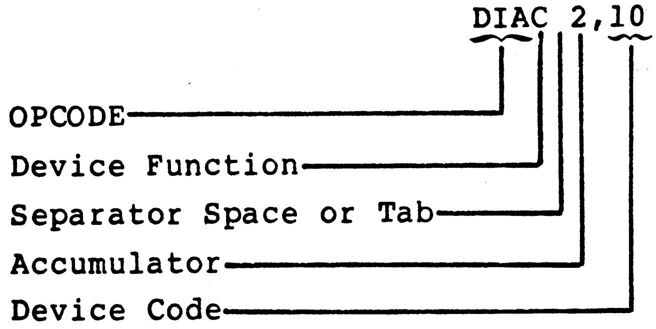
When the OPCODE field (bits 5-7) contains 111, the Control field (bits 8 & 9) selects the flag to be tested in the conditional I/O skip. The control codes for skip operations are shown in Table 5-13.

TABLE 5-13. I/O SKIP INSTRUCTIONS

OPCODE Bits 5-7	Control Bits 8-9	Control Mnemonic	Definition
111	00	SKPBN	Skip the next instruction if the Busy flag in the device is nonzero.
111	01	SKPBZ	Skip the next instruction if the Busy flag in the device is zero.
111	10	SKPDN	Skip the next instruction if the Done flag in the device is nonzero.
111	11	SKPDZ	Skip the next instruction if the Done flag in the device is zero.

5.6.1.3 Assembler Language Conventions and Examples

An assembler language I/O Transfer Statement consists of the instruction mnemonic, an optional control function, an accumulator and octal device code. Table 5-14 shows the programming conventions for regular input/output instructions. For example



This instruction performs the function: Move the data from register A of device 10 into A2. Clear (reset) the device.

The device code may be represented by a device mnemonic. Thus,

```
    DIAC 2,TTI
```

is equivalent to the previous example because TTI represents device 10 (input buffer for a Teletype or CRT terminal).

A No I/O (NIO) or I/O Skip instruction will not specify an accumulator since no data transfer occurs:

```
    NIOS TTI

    SKPBZ TTI
```

TABLE 5-14. ASSEMBLER LANGUAGE CONVENTIONS FOR INPUT/OUTPUT INSTRUCTIONS

Instruction Function	OPCODE Mnemonic	Optional Device Function	Separator Space or Tab	Acc	,	Device Code
No Input/ Output	NIO			none	none	00-76 octal
Data In Buffer A	DIA	S Start* C Clear* P General Pulse*		0 1 2 3	,	
Data Out Buffer A	DOA					
Data In Buffer B	DIB					
Data Out Buffer B	DOB					
Data In Buffer C	DIC					
Data Out Buffer C	DOC					
Skip if Busy Flag is Nonzero	SKPBN					
Skip if Busy Flag is Zero	SKPBZ					
Skip if Done Flag is Nonzero	SKPDN					
Skip if Done Flag is Zero	SKPDZ					

*Optional

5.6.2 SPECIAL CODE 77 (CPU) INSTRUCTIONS

Certain system functions, setting and testing of processor flags and interrupt processing control are accomplished via I/O instructions with the octal code 77 in bits 10-15. These instructions do not directly address a particular device and the device code mnemonic is CPU.

CPU instructions have the same general format as regular I/O instructions. The OPCODE field and Control field, however, are interpreted differently.

OPCODE Field:

- Channel A (DIA) is used to read the mini-switches at the front edge of the CPU board, or the DATA display on the optional Operator Control Unit. DOA is not defined.

- Channel B addresses all I/O devices simultaneously for certain interrupt control functions.

- Channel C does no data transfer. DIC and DOC are used for resetting all I/O devices and for halting the computer, respectively.

Control pulses:

- S and C are used to enable or disable interrupts.

- P is used to set 32K or 64K addressing mode, depending on the state of the LSB of A0 (see Table 5-15).

The CPU has two flags which can be tested by the I/O Skip instructions:

- Busy = ION set (Interrupts are enabled)

- Done = Power-failure has been detected (will cause interrupt if ION set)

The assembler also recognizes several special mnemonics for CPU instructions.

Table 5-15 gives both the regular instruction mnemonic and the special mnemonic. It also provides a definition of the special function of the CPU instructions.

TABLE 5-15. SPECIAL CPU INSTRUCTIONS

Instruction	Special Mnemonic	Definition
NIO CPU		No action.
NIOS CPU	INTEN	Set the processor's Interrupt On (ION) flag. The processor will now respond to interrupt requests from devices, after execution of one more instruction.
NIOC CPU	INTDS	Clear the Interrupt On flag, so that the processor will not respond to interrupt requests.
NIOP CPU	None	Set 32K or 64K addressing mode, depending on the LSB value of A0. An LSB value of 0 sets 32K mode; a value of 1 sets 64K mode. Set or clear EIS mode, depending on the value of the MSB. An MSB value of 0 clears EIS mode; a value of 1 sets EIS mode.
DIA ac,CPU	READS ac	Read the setting of the mini-switches at the front edge of the CPU board (or the value in the DATA readout of the optional Operator Control Unit, if installed) into accumulator ac.
DIB ac,CPU	INTA ac	Read the device code of the highest priority device requesting an interrupt into accumulator ac.
DOB ac,CPU	MSKO ac	Set up Interrupt Disable flags in all devices simultaneously, according to the mask code in accumulator ac. Each device is associated with one of the bits in the accumulator, and its flag is set (mask bit=1) disabling interrupt from that device, or cleared (mask bit=0) enabling interrupts from it. A MSKO with ac=177777 disables interrupts from all devices.
DICC ac,CPU	IORST	Generate I/O Reset to clear the Busy, Done, and Interrupt Disable flags in all devices. This instruction also clears the processor's ION flag and sets it to 32K addressing mode. (Does not change contents of selected accumulator.)
DOC ac,CPU	HALT	Halt the processor. Requires manual action to restart processor.

Note that the special mnemonic does not allow the programmer to specify the S and C functions. For example

```
    READS 3
```

when executed, deposits the value entered via the Operator Control Unit or mini-switches into A3. If the programmer wishes to also set ION, the assembler instruction mnemonic would have to be used:

```
    DIAS 3,CPU
```

This instruction sets ION after reading the DATA display on the Operator Control Unit, or the mini-switches.

The instruction IORST, however, assumes the C function. All I/O device flags are reset and the ION flag is cleared. In order to reset the I/O devices without clearing the ION flag, the regular assembler instruction must be used:

```
    DIC 0,CPU
```

As with regular I/O instructions, a value of 111 in bits 5-7 signifies a conditional skip instruction. The function field in this case indicates which processor flag (Interrupt On or Power Fail) will be tested, as shown in Table 5-16.

TABLE 5-16. CONDITIONAL SKIP INSTRUCTIONS

Bits 8&9	Instruction	Definition
00	SKPBN CPU	Skip next instruction if Interrupt On is nonzero.
01	SKPBZ CPU	Skip next instruction if Interrupt On is zero.
10	SKPDN CPU	Skip next instruction if the Power Failure flag is nonzero.
11	SKPDZ CPU	Skip next instruction if the Power Failure flag is zero.

5.6.2.1 Assembler Language Conventions

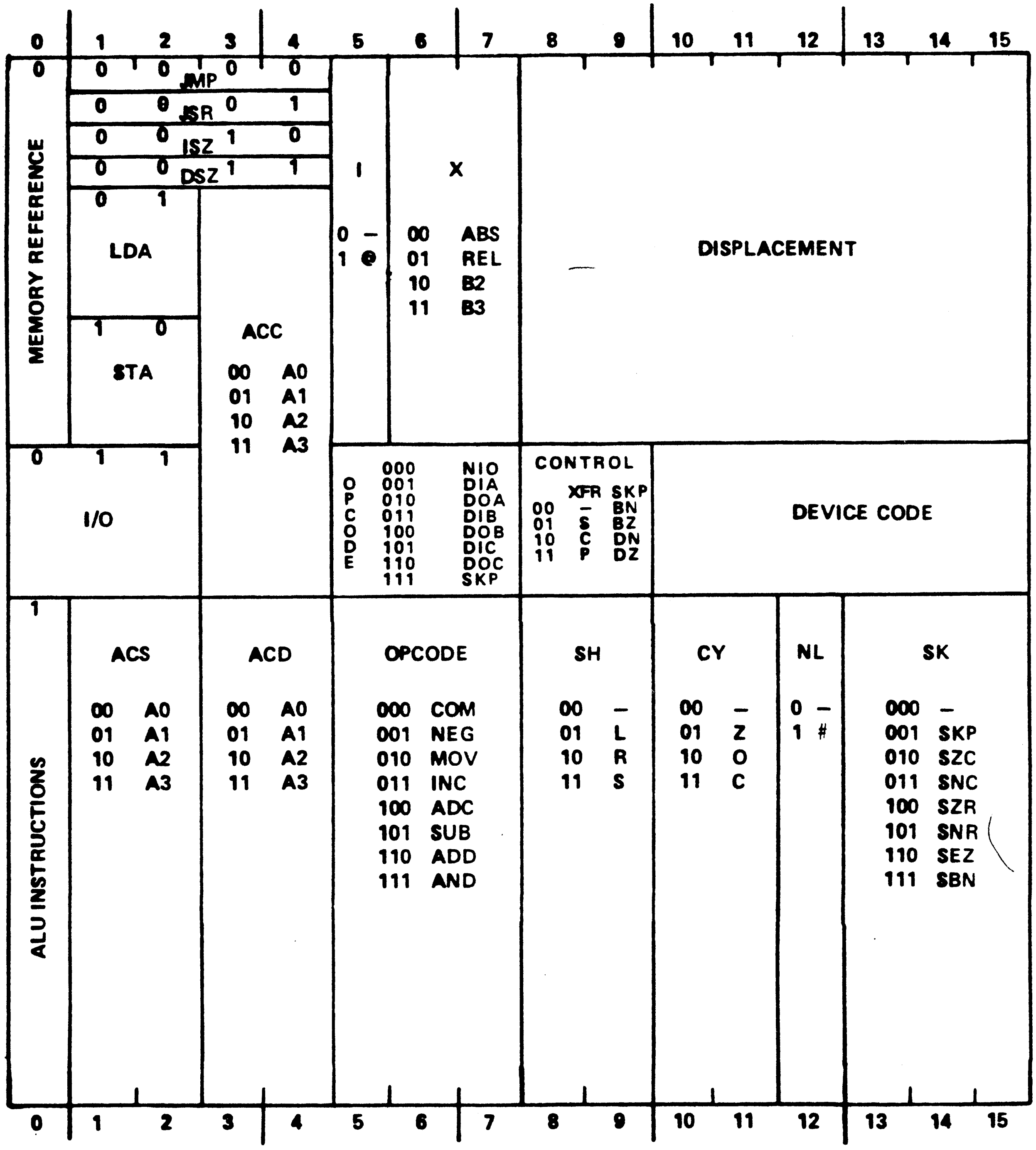
CPU instructions are usually written using the special mnemonics shown in Section 5.6.2; however, they may also be written in the same manner as regular I/O instructions, specifying the instruction mnemonic, optional control function, optional accumulator, and a device code of 77 octal (mnemonic CPU). For example

```
    NIOS CPU
```

sets the ION flag in the processor.

APPENDICES

Appendix A
VON NEUMANN MAP OF POINT 4 MARK 8 INSTRUCTIONS



082-37

Appendix B
POINT 4 MARK 8 INSTRUCTION REFERENCE CHART

ARITH/LOGIC

100000	COM
100400	NEG
101000	MOV
101400	INC
102000	ADC
102400	SUB
103000	ADD
103400	AND

SOURCE	
0	0
20000	1
40000	2
60000	3

DESTINATION	
0	0
4000	1
10000	2
14000	3

SHIFT	
100	L
200	R
300	S

CARRY	
20	Z
40	O
60	C

NO-LOAD	
10	#

SKIP CONDITION	
1	SKP
2	SZC
3	SNC
4	SZR
5	SNR
6	SEZ
7	SBN

MEMORY REFERENCE

0	JMP
4000	JSR
10000	ISZ
14000	DSZ
20000	LDA
40000	STA

ACCUMULATOR	
0	0
4000	1
10000	2
14000	3

INDIRECT	
2000	@

ADDRESS MODE	
0	ABS
400	REL
1000	BASE2
1400	BASE3

DISPLACEMENT	
0–177	POS.
200–377	NEG.

SPECIAL ARITHMETIC TESTS

101014	SKZ
101015	SNZ
101112	SSP
101113	SSN
102032	SGE
102033	SLS
102414	SEQ
102415	SNE
102432	SGR
102433	SLE

INPUT/OUTPUT

60000	NIO
60400	DIA
61000	DOA
61400	DIB
62000	DOB
62400	DIC
63000	DOC

ACCUMULATOR	
0	0
4000	1
10000	2
14000	3

I/O PULSE	
100	S
200	C
300	P

I/O SKIP	
63400	SKPBN
63500	SKPBZ
63600	SKPDN
63700	SKPDZ

DEVICE CODE	
10	TTI
11	TTO
12	PTR
13	PTP
14	RTC
17	LPT

SPECIAL CPU INSTRUCTIONS	
60177	INTEN
60277	INTDS
60477	READS
61477	INTA
62077	MSKO
62677	IORST
63077	HALT

082–38

Appendix C
ASCII CODE CHART

ASCII CODE in OCTAL

000	NUL	<CTRL-@>	040	BLANK	100	@	140	`
001	SOH	<CTRL-A>	041	!	101	A	141	a
002	STX	<CTRL-B>	042	"	102	B	142	b
003	ETX	<CTRL-C>	043	#	103	C	143	c
004	EOT	<CTRL-D>	044	$	104	D	144	d
005	ENO	<CTRL-E>	045	%	105	E	145	e
006	ACK	<CTRL-F>	046	&	106	F	146	f
007	BEL	<CTRL-G>	047	'	107	G	147	g
010	BKSP	<CTRL-H>	050	(	110	H	150	h
011	HTAB	<CTRL-I>	051	)	111	I	151	i
012	LF	<CTRL-J>	052	*	112	J	152	j
013	VTAB	<CTRL-K>	053	+	113	K	153	k
014	FF	<CTRL-L>	054	,	114	L	154	l
015	CR	<CTRL-M>	055	-	115	M	155	m
016	SO	<CTRL-N>	056	.	116	N	156	n
017	SI	<CTRL-O>	057	/	117	O	157	o
020	DLE	<CTRL-P>	060	0	120	P	160	p
021	XON	<CTRL-Q>	061	1	121	Q	161	q
022	AUXON	<CTRL-R>	062	2	122	R	162	r
023	XOFF	<CTRL-S>	063	3	123	S	163	s
024	AUXOFF	<CTRL-T>	064	4	124	T	164	t
025	NAK	<CTRL-U>	065	5	125	U	165	u
026	SYN	<CTRL-V>	066	6	126	V	166	v
027	ETB	<CTRL-W>	067	7	127	W	167	w
030	CAN	<CTRL-X>	070	8	130	X	170	x
031	ENDMD	<CTRL-Y>	071	9	131	Y	171	y
032	SUB	<CTRL-Z>	072	:	132	Z	172	z
033	ESC	<CTRL-[>	073	;	133	[	173	{
034	F SEP	<CTRL-\>	074	<	134	\	174	|
035	G SEP	<CTRL-]>	075	=	135	]	175	}
036	R SEP	<CTRL-^>	076	>	136	^	176	~
037	U SEP	<CTRL-_>	077	?	137	_	177	DEL

Appendix D
VIRTUAL FRONT PANEL COMMANDS

APL does Auto-Boot if mini switches set to 200 + device code.

A — Display PC, A0, A1, A2, A3, and carry.

Cx,y (x<5) — Change accumulator x (or carry if x=4) to value y.

Cx (x>4) — Convert real address x to virtual.

Dx* — Dump memory in octal, beginning at address x.

Ex* — Enable entry at address x.

Fx,y — Establish offset x-y, where x=real memory address, y=virtual (listing) address.

Jx — Jump to location x, after restoring accumulators and carry.

Kx,y,z — Store constant z in memory locations x through y.

Mx,y,z — Move memory block x through y to location z.

Nx,y,z,m — Search memory x through y for not-equal to z, using mask m (optional).

Ox* — Output memory in ASCII, starting at location x, until a zero byte is encountered.

*Address x may be followed by mode designator 0, 1, 2, or 3:

Mode	Meaning
None	Word address, including "F" offset, if any
0	Word address, absolute
1	Byte address, using offset, if any
2	Byte address, lower 32K
3	Byte address, upper 32K

Px Program load from DMA device code x. If x omitted, reads switches.

Sx,y,z,m Search memory x through y for the value z, using mask m (optional).

Xx,y Calculate checksum over x through y.

Yx Set up a delay after each CR/LF. x=0 for maximum delay; x=177777 for none.

x:y Enter value y at address x, and open next cell for entry.

^ Open previous cell for entry.

CTRL ( ) Transmit Control Character to CTU.

Appendix E
PROGRAMMING EXAMPLES

E.1 NUMBER HANDLING

E.1.1 GENERATING NUMBERS

Six numbers can be generated with single instructions:

```
        SUB     0,0 --> 0
        SUBZL   0,0 --> 1
        SUBZR   0,0 --> 100000
        ADC     0,0 --> 177777  (= -1)
        ADCZL   0,0 --> 177776  (= -2)
        ADCZR   0,0 --> 77777
```

E.1.2 NUMBER TESTING

Twenty different sets of numbers can be tested with a single instruction. Figure E-1 shows the conditions under which each of the basic arithmetic test instructions will skip. Figure E-2 shows which instructions to use to test the 20 sets of numbers. The skip condition can be changed from a zero-test to a nonzero-test to obtain the complements of the 20 sets.

Instr.	SZR skips if:	SZC skips if:
MOVZ	0	all
MOVO	0	none
MOVZL	0,100000	>=0
MOVOL	none	>=0
MOVZR	0,1	even
MOVOR	none	even
COMZ	-1	0
COMO	-1	none
COMZL	-1,-2	<0
COMOL	none	<0
COMZR	-1,77777	odd
COMOR	none	odd
INCZ	-1	not -1
INCO	-1	-1
INCZL	77777	-1<=x<=77777
INCOL	-1	-1<=x=77777
INCZR	0	odd
INCOR	-1	odd
NEGZ	0	not 0
NGEO	0	-1
NEGZL	100000	-77777<=X<0
NEGOL	0	-77777<=x<0
NEGZR	-1	even
NEGOR	0	even
ADDZ	0,100000	>=0
ADDO	0,100000	<0
ADDZL	0	2d bit = 0
ADDOL	100000	2d bit = 0
ADDZR	0	all
ADDOR	100000	all

NOTE

For ADD instructions, ACS and ACD must be the same.

Figure E-1. Conditions Under Which Each Of The Basic Arithmetic Test Instructions Will Skip

											+ ←	→ −										
INSTRUCTION	TESTS FOR	0	1	2	3	37776	37777	40000	40001	77776	77777	100000	100001	137776	137777	140000	140001	177774=-4	177775=-3	177776=-2	177777=-1	
MOV 0,0,SZR	0	●																				1 VALUE
INCZL 0,0,SZR	77777										●											
NEGZL 0,0,SZR	100000											●										
COM 0,0,SZR	-1																				●	
MOVZR 0,0,SZR	0 , 1	●	●																			2 VALUES
COMZR 0,0,SZR	-1 , 2																			●	●	
MOVZL 0,0,SZR	0 , 100000	●										●										
COMZL 0,0,SZR	77777 , -1										●										●	
MOVL 0,0,SZC	≥0	●	●	●	●	●	●	●	●	●	●											32K
INCL 0,0,SZC	-1 ≤ X ≤ 77776	●	●	●	●	●	●	●	●	●											●	
NEGL 0,0,SZC	-77777 ≤ X ≤ 0	●											●	●	●	●	●	●	●	●	●	
MOVR 0,0,SZC	EVEN	●		●		●		●		●		●		●		●		●		●		
ADDL 0,0,SZC	2D BIT = 0	●	●	●	●	●	●					●	●	●	●							
MOVZR 0,0,SEZ	EVEN OR 1	●	●	●		●		●		●		●		●		●		●		●		32K ÷ 1
NEGZR 0,0,SEZ	EVEN OR -1	●		●		●		●		●		●		●		●		●		●	●	
INCZR 0,0,SEZ	ODD OR 0	●	●		●		●		●		●		●		●		●		●		●	
COMZR 0,0,SEZ	ODD OR-2		●		●		●		●		●		●		●		●		●	●	●	
MOVZL 0,0,SEZ	≥0 OR 100000	●	●	●	●	●	●	●	●	●	●	●										32K + 1
COMZL 0,0,SEZ	<0 OR 77777										●	●	●	●	●	●	●	●	●	●	●	
NEGZL 0,0,SEZ	≤0	●										●	●	●	●	●	●	●	●	●	●	

082-39

Figure E-2. The 20 Different Sets Of Numbers Which Can Be Tested With A Single Instruction

E.2 BIT TESTING

Any bit in a word can be tested with a maximum of three instructions, without requiring another accumulator. Three bit positions can be tested with just one instruction (bits 0, 1, and 15). Seven bit positions (bits 2, 3, 7, 8, 9, 10 and 14) require two instructions, and the other six require three. Figure E-3 shows which instructions to use to test for any bit in a word.

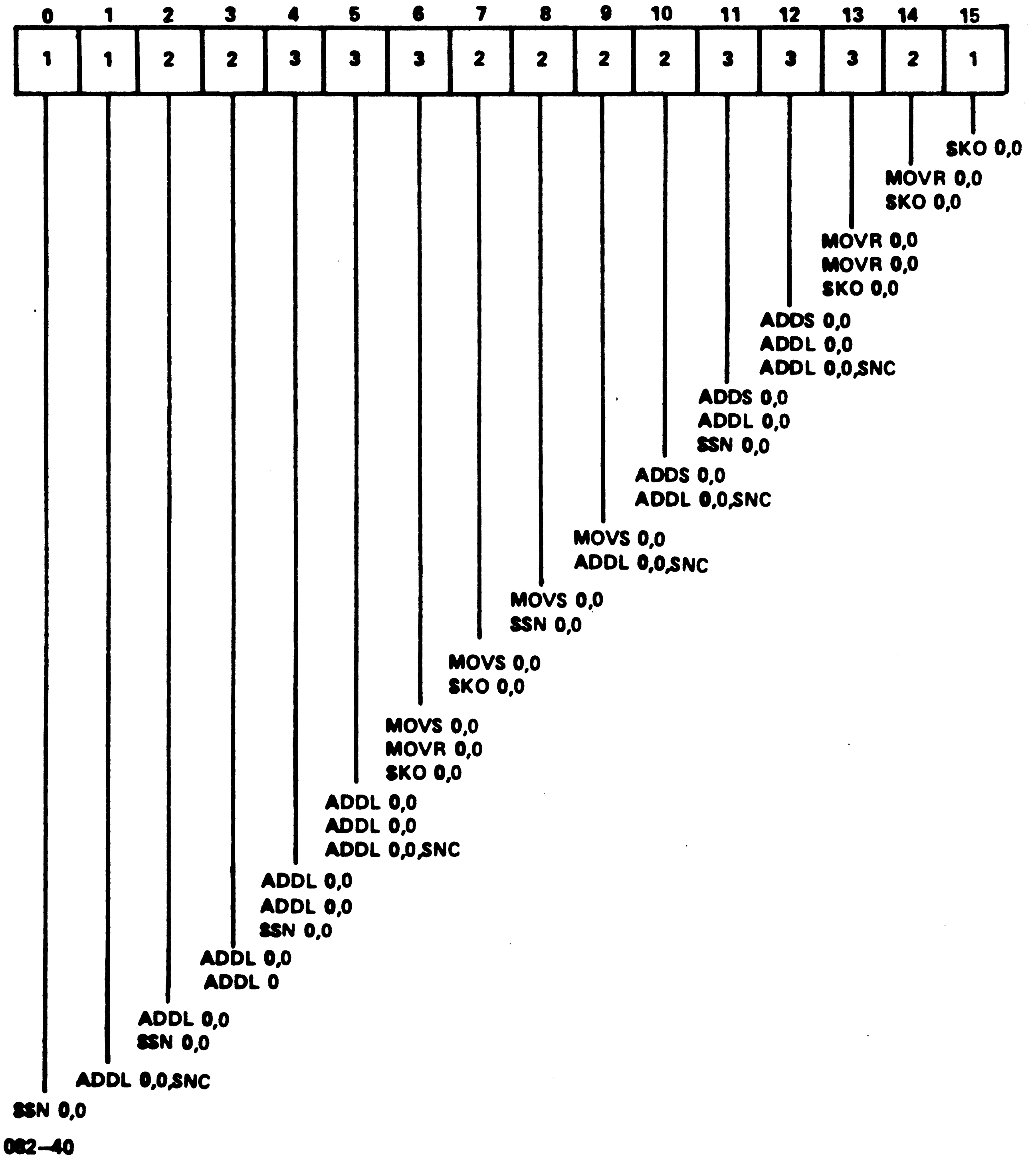


Figure E-3. How To Test For Any Bit In A Word

E.3 ACCUMULATOR HANDLING

E.3.1 TO "OR" TWO ACCUMULATORS

The following routine forms the inclusive-OR of A0 and A1 in A1. The routine uses the fact that an arithmetic ADD is equivalent to an OR if corresponding bits in the two operands are not both 1.

```
    COM    0,0
    AND    0,1   ;remove those bits where both are 1
    ADC    0,1   ;then add original value
```

E.3.2 TO "EXCLUSIVE-OR" TWO ACCUMULATORS

This routine utilizes the fact that an arithmetic ADD is the same as an exclusive-OR except for the carry bits.

```
    MOV    1,2
    ANDZL  0,2   ;form the carry bits
    ADD    0,1   ;add the original operands
    SUB    2,1   ;remove the carry bits
```

This routine destroys the carry flag. To preserve the Carry at the expense of an additional instruction, use the following:

```
    COM    1,2               __
    AND    0,2   ;forms A0 * A1
    COM    0,0    __
    AND    0,1   ;A0 * A1        __   __
    ADD    2,1   ;A0 + A1 = A0 * A1 + A0 * A1
```

E.3.3 TO "DECREMENT" AN ACCUMULATOR

The following routine is used to decrement an accumulator:

```
    NEG    0,0
    COM    0,0
```

E.3.4 TO "COMPLEMENT" THE MOST SIGNIFICANT BIT

The following instruction complements the MSB and clears the carry bit:

```
    ADDOR  0,0
```

E.4 PARITY GENERATION OR CHECKING

The two-instruction loop

```
ADD   0,0,SZR
JMP   .-1
```

will complement the original carry if A0 had odd parity, and leave it unchanged if A0 had even parity.

E.5 I/O PROGRAMMING FOR THE MASTER TERMINAL

The assembler listings in Figure E-4 provide examples of inputting and outputting to a Master Terminal (Teletype or CRT), using the standard Device Code 10/11-type controller. They also illustrate byte handling and interrupt handling conventions. For further information on I/O interfaces, see Section 4.

```
; PROGRAMMING EXAMPLES

           1000       .LOC   1000

; INPUT/OUTPUT ROUTINES

; MASTER TERMINAL INPUT - ACCEPTS CHARACTER INTO A0

    1000  63610       SKPDN  TTI      ;IS THERE ANY INPUT AVAILABLE ?
    1001    777       JMP    .-1      ;  NO, KEEP WAITING
    1002  60610       DIAC   0,TTI    ;YES, ACCEPT IT & CLEAR TTI DONE FLAG

;   NOTE: FOR PAPER TAPE READER USE DIAS TO START READING NEXT CHARACTER


; MASTER TERMINAL OUTPUT - ASSUMES OUTPUT CHARACTER IS IN A0

    1003  63511       SKPBZ  TTO      ;TTO STILL BUSY FROM A PREV. OUTPUT ?
    1004    777       JMP    .-1      ;  YES, WAIT FOR IT TO FINISH
    1005  61111       DOAS   0,TTO    ;  NO, OUTPUT THE CHAR. AND START TTO
```

Figure E-4. Master Terminal I/O Programming Examples (1 of 4)

```
- PAGE 2 -

; SUBROUTINE TO TYPE ASCII TEXT

; INITIAL CONDITIONS: NONE
; CALLING SEQUENCE:
;     JSR    TYPE
;     (ASCII TEXT,
;     PACKED 2 CHARACTERS/WORD
;     WITH 0 BYTE TERMINATOR)
;     RETURNS HERE
; RETURN CONDITIONS: A0 = 0, A2 = PRESERVED

    1006  25400 TYPE: LDA    1,0,3     ;PICK UP 2 ASCII CHARACTERS
    1007 175420       INCZ   3,3       ;ADV. RETURN PNTR; C=BYTE CNTR
    1010  20411 TYPE2:LDA    0,C377L   ;LEFT-BYTE MASK
    1011 123705       ANDS   1,0,SNR   ;EXTRACT A BYTE - IS IT 0 ?
    1012   1400       JMP    0,3       ;  YES, RETURN TO CALLER
    1013  63511       SKPBZ  TTO       ;  NO, WAIT FOR TTO NOT BUSY
    1014    777       JMP    .-1
    1015  61111       DOAS   0,TTO     ;OUTPUT THE CHARACTER
    1016 125362       MOVCS  1,1,SZC   ;SWAP THE 2 ASCII CHAR.; CK. BYTE CNT
    1017    771       JMP    TYPE2     ;  TYPE THE SECOND CHARACTER
    1020    766       JMP    TYPE      ;  GET 2 MORE CHARACTERS TO TYPE

    1021 177400 C377L:177400           ;OCTAL 377 IN LEFT BYTE
```

```
; SUBROUTINE TO TYPE A NUMBER IN OCTAL FORM

; INITIAL CONDITIONS: A1 = NUMBER TO BE TYPED
; CALLING SEQUENCE:
;     JSR    TPOCT
;     RETURNS HERE
; RETURN CONDITIONS: A1 = 0, A2 = PRESERVED, C = 1

    1022  20413 TPOCT:LDA    0,C.BIT
    1023 101120 TPOC2:MOVZL  0,0       ;PRESET CARRY (MSB:1, OTHERS:0)
    1024 125105       MOVL   1,1,SNR   ;LEFT-SHIFT A BIT OUT OF A1 INTO C
    1025   1400       JMP    0,3       ;RETURN WHEN PUSHER BIT IS GONE
    1026 101103       MOVL   0,0,SNC   ;ASSEMBLE ASCII DIGIT; COMPLETE ?
    1027    775       JMP    .-3       ;  NO, GET MORE BITS
    1030  63511       SKPBZ  TTO       ;WAIT FOR TTO NOT BUSY
    1031    777       JMP    .-1
    1032  61111       DOAS   0,TTO     ;OUTPUT THE ASCII DIGIT
    1033  20403       LDA    0,C.OCT
    1034    767       JMP    TPOC2     ;CONTINUE THE LOOP

    1035 140014 C.BIT:140014 ;CONST. TO STRIP OFF 1 BIT & CNVT. TO ASCII
    1036  10003 C.OCT:010003 ;CONST. TO STRIP OFF 3 BITS & CNVT. TO ASCII
```

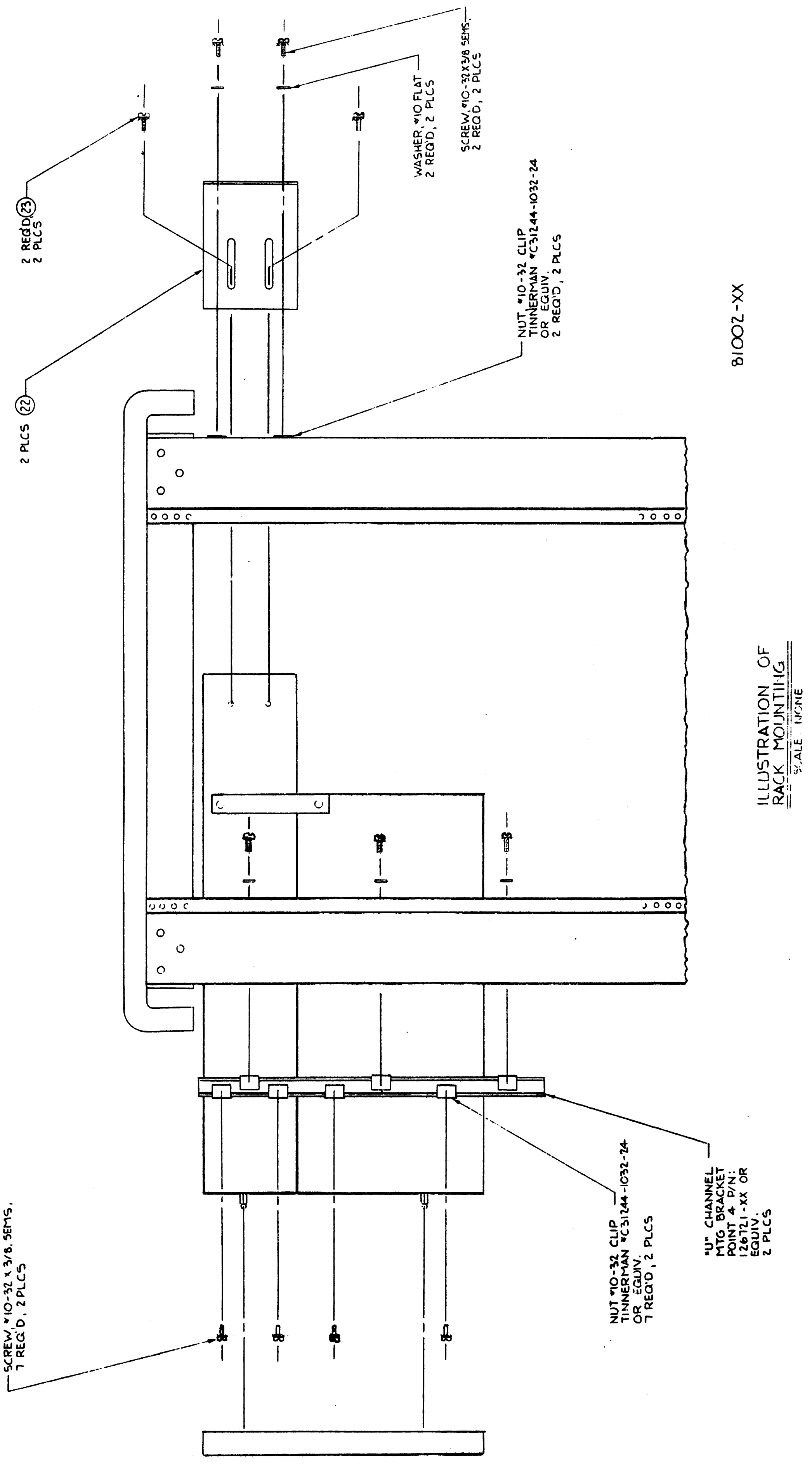
Figure E-4. Master Terminal I/O Programming Examples (2 of 4)

```
- PAGE 3 -

; BYTE MOVE SUBROUTINES

; ASSUMPTION: ALL BYTE ADDRESSES REFER TO LOWER 32K OF MEMORY


; GET A BYTE INTO A0 FROM BYTE ADDRESS GIVEN IN A1

; INITIAL CONDITIONS: A1 = BYTE ADDRESS
; CALLING SEQUENCE:
;     JSR    GETBY
;     RETURNS HERE
; RETURN CONDITIONS: A0 = DESIRED BYTE, A1 = UNCHANGED

     1037 131220 GETBY:MOVZR  1,2       ;CONVERT BYTE ADDRESS INTO WORD ADDRESS
     1040  21000       LDA    0,0,2     ;FETCH WORD CONTAINING DESIRED BYTE
     1041 101003       MOV    0,0,SNC   ;DO WE WANT LEFT BYTE ?
     1042 101300       MOVS   0,0       ;  YES, SWAP THE WORD
     1043  30403       LDA    2,C377    ;RIGHT BYTE MASK
     1044 143400       AND    2,0       ;MASK THE RIGHT BYTE
     1045   1400       JMP    0,3       ;RETURN

     1046    377 C377: 377


; PUT A BYTE FROM A0 INTO MEMORY AT BYTE ADDRESS GIVEN IN A1

; INITIAL CONDITIONS: A0 = GIVEN BYTE IN RIGHT HALF, LEFT HALF IMMATERIAL
;     A1 = BYTE ADDRESS
; CALLING SEQUENCE:
;     JSR    PUTBY
;     RETURNS HERE
; RETURN CONDITIONS: A0, A1 UNCHANGED

     1047  54414 PUTBY:STA    3,PUTBR   ;SAVE RETURN ADDRESS
     1050 131220       MOVZR  1,2       ;FORM WORD ADDRESS FROM BYTE ADDR.
     1051  34775       LDA    3,C377    ;GET MASK FOR RIGHT HALF
     1052 163403       AND    3,0,SNC   ;MASK GIVEN BYTE; GOES IN LEFT HALF ?
     1053 101301       MOVS   0,0,SKP   ;  YES, SWAP THE BYTE
     1054 175300       MOVS   3,3       ;  NO, SWAP THE MASK
     1055  25000       LDA    1,0,2     ;FETCH THE WORD WHERE BYTE IS TO GO
     1056 167400       AND    3,1       ;MAKE ROOM FOR THE BYTE
     1057 107000       ADD    0,1       ;INSERT THE BYTE
     1060  45000       STA    1,0,2     ;PUT THE WORD BACK
     1061 145100       MOVL   2,1       ;RESTORE A1
     1062   2401       JMP    @PUTBR    ;RETURN

     1063      0 PUTBR:0                ;SAVE RETURN ADDRESS
```

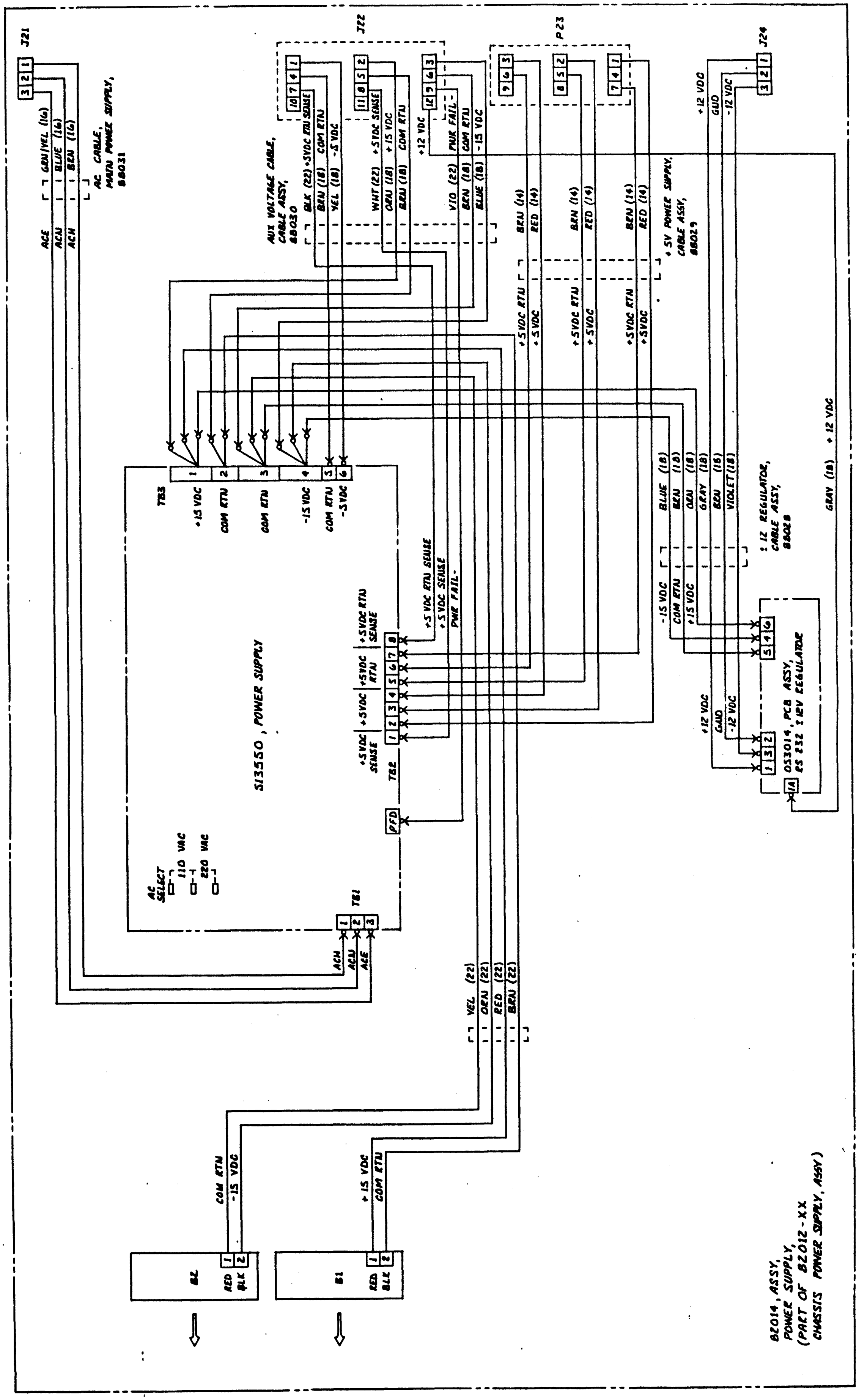
Figure E-4. Master Terminal I/O Programming Examples (3 of 4)

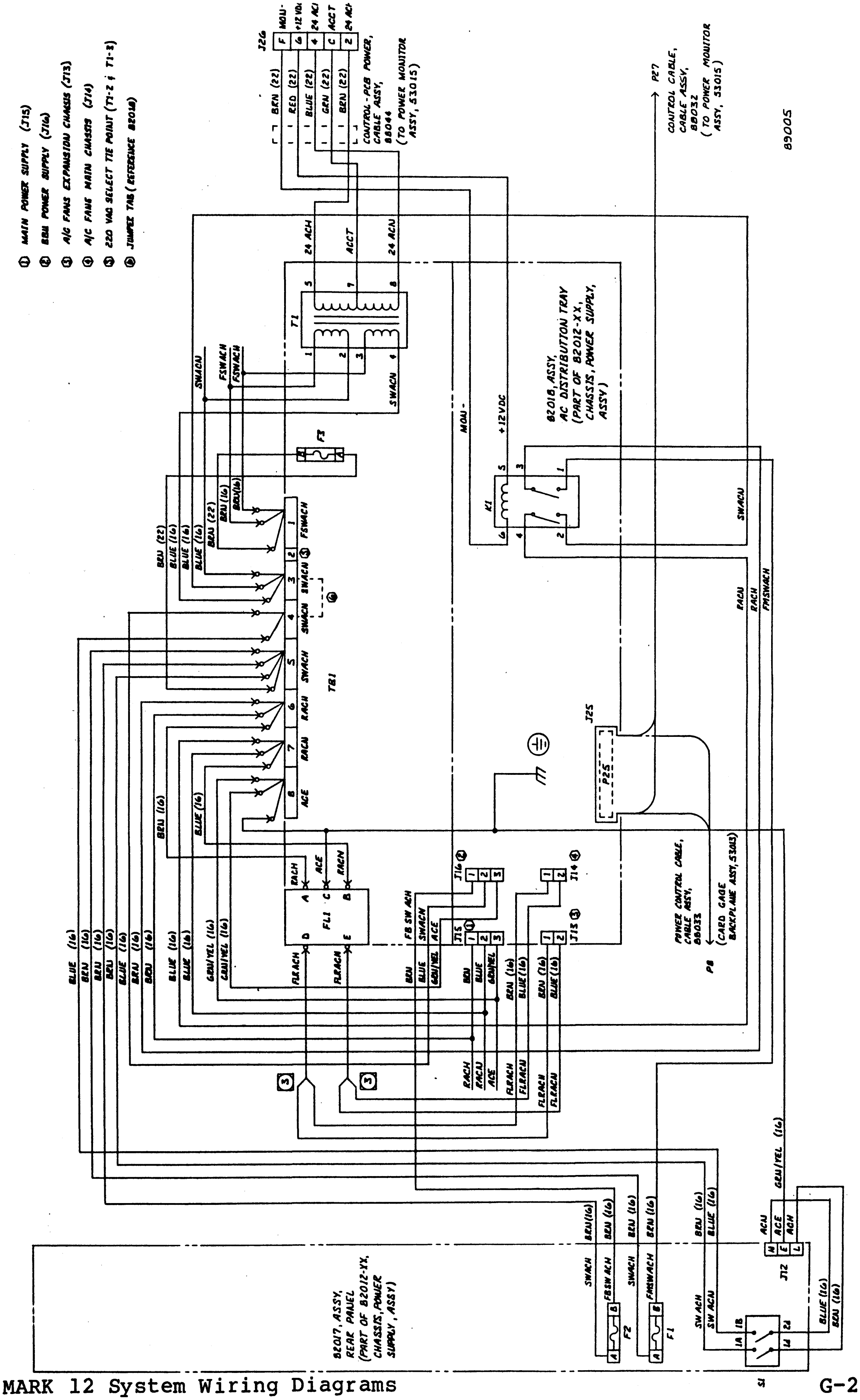
```
- PAGE 4 -

; SIMPLE, SINGLE-LEVEL INTERRUPT VECTORING

              0       .LOC   0
       0      0       0                 ;INTERRUPTED P.C. WILL BE STORED HERE
       1   2000       INTSV             ;POINTER TO INTERRUPT SERVICE

           2000       .LOC   2000
    2000  40422 INTSV:STA    0,INTS0    ;\
    2001  44422       STA    1,INTS1    ; \
    2002  50422       STA    2,INTS2    ;  \ SAVE ACCUMULATORS
    2003  54422       STA    3,INTS3    ;  / AND CARRY
    2004 101100       MOVL   0,0        ; /
    2005  40421       STA    0,INTSC    ;/
    2006  30421       LDA    2,.INTV    ;POINTER TO INTERRUPT VECTOR TABLE
    2007  61477       INTA   0          ;GET CODE OF INTERRUPTING DEVICE
    2010 113000       ADD    0,2        ;COMPUTE DESIRED INTERRUPT VECTOR
    2011   7000       JSR    @0,2       ;JUMP TO INDICATED SERVICE ROUTINE

    2012  20414 INTSR:LDA    0,INTSC    ;RETURN FROM SERVICE ROUTINE
    2013 101200       MOVR   0,0        ; \
    2014  20406       LDA    0,INTS0    ;  \ RESTORE ACCUMULATORS
    2015  24406       LDA    1,INTS1    ;  / AND CARRY
    2016  30406       LDA    2,INTS2    ; /
    2017  34406       LDA    3,INTS3    ;/
    2020  60177       INTEN             ;RE-ENABLE INTERRUPTS
    2021   2000       JMP    @0         ;RETURN TO INTERRUPTED PROGRAM


    2022      0 INTS0:0                 ;SAVE A0
    2023      0 INTS1:0                 ;SAVE A1
    2024      0 INTS2:0                 ;SAVE A2
    2025      0 INTS3:0                 ;SAVE A3
    2026      0 INTSC:0                 ;SAVE CARRY
    2027   2100 .INTV:INTVT             ;POINTER TO INTERRUPT VECTOR TABLE


           2100       .LOC   2100       ;INTERRUPT VECTOR TABLE
U   2100   2100 INTVT:IS00   ;INTERRUPT SERVICE ADDRESS FOR DEVICE 00
U   2101   2101       IS01   ;FOR DEVICE 01
                      ;ETC.

; NOTE: MANY OF THE INTERRUPT SERVICE VECTORS MAY POINT TO THE SAME
;       DEFAULT SERVICE ROUTINE
```

Figure E-4. Master Terminal I/O Programming Examples (4 of 4)

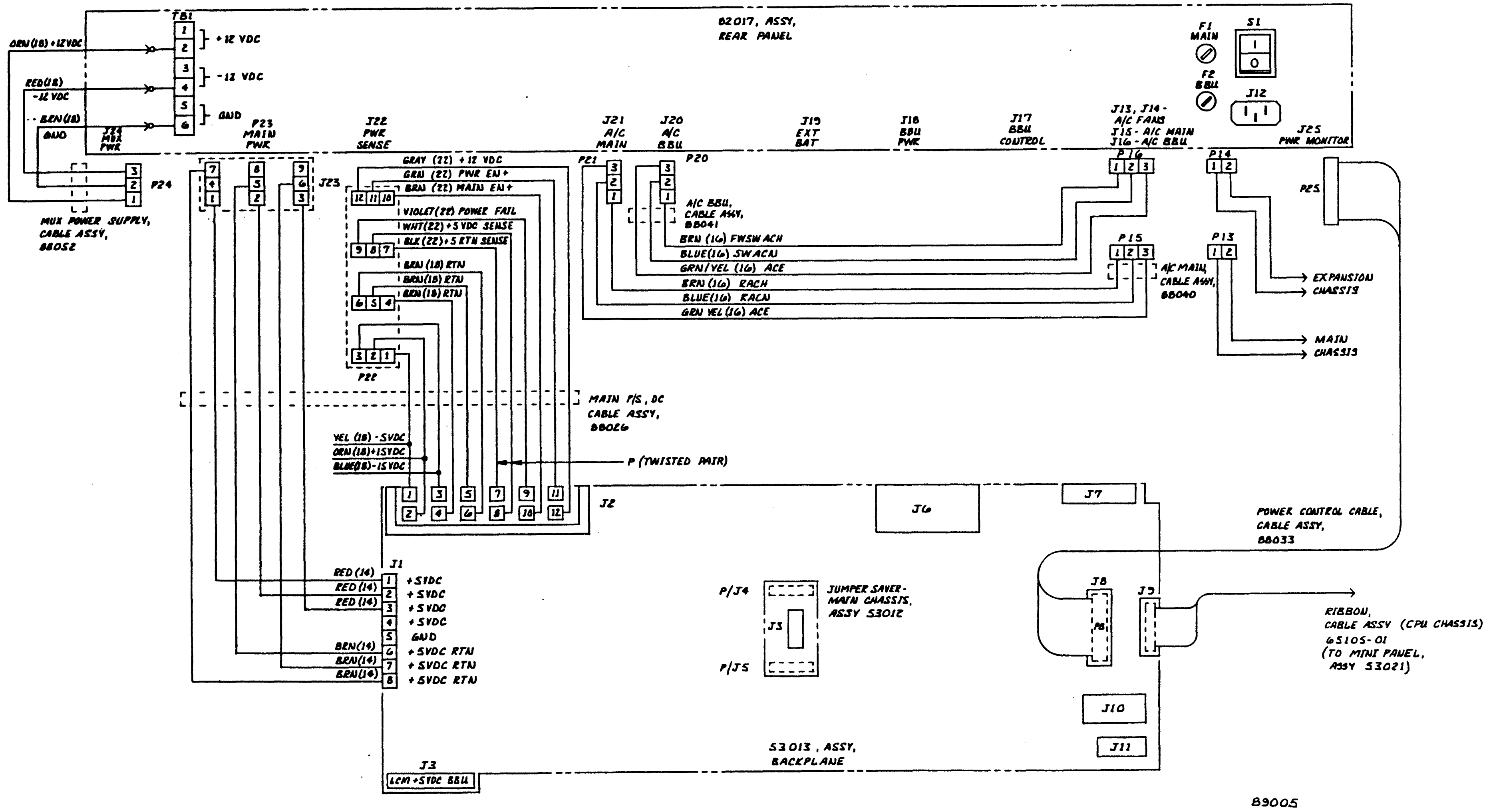


MARK 12 Rack-Mounting Illustration

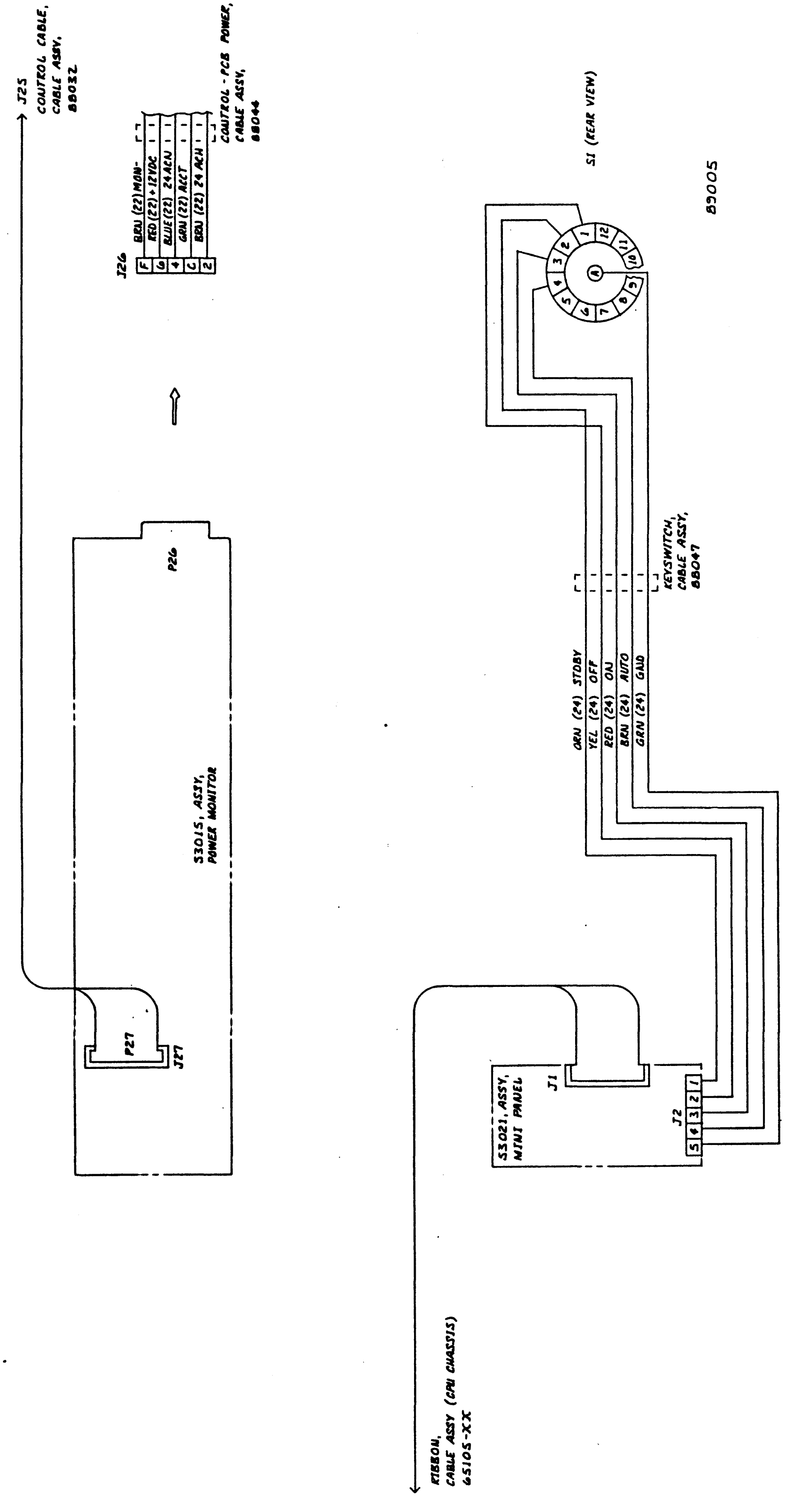

J21
AC CABLE, MAIN POWER SUPPLY, 88031
ACE
ACN
ACH
GRN/YEL (16)
BLUE (16)
BRN (16)
J22
AUX VOLTAGE CABLE, CABLE ASSY, 88030
BLK (22)
BRN (18)
YEL (18)
WHT (22)
ORN (18)
BRN (18)
VIO (22)
BRN (18)
BLUE (18)
+5VDC RTN SENSE
COM RTN
-5VDC
+5VDC SENSE
+15 VDC
COM RTN
+12 VDC
PWR FAIL-
COM RTN
-15 VDC
P23
BRN (14)
RED (14)
BRN (14)
RED (14)
BRN (14)
RED (14)
+5V POWER SUPPLY, CABLE ASSY, 88029
+5VDC RTN
+5VDC
+5VDC RTN
+5VDC
+5VDC RTN
+5VDC
J24
+12 VDC
GND
-12 VDC
89005
TB3
+15 VDC
COM RTN
COM RTN
-15 VDC
COM RTN
-5VDC
+5 VDC RTN SENSE
+5 VDC SENSE
PWR FAIL-
S13550, POWER SUPPLY
TB2
+5VDC SENSE
+5VDC
+5VDC RTN
+5VDC RTN SENSE
PFD
AC SELECT
110 VAC
220 VAC
TB1
ACH
ACN
ACE
YEL (22)
ORN (22)
RED (22)
BRN (22)
COM RTN
-15 VDC
+15 VDC
COM RTN
B2
RED
BLK
B1
RED
BLK
BLUE (18)
BRN (18)
ORN (18)
GRAY (18)
BRN (18)
VIOLET (18)
± 12 REGULATOR, CABLE ASSY, 88028
GRAY (18)
+12 VDC
-15 VDC
COM RTN
+15 VDC
+12 VDC
GND
-12 VDC
1A
053014, PCB ASSY, RS 232 ±12V REGULATOR
82014, ASSY, POWER SUPPLY, (PART OF 82012-XX CHASSIS POWER SUPPLY, ASSY)

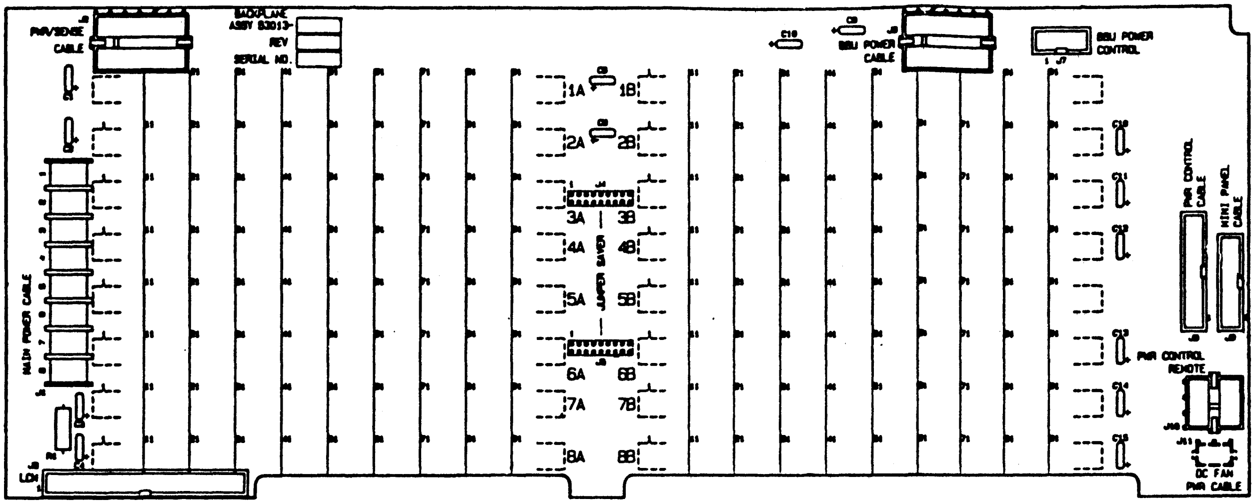


1 MAIN POWER SUPPLY (J15)
2 BBU POWER SUPPLY (J16)
3 A/C FANS EXPANSION CHASSIS (J13)
4 A/C FANS MAIN CHASSIS (J14)
5 220 VAC SELECT TIE POINT (T1-2 & T1-3)
6 JUMPER TAB (REFERENCE B2018)
J26
MOV-
+12 VDC
24 ACN
ACCT
24 ACN
BRN (22)
RED (22)
BLUE (22)
GRN (22)
BRN (22)
CONTROL-PCB POWER, CABLE ASSY, 88044 (TO POWER MONITOR ASSY, 53015)
P27
CONTROL CABLE, CABLE ASSY, 88032 (TO POWER MONITOR ASSY, 53015)
89005
T1
SWACN
FSWACH
B2018, ASSY, AC DISTRIBUTION TRAY (PART OF B2012-XX, CHASSIS, POWER SUPPLY, ASSY)
F3
K1
TB1
FSWACH
SWACH
SWACN
RACH
RACN
ACE
RACN
RACH
FMSWACH
J25
P25
POWER CONTROL CABLE, CABLE ASSY, 88033
P8
(CARD GAGE BACKPLANE ASSY, 53013)
FL1
FLRACH
FLRACN
FB SW ACH
SWACN
J16
J15
J14
J13
BLUE (16)
BRN (16)
GRN/YEL (16)
B2017, ASSY, REAR PANEL (PART OF B2012-XX, CHASSIS, POWER SUPPLY, ASSY)
FBSWACH
FMSWACH
F1
F2
J12
SW ACH
SW ACN
S1


ORN(18) +12VDC
RED(18)
-12 VDC
BRN(18)
GND
TB1
+12 VDC
-12 VDC
GND
J24
MUX
PWR
P24
MUX POWER SUPPLY,
CABLE ASSY,
88052
82017, ASSY,
REAR PANEL
F1
MAIN
F2
BBU
S1
J12
P23
MAIN
PWR
J23
J22
PWR
SENSE
J21
A/C
MAIN
J20
A/C
BBU
J19
EXT
BAT
J18
BBU
PWR
J17
BBU
CONTROL
J13, J14 -
A/C FANS
J15 - A/C MAIN
J16 - A/C BBU
J25
PWR MONITOR
P21
P20
P16
P14
P25
P15
P13
GRAY (22) +12 VDC
GRN (22) PWR EN+
BRN (22) MAIN EN+
VIOLET(22) POWER FAIL
WHT(22) +5 VDC SENSE
BLK (22) +5 RTN SENSE
BRN (18) RTN
BRN(18) RTN
BRN(18) RTN
P22
A/C BBU,
CABLE ASSY,
88041
BRN (16) FWSW ACH
BLUE(16) SWACN
GRN/YEL (16) ACE
BRN (16) RACH
BLUE(16) RACN
GRN YEL (16) ACE
A/C MAIN,
CABLE ASSY,
88040
EXPANSION
CHASSIS
MAIN
CHASSIS
MAIN P/S, DC
CABLE ASSY,
88026
YEL (18) -5VDC
ORN(18)+15VDC
BLUE(18)-15VDC
P (TWISTED PAIR)
J2
J6
J7
POWER CONTROL CABLE,
CABLE ASSY,
88033
J1
RED (14)
RED (14)
RED (14)
BRN(14)
BRN(14)
BRN(14)
+5VDC
+5VDC
+5VDC
+5VDC
GND
+5VDC RTN
+5VDC RTN
+5VDC RTN
P/J4
P/J5
J3
JUMPER SAVER -
MAIN CHASSIS,
ASSY 53012
J8
P8
J9
RIBBON,
CABLE ASSY (CPU CHASSIS)
65105-01
(TO MINI PANEL,
ASSY 53021)
J10
J11
53013, ASSY,
BACKPLANE
J3
LCM +5VDC BBU
89005

REAR VIEW


J25
CONTROL CABLE,
CABLE ASSY,
88032
J26
F
G
4
C
2
BRN (22) MON-
RED (22) +12VDC
BLUE (22) 24ACN
GRN (22) ACCT
BRN (22) 24 ACH
CONTROL - PCB POWER,
CABLE ASSY,
88044
S1 (REAR VIEW)
89005
1
2
3
4
5
6
7
8
9
10
11
12
A
P26
53015, ASSY,
POWER MONITOR
P27
J27
KEYSWITCH,
CABLE ASSY,
88047
ORN (24) STDBY
YEL (24) OFF
RED (24) ON
BRN (24) AUTO
GRN (24) GND
J1
J2
5
4
3
2
1
53021, ASSY,
MINI PANEL
RIBBON,
CABLE ASSY (CPU CHASSIS)
65105-XX



53013

MARK 12 Backplane Board Illustration

CPU ONLY

1A				1B			
TOP		BOTTOM		TOP		BOTTOM	
PIN	SIGNAL	PIN	SIGNAL	PIN	SIGNAL	PIN	SIGNAL
1	GND	2	GND	1	GND	2	GND
3	+5V	4	+5V	3	+5V	4	+5V
5	+5BU	6	-5V	5	V31-	6	
7	PWRGON-	8	V0-	7	VB1-	8	WRITE-
9	-5BU	10	+15V	9	LWORD-	10	VB2-
11	PWRF-	12		11		12	AS-
13	V2-	14	V3-	13		14	DS-
15	V4-	16	V5-	15		16	VB3-
17	V6-	18	V7-	17	DCHMO-	18	SYSCLK-
19	V8-	20	V9-	19		20	VB4-
21	V10-	22	V11-	21		22	VB5-
23	V12-	24	V13-	23		24	VB6-
25	V14-	26	V15-	25		26	BBUSY-
27	V16-	28	V17-	27		28	BERR-
29	V18-	30	V19-	29	INTR-	30	BERC-
31	V20-	32	V1-	31		32	DTACK-
33	GND	34	GND	33	DCHO+	34	
35	V21-	36	V22-	35	DCHR-	36	
37	V23-	38	MSKO-	37	DCHI+	38	
39	V24-	40	INTA+	39		40	
41	V25-	42	DATIB+	41	RQENB-	42	BACKUP+
43	V26-	44	DATIA+	43	+5BU	44	+5BU
45	V27-	46	DS3-	45		46	+15V
47		48	DATOC+	47		48	
49		50	CLR+	49		50	GND
51	V28-	52	STRT+	51		52	
53	V29-	54	DATIC+	53		54	
55	V30-	56	DATOB+	55	DATA7-	56	DATA14-
57		58	DATOA+	57	DATA5-	58	DATA11-
59		60	DCHA-	59	DATA12-	60	DATA8-
61		62	DS4-	61	DATA4-	62	DATA0-
63		64	DS5-	63	DATA9-	64	DATA13-
65		66	DS2-	65	DATA1-	66	DATA15-
67		68	DS1-	67		68	
69		70	IORST+	69		70	
71		72	DS0-	71		72	
73		74	IOPLS+	73	DATA3-	74	
75		76		75	DATA10-	76	
77		78		77		78	
79		80	SELD-	79		80	
81		82	SELB-	81	-5V	82	DATA2-
83		84	PEL-	83		84	+15V
85		86	RUNL-	85		86	
87		88	CL-	87		88	
89		90	CNT-	89		90	
91		92	STP-	91	-15V	92	GND
93		94		93	-15V	94	+12BU
95		96		95	DATA6-	96	+5BU
97	+5V	98	+5V	97	+5V	98	+5V
99	GND	100	GND	99	GND	100	GND

MEMORY / I/O

2A				2B			
TOP		BOTTOM		TOP		BOTTOM	
PIN	SIGNAL	PIN	SIGNAL	PIN	SIGNAL	PIN	SIGNAL
1	GND	2	GND	1	GND	2	GND
3	+5V	4	+5V	3	+5V	4	+5V
5	+5BU	6	-5V	5	V31-	6	
7	PWRGON-	8	V0-	7	VB1-	8	WRITE-
9		10	+15V	9	LWORD-	10	VB2-
11	PWRF-	12		11		12	AS-
13	V2-	14	V3-	13		14	DS-
15	V4-	16	V5-	15		16	VB3-
17	V6-	18	V7-	17	DCHMO-	18	SYSCLK-
19	V8-	20	V9-	19		20	VB4-
21	V10-	22	V11-	21		22	VB5-
23	V12-	24	V13-	23		24	VB6-
25	V14-	26	V15-	25		26	BBUSY-
27	V16-	28	V17-	27		28	BERR-
29	V18-	30	V19-	29	INTR2-	30	BERC-
31	V20-	32	V1-	31		32	DTACK-
33	GND	34	GND	33	DCHO+	34	
35	V21-	36	V22-	35	DCHR2-	36	
37	V23-	38	MSKO-	37	DCHI+	38	
39	V24-	40	INTA+	39		40	
41	V25-	42	DATIB+	41	RQENB-	42	BACKUP+
43	V26-	44	DATIA+	43	+5BU	44	+5BU
45	V27-	46	DS3-	45		46	+15V
47		48	DATOC+	47		48	
49		50	CLR+	49		50	GND
51	V28-	52	STRT+	51		52	
53	V29-	54	DATIC+	53		54	
55	V30-	56	DATOB+	55	DATA7-	56	DATA14-
57		58	DATOA+	57	DATA5-	58	DATA11-
59		60	DCHA-	59	DATA12-	60	DATA8-
61		62	DS4-	61	DATA4-	62	DATA0-
63		64	DS5-	63	DATA9-	64	DATA13-
65		66	DS2-	65	DATA1-	66	DATA15-
67		68	DS1-	67		68	
69		70	IORST+	69		70	
71		72	DS0-	71		72	
73		74	IOPLS+	73	DATA3-	74	
75		76		75	DATA10-	76	
77		78		77		78	
79		80	SELD-	79		80	
81		82	SELB-	81	-5V	82	DATA2-
83		84		83		84	+15V
85		86		85		86	
87		88		87		88	
89		90		89		90	
91		92		91	-15V	92	GND
93		94	DCHPIN2-	93	-15V	94	
95		96	INTPIN2-	95	DATA6-	96	+5BU
97	+5V	98	+5V	97	+5V	98	+5V
99	GND	100	GND	99	GND	100	GND

I/O ONLY

6A				6B			
TOP		BOTTOM		TOP		BOTTOM	
PIN	SIGNAL	PIN	SIGNAL	PIN	SIGNAL	PIN	SIGNAL
1	GND	2	GND	1	GND	2	GND
3	+5V	4	+5V	3	+5V	4	+5V
5		6	-5V	5		6	
7		8		7		8	
9		10	+15V	9		10	
11		12		11		12	
13		14		13		14	
15		16		15		16	
17		18		17	DCHMO-	18	
19		20		19		20	
21		22		21		22	
23		24		23		24	
25		26		25		26	
27		28		27		28	
29		30		29	INTR6-	30	
31		32		31		32	
33	GND	34	GND	33	DCHO+	34	
35		36		35	DCHR6-	36	
37		38	MSKO-	37	DCHI+	38	
39		40	INTA+	39		40	
41		42	DATIB+	41	RQENB-	42	
43		44	DATIA+	43		44	
45		46	DS3-	45		46	+15V
47		48	DATOC+	47		48	
49		50	CLR+	49		50	GND
51		52	STRT+	51		52	
53		54	DATIC+	53		54	
55		56	DATOB+	55	DATA7-	56	DATA14-
57		58	DATOA+	57	DATA5-	58	DATA11-
59		60	DCHA-	59	DATA12-	60	DATA8-
61		62	DS4-	61	DATA4-	62	DATA0-
63		64	DS5-	63	DATA9-	64	DATA13-
65		66	DS2-	65	DATA1-	66	DATA15-
67		68	DS1-	67		68	
69		70	IORST+	69		70	
71		72	DS0-	71		72	
73		74	IOPLS+	73	DATA3-	74	
75		76		75	DATA10-	76	
77		78		77		78	
79		80	SELD-	79		80	
81		82	SELB-	81	-5V	82	DATA2-
83		84		83		84	+15V
85		86		85		86	
87		88		87		88	
89		90		89		90	
91		92		91	-15V	92	GND
93		94	DCHPIN6-	93	-15V	94	
95		96	INTPIN6-	95	DATA6-	96	
97	+5V	98	+5V	97	+5V	98	+5V
99	GND	100	GND	99	GND	100	GND

MAIN POWER CABLE

J1	
PIN	SIGNAL
1	+5V
2	+5V
3	+5V
4	+5V
5	GND
6	GND
7	GND
8	GND

PWR/SENSE CABLE

J2	
PIN	SIGNAL
1	-5V
2	+15V
3	-15V
4	GND
5	GND
6	GND
7	GND
8	+5V
9	PFAIL-
10	MAINEN+
11	PWREN+
12	+12V

LCM +5BU PWR CABLE

J3	
PIN	SIGNAL
1	GND
2	GND
3	+5BU
4	+5BU
5	GND
6	GND
7	+5BU
8	+5BU
9	GND
10	GND
11	+5BU
12	+5BU
13	GND
14	GND
15	+5BU
16	+5BU
17	GND
18	GND
19	+5BU
20	+5BU
21	GND
22	GND
23	
24	
25	
26	
27	
28	
29	GND
30	GND
31	+5BU
32	+5BU
33	GND
34	GND
35	+5BU
36	+5BU
37	GND
38	GND
39	+5BU
40	+5BU
41	GND
42	GND
43	+5BU
44	+5BU
45	GND
46	GND
47	+5BU
48	+5BU
49	GND
50	GND

JUMPER SAVER

J4	
PIN	SIGNAL
1	DCHR-
2	DCHR5-
3	DCHR4-
4	DCHR3-
5	DCHR2-
6	DCHR6-
7	DCHR7-
8	DCHR8-
9	+5V
10	GND
11	
12	DCHPIN6-
13	DCHPIN7-
14	DCHPIN8-
15	DCHPIN2-
16	DCHPIN3-
17	DCHPIN4-
18	DCHPIN5-

JUMPER SAVER

J5	
PIN	SIGNAL
1	INTR-
2	INTR5-
3	INTR4-
4	INTR3-
5	INTR2-
6	INTR6-
7	INTR7-
8	INTR8-
9	+5V
10	GND
11	
12	INTPIN6-
13	INTPIN7-
14	INTPIN8-
15	INTPIN2-
16	INTPIN3-
17	INTPIN4-
18	INTPIN5-

BBU POWER CABLE

J6	
PIN	SIGNAL
1	+5BU
2	+5BU
3	+5V
4	+12BU
5	+12V
6	BACKUP+
7	GND
8	GND
9	GND
10	-5BU
11	-5V
12	GND

BBU POWER CONTROL

J7	
PIN	SIGNAL
1	BTRYOK-
2	PFBBU-
3	
4	PWREN+
5	GND
6	GND
7	BTROFF-
8	SPARE1
9	+5BU
10	GND

PWR CONTROL CABLE

J8	
PIN	SIGNAL
1	GND
2	GND
3	+5BU
4	+12BU
5	PWREN+
6	+15V
7	+5AUX
8	-5V
9	PFAUX-
10	-15V
11	GND
12	BACKUP+
13	GND
14	PWRF-
15	PWRGON-
16	-5BU
17	+5V
18	MAINEN+
19	PWRL-
20	PFBBU-
21	+5V
22	PFAIL-
23	+5MPL
24	+5MPL
25	GND
26	GND

MINI PANEL CABLE

J9	
PIN	SIGNAL
1	+5MPL
2	RUNL-
3	PWREN+
4	PEL-
5	+5MPL
6	CNT-
7	+5BU
8	CL-
9	+5BU
10	STP-
11	GND
12	BTRYOK-
13	GND
14	PWRGON-
15	GND
16	SPARE1
17	GND
18	PWRL-
19	GND
20	BTROFF-

PWR CONTROL REMOTE

J10	
PIN	SIGNAL
1	PWREN+
2	SPARE1
3	GND
4	BTROFF-
5	+5AUX
6	PFAUX-

FAN CABLE

J11	
PIN	SIGNAL
1	GND
2	+15V

POINT 4
DATA CORPORATION

15442 Del Amo Avenue
Tustin, CA 92680
(714) 259-0777