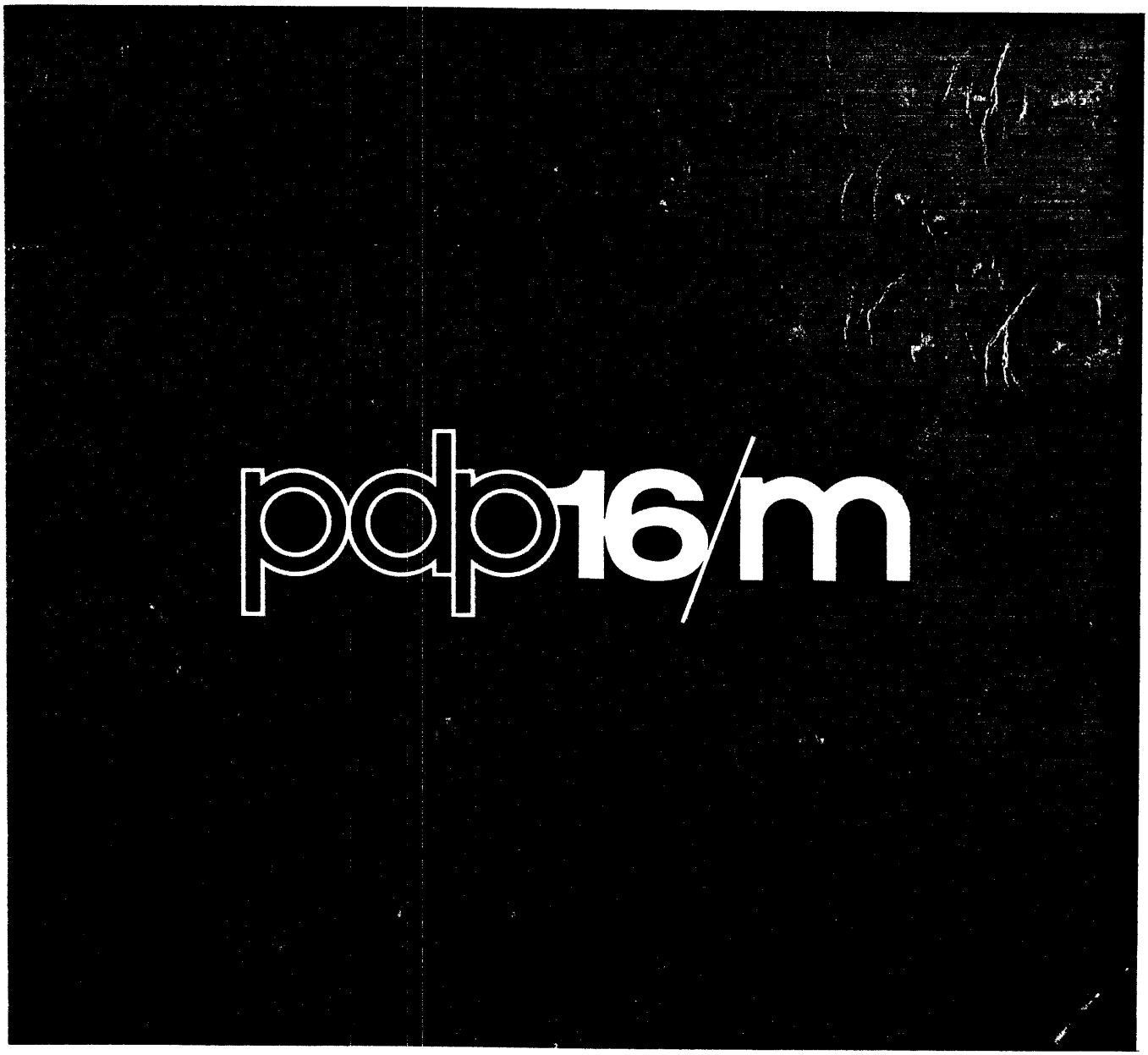
Digital Equipment Corporation
Maynard, Massachusetts

digital

USER'S GUIDE



DEC-16-IMUGA-A-D

PDP16-M
USER'S GUIDE

digital equipment corporation · maynard, massachusetts

1st Edition March 1973

CONTENTS

CONTENTS (Cont)

ILLUSTRATIONS

ILLUSTRATIONS (Cont)

TABLES

FOREWORD

All information necessary for interfacing, programming, and debugging a PDP16-M is contained in this manual. The manual contains six chapters and eight appendices.

Chapter 1 provides an overview of the PDP16-M. Applications, hardware features, and software are summarized.

Chapter 2 contains a brief description of the hardware. Functional and physical descriptions, specifications, and configuration data are included.

Chapter 3 contains details for interfacing the PDP16-M with the outside world. All input and output signals, with respective loading information, are identified. Specific details are included for interfacing with serial data communication devices and low-speed PDP-11 peripheral devices.

Chapter 4 covers the basic instruction set, assembler directives, and the instructions implemented through options. The chapter details program formats and conventions, using many examples.

Chapter 5 describes how to prepare and assemble a PDP16-M control program. The RIM and BIN Loaders, the Symbolic Editor, and the PAL16 Assembler are described in detail.

All features of the utility option for the PDP16-M are discussed in Chapter 6. Detailed interface information is included in this chapter to allow the user to interconnect his utility system.

Appendix A contains a summary of low speed PDP-11 peripheral devices. Their status control and data word formats and addresses are specified. Option DA16-F permits the user to utilize low speed PDP-11 peripheral devices in a PDP16-M system.

Appendix B lists the octal codes for the ASCII and EBCDIC character sets.

Appendix C contains various code conversion tables.

Appendix D lists the complete instruction set of the PDP16-M. The list is ordered by classification.

Appendix E lists the complete instruction set of the PDP16-M. The list is ordered by octal machine code.

Appendix F is a listing of the definition tape used to initialize the PDP-16 Assembler.

Appendix G lists all PDP16-M and PDP-11 I/O signals. The lists are in alphabetic order.

Appendix H is a summary describing options available to the small and large users.

CHAPTER 1
OVERVIEW

1.1 INTRODUCTION

The PDP16-M resembles both a hard-wired controller and a general purpose minicomputer. It is similar to a hard-wired controller because the program is stored in a memory that is impervious to noise interference and process sensors/controls can be easily interfaced. The PDP16-M also resembles a general purpose minicomputer because it features arithmetic and data memory capability in addition to the logic and I/O capability typical of a hard-wired controller. In general, applications tend to be where a conventional minicomputer may be too costly and a hard-wired design too time consuming. Some typical application areas for the PDP16-M are:

- Machinery control, alarm scanning, data collection, monitoring, and remote telemetry in the *industrial* field.
- Fourier Analysis, auto and cross correlation, and waveform synthesis in the *laboratory*.
- Input terminals for automated stock control, warehousing, and inventory control in the *commercial* market.
- Data format conversion, code conversion, complex arithmetic, and intelligent front-end processor to larger computers in the *data processing* field.

1.1.1 Functional Programming

The PDP16-M is a function-oriented minicomputer, with a simple design that novice computer users will quickly grasp and experienced computer users will fully appreciate. It is termed a "function-oriented" computer because it responds to a simple set of function reference instructions. The functions are specified by plain language statements that are similar to those used in BASIC and FORTRAN programming languages. This makes the task of programming the PDP16-M much easier than programming a conventional minicomputer. The functions implemented in the PDP16-M include arithmetic, Boolean logic, data memory transfers, and I/O operations.

1.1.2 Multiple Memory Features

The PDP16-M features two segregated memories: one for control instruction storage and another for data storage. This feature eliminates the need for direct memory reference instructions and the programming complexities associated with memory reference instructions that are inherent with conventional minicomputers.

The PDP16-M also uses an inexpensive, state-of-the-art, programmable read-only memory (PROM) for program and permanent data storage. PROMs are not susceptible to environmental noise, and their contents cannot be changed inadvertently. Thus, they provide an ideal program and data storage medium for dedicated control applications in a noisy environment. The use of PROMs in the PDP16-M also eliminates the need for expensive paper tape I/O equipment or a computer console, required by computers that employ read/write program storage.

1.2 MACHINE DESCRIPTION AND FEATURES

Figure 1-1, a simplified block diagram of the PDP16-M, illustrates some of the main features of the PDP16-M: *segregated memory* for instructions and data, *arithmetic and logical capabilities* offered by the ALU (arithmetic and logic unit), and *I/O capabilities*. The ALU, Data Memory, and I/O channels are all interconnected via a common data highway called the Register Transfer Module (RTM) data bus. Notice that the instruction memory is not connected to the bus. *Function reference instructions* are assigned 192 instruction codes for operating on data in the ALU and for transferring data between the ALU, Data Memory, and I/O channels. In addition, 64 instruction codes are assigned to *unconditional* and *conditional branch instructions* and *subroutine* CALL and EXIT instructions. These instructions operate only on the program control sequencer – not the RTM bus. The application program is stored in a PROM designated the instruction (control) memory. The program is written for a specific application by the user, using the simple PDP16-M instruction set. All Boolean, arithmetic, and data transfer functions found in other computers are available in the PDP16-M. Functions like AND, OR, exclusive-OR, complement, shift, add, subtract, binary multiply, binary divide, move data, load data memory, read from data memory, interrogate I/O devices, etc., can all be handled with ease in the PDP16-M. After the application program is assembled and debugged, it is loaded into the PROM. The PROM can then be installed in its preassigned slot on the PDP16-M logic assembly. After starting the program, the instructions are fetched and executed by the program control sequencer one at a time without operator intervention. The instruction memory can be expanded from its basic 256 words to 1024 words in 256-word increments, permitting the user to implement only the program memory he needs.

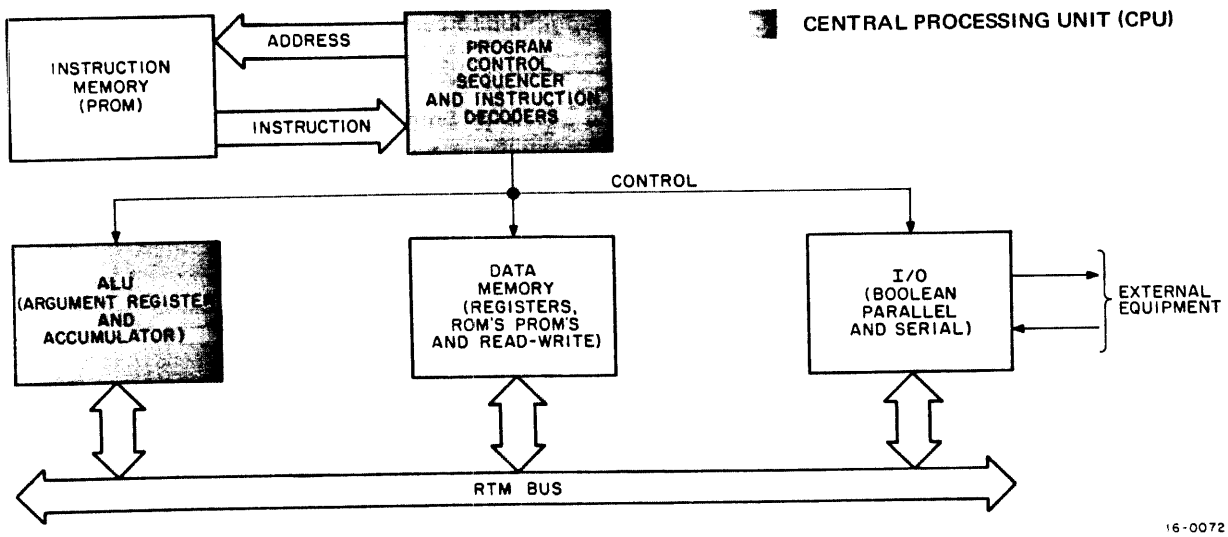


Figure 1-1 PDP16-M Simplified Block Diagram

1.2.1 Programmable Read-Only Memory (PROM)

The PROM is an electrically programmable read-only memory. It contains 256 addressable 8-bit storage locations. The PROM is housed in a 24-pin dual-in-line ceramic package with a quartz window that exposes the MOS chip. The chip is a silocon gate matrix with switchable links. These links can be reset by exposing the chip to ultraviolet light. After the links are reset, the PROM can be reprogrammed. The PROM may be reprogrammed a minimum of 100 times without affecting reliability.

1.2.2 Program Control Sequencer (PCS)

The PCS fetches and executes the instructions one at a time. It generates control signals in response to the instruction code to operate on data in the ALU and to transfer data between the ALU, Data Memory, and I/O channels. The branch and subroutine instructions are executed by internal logic of the PCS.

1.2.3 Register Transfer Module (RTM)

The functional elements that are connected to the RTM bus are called Register Transfer Modules because data can be transferred between the modules via the RTM bus simply by asserting the appropriate control inputs. Since a functional element, when not accessing the bus, presents essentially no load to the RTM bus, the system can be expanded to accommodate virtually an unlimited number of elements. The standard and optional functional elements that are available for the PDP16-M include:

a. Arithmetic and Logic Unit

b. Data Memory

- Scratchpad Registers
- Constant ROMs
- Data PROMs
- Data Read-Write Memory

c. I/O Channels

- Boolean I/O (TTL)
- Serial I/O (TTY or TTL)
- Parallel I/O (16-bit)
- PDP-11 Peripheral I/O

1.2.4 Modularity

All components (modules and functional elements) of the PDP16-M are housed on a single prewired logic assembly. The standard and optional components can be implemented simply by inserting the components in their preassigned slot. Except for I/O interfacing, no other wiring is required.

1.3 SOFTWARE

An extensive software package is available for the PDP16-M. Software is available for:

a. Program development, including preparation and assembly.

b. Program debugging and loading.

c. Diagnostics for basic machine and all available options.

Program development, debugging, and loading must be done with the aid of a PDP-8/E. For those customers who do not have a PDP-8/E or do not wish to purchase one, Digital Equipment Corporation offers to load the Instruction and Data Memory PROMs at a minimal cost. In this case, the user need only provide DEC with the source tape in ASCII format. Refer to Appendix H.

The diagnostic programs are preloaded on PROMs. To run the diagnostics, the appropriate PROMs are inserted in the instruction memory slots and the program is started by depressing the START switch on the console. If a failure occurs, the diagnostic program will halt. Then, the diagnostic listing and optional maintenance modules may be used to quickly isolate the malfunction. The maintenance modules feature instruction and data readout displays, and single step and breakpoint functions to aid in isolating the malfunction.

CHAPTER 2
A VIEW OF THE HARDWARE

The *PDP16-M Maintenance Manual* should be consulted by readers who are interested in specific hardware details.

2.1 FUNCTIONAL DESCRIPTION

The PDP16-M has an asychronous control (instruction) memory bus and an asychronous register transfer bus (Figure 2-1). Control PROMs connect to the control memory bus and all register transfer modules connect to the register transfer bus. All data transfers on the control memory bus and register transfer bus are controlled by the CPU. The CPU contains the instruction decoder, state generator, program counter, subroutine stack, arithmetic and logic unit, LINK, A register, B register, and TR register (Figure 2-2).

The basic PDP16-M is equipped with one control PROM; three additional PROMs may be added to the memory bus. The PROM has 256 8-bit storage locations.

When the PROM is addressed by the program counter (PC) of the CPU, the word in that PROM location is transferred to the CPU. Both the address and data are transferred via the memory bus. In the CPU, the control word is decoded and executed. Only the LET instruction (register transfer) operates on the register transfer bus. The bus consists of 16 data lines and 5 control lines. All arithmetic, logical, memory, and I/O register transfer operations transpire over the bus. The GOTO and IF instructions simply cause the PC to be incremented to fetch the jump address. The CALL/EXIT instructions operate on the hardware stack to store and retrieve the return address.

All arithmetic and logical operations are implemented by instructions that operate on the ALU via the A, B, and LINK registers. The A register can be used as an accumulator, the B register serves as an argument register, and the LINK register facilitates carry manipulation. The TR register, which is byte and word addressable, can be used for temporary storage or for data manipulation. The register transfer bus provides the data path for transferring the data between these registers and any other register transfer modules that are connected to the bus. (Refer to Appendix D for a list of all legal register transfer instructions.)

The basic PDP16-M is equipped with one 4-word constant ROM and one parallel I/O, both of which are connected to the bus. In addition, the basic PDP16-M contains three flags/Boolean output channels and six Boolean input channels. A wide variety of memory and I/O options can be added to the register transfer bus (Figure 2-1). These options may be added to expand data/constant memory and I/O capabilities. ROMs are available for storing program constants, data, or text information. Read/write MOS memories may be used for accumulating data for I/O, and I/O modules may be added to expand the interfacing capabilities. The standard data options that are available from DEC are:

DATA MEMORY

- Two 16 X 16 (16 words) Scratchpad Registers, MS16-C
- Two Data R/W MOS Memories. Any of the following combinations can be implemented:

MS16-D (16 X 256)	MS16-E (16 X 1024)	Memory Size (words)
1	0	256
2	0	512
0	1	1024
1	1	1280
0	2	2048

- One 16 X 24 Constant ROM, MR16-D
- One 8 X 256 Data PROM (high or low order 8 bits), MR16-E or 16 X 256 Data PROM, MR16-F

INPUT/OUTPUT

- Two 16-bit Parallel I/O Channels, DB16-A
- Two Serial I/O Channels (TTY or TTL), DC16-A
- One PDP-11 Peripheral Interface, DA16-F
- Boolean Input Channels (16), PCS16-D
- Flags/Boolean Output Channels (3), KFL16

The main frame of the PDP16-M is prewired to accept any of the above options simply by inserting the option modules in the preassigned slot.

2.2 PHYSICAL DESCRIPTION

The PDP16-M is packaged in a small table top or rack-mountable cabinet with a self-contained power supply, cooling fans, an air filter, and a simple front panel.

Each slot of the logic assembly has been wired to accept only a specific module type.

CAUTION

Damage may result to the machine if an option is inserted into the wrong slot.

Figure 2-3 and Table 2-1 illustrate the prewired configuration and describe each module of the PDP16-M. The corresponding model number for each module and option is given in Table 2-1. Four additional optional modules are available from DEC. These modules are designed to facilitate maintenance and program debugging. (Refer to Chapter 5 of the *PDP16-M Maintenance Manual*.)

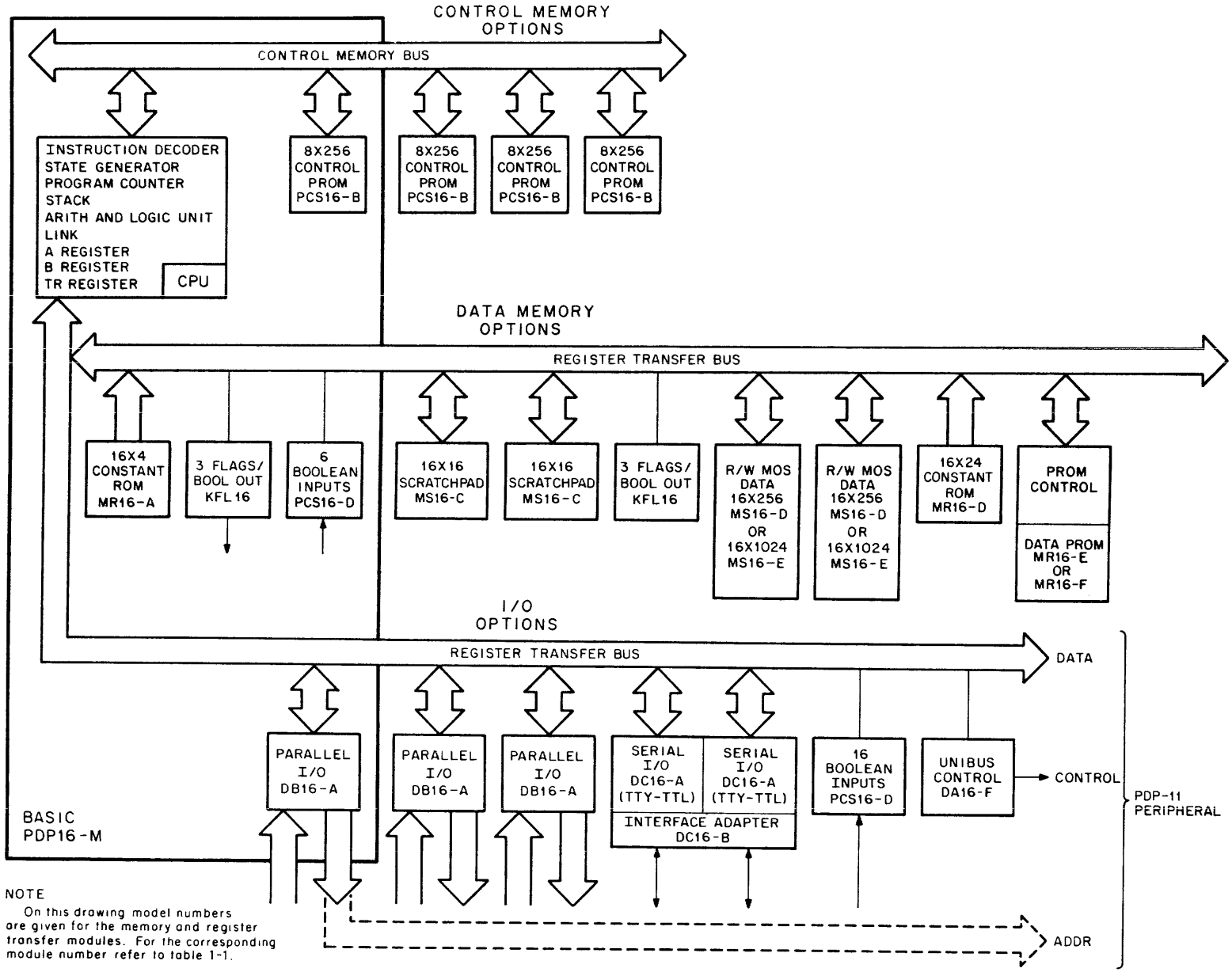


Figure 2-1 PDP16-M Configuration

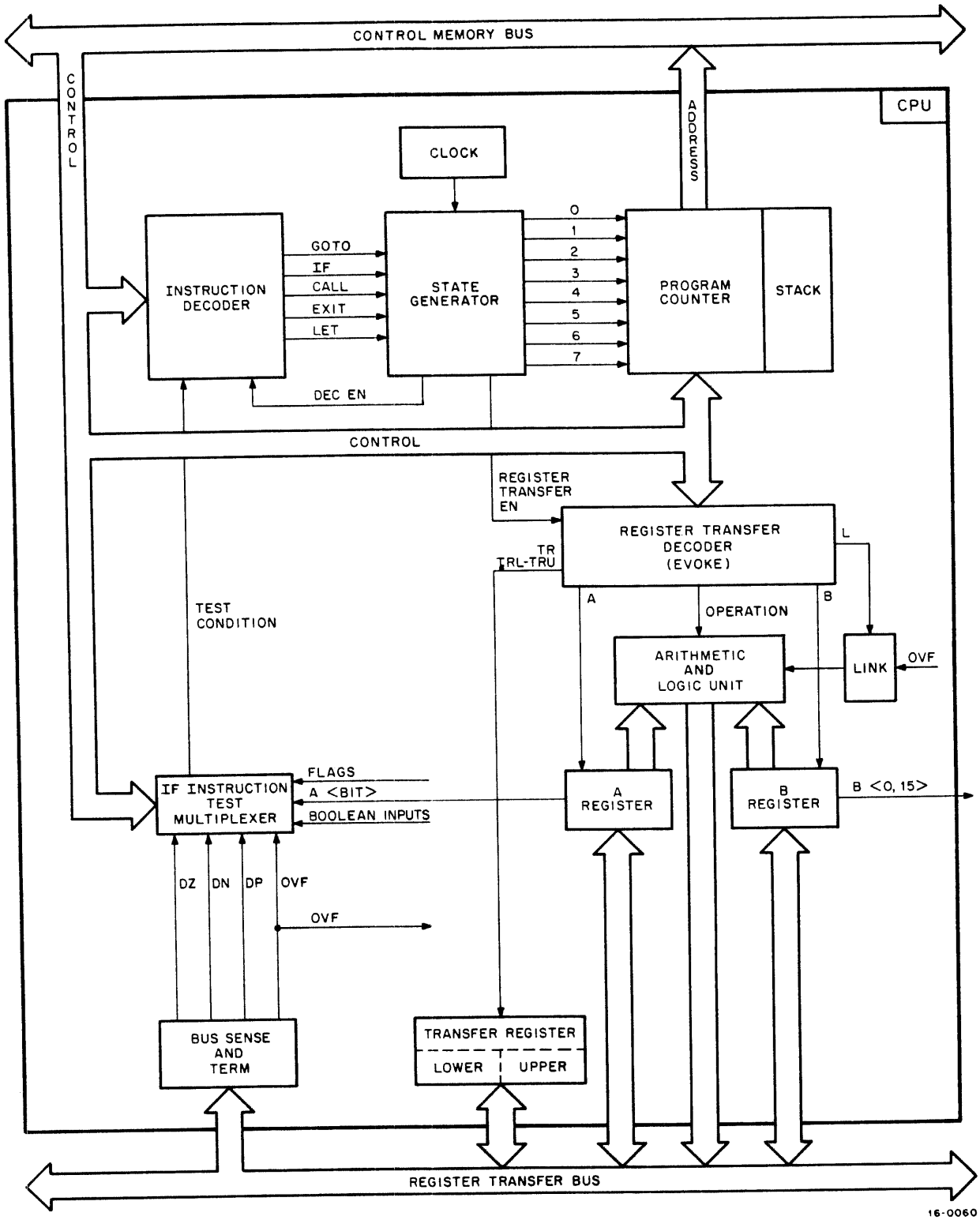


Figure 2-2 CPU Block Diagram

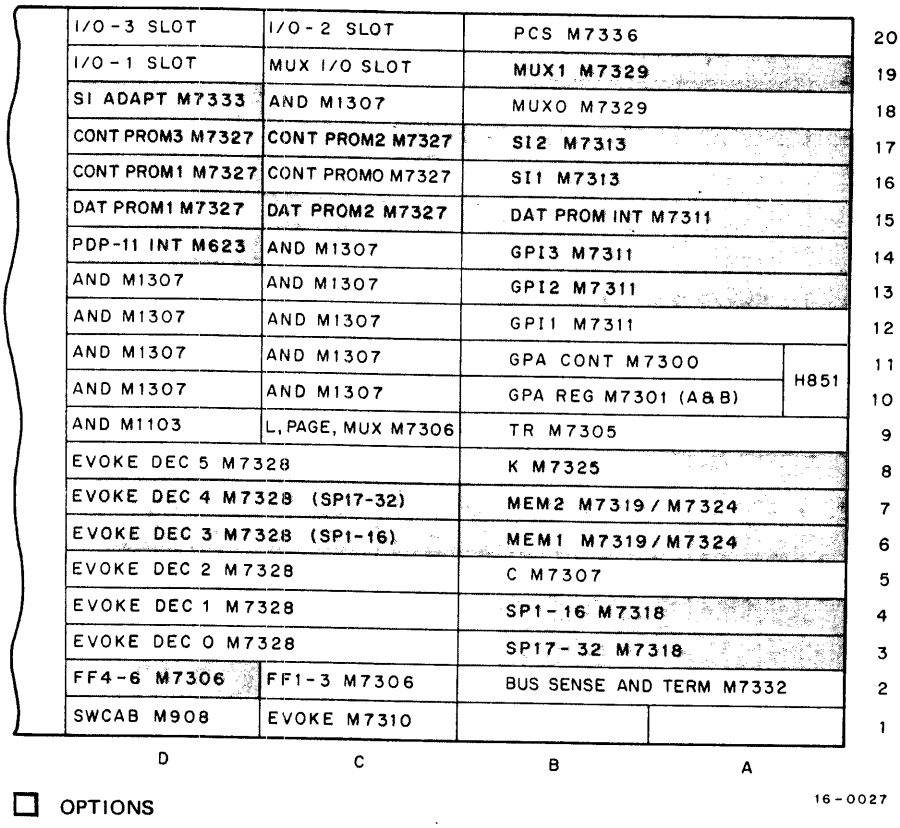


Figure 2-3 Logic Assembly Configuration Diagram

Table 2-1
Module Slot Assignment and Description

Slot	Module No.	Model No.	Configuration	Description
			Row A	
1				Test Slot
2	M7332	KBS16-A	Basic	Timing Control, Data Testing, Bus Terminator
3	M7318	MS16-C	Option	16-Bit Registers SP17 through SP32
4	M7318	MS16-C	Option	16-Bit Registers SP1 through SP16
5	M7307	MR16-A	Basic	4-Word Constant Generator
6	M7319	MS16-D	Option	MEM1 – 256 X 16 R/W MOS Memory
7	M7319	MS16-D	Option	MEM2 – 256 X 16 R/W MOS Memory
8	M7325	MR16-D	Option	24-Word Constant Generator
9	M7305	MS16-A	Basic	16-Bit Register with Byte Control
10	M7301	KAR16	Basic	ALU and Registers A and B
11	M7300	KAC16	Basic	ALU Control Unit
12	M7311	DB16-A	Basic	GPI1 – 16-Bit I/O TTL Interface
13	M7311	DB16-A	Option	GPI2 – 16-Bit I/O TTL Interface
14	M7311	DB16-A	Option	GPI3 – 16-Bit I/O TTL Interface
15	M7311	DB16-A	Option	Interface for 8 or 16 X 246 Data PROM
16	M7313	DC16-A	Option	SI1 – Asynchronous Serial I/O Interface

Table 2-1 (Cont)
Module Slot Assignment and Description

Slot	Module No.	Model No.	Configuration	Description
17	M7313	DC16-A	Option	SI2 – Asynchronous Serial I/O Interface
18	M7329	PCS16-D	Basic	MUX0 – Input Multiplexer
19	M7329	PCS16-D	Option	MUX1 – Input Multiplexer
20	M7336	PCS16-A	Basic	Processor Control

NOTE
The following variation is permitted for sockets A6 and A7.

6	M7324	MS16-E	Option	MEM1 – 1K X 16 R/W MOS Memory
7	M7324	MS16-E	Option	MEM2 – 1K X 16 R/W MOS Memory
		Row B		
1				Test Slot
		Row C		
1	M7310	KEV16	Basic	PAGE0 and PAGE1 Control
2	M7306	KFL16	Basic	FF1, FF2 and FF3
3	M7328	PCS16-C	Basic	Evoke Decoder 000–037
4	M7328	PCS16-C	Basic	Evoke Decoder 040–077
5	M7328	PCS16-C	Basic	Evoke Decoder 100–137
6	M7328	PCS16-C	Option	SP1–SP16 Evoke Decoder 140–177
7	M7328	PCS16-C	Option	SP17–SP32 Evoke Decoder 200–237
8	M7328	PCS16-C	Basic	Evoke Decoder 240–277
9	M7306	KFL16	Basic	Link, MUX Select, Page Select
10	M1307	KOR16-B	Basic	Control Logic
11	M1307	KOR16-B	Basic	Control Logic
12	M1307	KOR16-B	Basic	Control Logic
13	M1307	KOR16-B	Basic	Control Logic
14	M1307	KOR16-B	Basic	Control Logic
15	M7327	MR16-FE	Option	8 X 246 Data PROM (lower 8 bits)
16	M7327	PCS16-B	Basic	Control PROM Loc 0000–0377
17	M7327	PCS16-B	Option	Control PROM Loc 1000–1377
18	M1307	KOR16-A	Basic	Control Logic
19		I/O Socket	Basic	EXT1–EXT23, FF1–FF6, SI1, SI2, MSYNC and SSYNC
20		I/O Socket	Basic	GPI2 I/O and FF2
		Row D		
1	M908	Panel Socket	Basic	Front Panel and Autostart (SWCAB)
2	M706	KFL16	Option	FF4, FF5 and FF6
9	M1103	KOR16-A	Basic	Control Logic
10	M1307	KOR16-B	Basic	Control Logic
11	M1307	KOR16-B	Basic	Control Logic

Table 2-1 (Cont)
Module Slot Assignment and Description

Slot	Module No.	Model No.	Configuration	Description
			Row D	
12	M1307	KOR16-B	Basic	Control Logic
13	M1307	KOR16-B	Basic	Control Logic
14	M623	DA16-F	Option	PDP-11 MSYNC and SSYNC Interface
15	M7327	MR16-E	Option	8 X 256 Data PROM (upper 8 bits)
16	M7327	PCS16-B	Option	Control PROM Loc 0400–0777
17	M7327	PCS16-B	Option	Control PROM Loc 1400–1777
18	M7333	DC16-B	Option	Serial I/O Interface Adapter
19		I/O Socket	Basic	GPI1 I/O and FF1
20		I/O Socket	Basic	GPI3 I/O and FF3

2.3 SPECIFICATIONS

2.3.1 Processor

Word Length

Control Program:	8 bits
Memory Address:	9 bits (10th bit is programmable)
Program Data:	16 bits

Memory

Programmed Instruction:	256-word reprogrammable control ROM (PROM) – expandable to 1024 in 256-word increments
Program Data (Constants):	4-word diode ROM – expandable by 24 words and/or 256 8 or 16-bit words
Auxiliary Data Storage:	256 to 2048 words of 16-bit read/write MOS memory

Control PROM

Type:	Electrically alterable quartz window ROM.
Organization:	256 8-bit words
Minimum Propagation Delay:	300 ns
Maximum Propagation Delay:	1 μs
Voltage Spec:	+5V ± 5% -9V ± 5%

Outputs:	1 TTL unit load drive; tri-state output or open collector drivers.
Address:	8-bit TTL address; internally decoded; 2 memory select inputs.
Programming:	The semiconductor PROM control memories are programmed by using a special electrical interface. The quartz window PROM can be erased with ultraviolet light and reprogrammed at least 100 times.
Scratchpad Register	1 (byte addressable) – expandable by 16- or 32 word-addressable registers.
Accumulators	1 (A)
Argument Register	1 (B)
I/O Channels	
Flags (Boolean Outputs):	3 – expandable to 6
Boolean Inputs:	6 – expandable to 22
Parallel:	1 (PDP-11 Unibus compatible) – expandable by 2 TTL 16-bit data I/O channels

NOTE

The three standard flags are available at the channel interface for I/O synchronization.

Serial: 2 (optional)

NOTE

The serial channels will accommodate baud rates of 110, 150, 300, 600, 1200, or 2400; one or two stop bits; and 5, 6, 7, or 8 data bits.

Instructions

	Maximum Execution Time	Machine Code
LET:	2.4 μs	$0–277_8$
GOTO:	3.2 μs	300_8 and 301_8
IF:	2 μs if false 3.2 μ s if true	$302_8–373_8$
CALL:	3.2 μs	374_8 and 375_8
EXIT:	3.2 μs	376_8 or 377_8

NOTE

The LET and EXIT instructions require one 8-bit memory location each and the GOTO, IF, and CALL instructions use two 8-bit locations each, one for the operation code and the other for the jump address. The GOTO and IF instructions are handled by the same logic with the condition for the GOTO instruction always true.

Bus

Pin Assignments:
(Slots A1—A17)

Bit	Pin	Bit	Pin
0	AA1 (LSB)	8	AK1
1	AB1	9	AL1
2	AC1	10	AM1
3	AD1	11	AN1
4	AE1	12	AP1
5	AF1	13	AR1
6	AH1	14	AS1
7	AJ1	15	AU1 (MSB)

Control	Pin
Overflow	BA1
Power Clear	BB1
Data Accept	BC1
Done	BD1
Data Ready	BE1

Voltage: Logic 1 = 0 to 0.4V
Logic 0 = 3.0 to 4.0V

Current: Logic 1 = 24 to 31 mA with one terminator
Logic 0 = 1.5 to 4.0 mA

2.3.2 Mechanical

Chassis

Dimensions: 19 X 13 X 10.44 in.; 48 X 33 X 26.5 cm

Fans: Two fans exhaust from left side of cabinet. Filter is located on right side of cabinet.

Weight: 55 lb (approx); 25 kg

Mounting: Chassis slides for rack mounting in standard 19 in. cabinet. Without slides, the cabinet may be used as a table top unit.

Front Panel

Run Light:	LED indicator is on when the program is running. It is turned off by the HALT instruction.
Power Light:	LED indicator is on when +5V is available from internal power supply.
START Switch:	Initiates program execution
Panel Lock:	Disables the START switch.

2.3.3 Electrical

Primary Power:	PDP-16/MA: 115 Vac; 47–63 Hz; 2A maximum
	PDP-16/MB: 230 Vac; 47–63 Hz; 1A maximum
H740 Power Supply:	+5V @ 17A, -15 @ 2A

2.3.4 Environmental

Temperature:	0 to 60°C ambient
Relative Humidity:	95% maximum (without condensation)
Altitudes:	10K feet; 3000 meters
Vibration:	1.89G RMS overall from 0–70 Hz. Acceleration spectral density – 0.029 G^2/Hz from 10–50 Hz with an approximate 8 dB/octave roll-off from 50 to 70 Hz.

2.4 CONFIGURATION DATA

The basic PDP16-M can be expanded simply by inserting the desired option into its preassigned slot. To implement some options, some prerequisites to the basic machine are required. These prerequisites, the resident slot in the logic assembly, the module number, the option number, and name and the purpose of each option are detailed in Table 2-2.

Table 2-2
Configuration Data

Purpose	Option Name	Option Model No.	Module No.	Slot	Prerequisite
Increase storage for control program to 512 words	Control PROM1	PCS16-B	M7327	D16	None
Increase storage for control program to 768 words	Control PROM 2 or 3	PCS16-B	M7327	C17 or D17	PCS16-B in slot D16

Table 2-2 (Cont)
Configuration Data

Purpose	Option Name	Option Model No.	Module No.	Slot	Prerequisite

NOTE

If only 768 words are implemented, it is desirable to insert the third control PROM in slot D17 (thereby defining it as the fourth PROM with the third PROM not implemented) to avoid complicated page linking code (Paragraph 4.5.2).

Increase storage for control program to 1024 words	Control PROM 3	PCS16-B	M7327	D17	PCS16-B in slots D16 and C17
Increase storage for program data (constants) from 4 to 28 words	Constant Generator (K)	MR16-D	M7325	AB8	None
Increase storage for program data (constants) from 4 to 260 words	Data PROM 1 and Data PROM 2	MR16-E MR16-E	M7327 M7327	D15 C15	DB16-A in slot AB15

NOTE

Data PROM 1 and 2 must both be used (MR16-F) if 16-bit data is required. If only 8-bit data is to be stored (such as characters for messages), then only one or the other need be implemented. Both the constant generator (K) and the Data PROMs can be implemented to extend data storage to 288 words.

Increase Scratchpad Registers from 1 to 17 or 1 to 33	Fast Registers (SP1 to 16)	MS16-C	M7318	AB4	PCS16-C in slot CD6
	Fast Registers (SP17 to 32)	MS16-C	M7318	AB3	PCS16-C in slot CD7

One scratchpad register, designated the transfer register (TR), is implemented in the basic machine. If additional scratchpad registers are required, either or both of the above options can be implemented.

Add MOS Read/Write Memory for Auxiliary Data Storage	MEM 1 and 2				
256 words	MEM 1	MS16-D	M7319	AB6	None

Table 2-2 (Cont)
Configuration Data

Purpose	Option Name	Option Model No.	Module No.	Slot	Prerequisite
512 words	MEM 2	MS16-D	M7319	AB7	MS16-D in slot AB6
1024 words	MEM 1	MS16-E	M7324	AB6	None
1280 words	MEM 2	MS16-D	M7319	AB7	MS16-E in slot AB6
2048 words	MEM 2	MS16-E	M7324	AB7	MS16-E in slot AB6

NOTE

MEM 2 can be implemented without MEM 1. The listing above serves only to illustrate what is required to expand the R/W memory from the minimum through all available sizes to the maximum.

Expand Boolean inputs (EXT) from 6 to 22; test even bits of A register; test LSB and MSB of B register; test LINK; or test PWOK using IF instruction	MUX 1	PCS16-D	M7329	AB19	None
Expand flags from three to six	Flags FF4 to 6	KFL16	M7306	D2	None
Add second Parallel I/O	GPI2	DB16-A	M7311	AB13	None
Add third Parallel I/O	GPI3	DB16-A	M7311	AB14	DB16-A in slot AB13
Add one serial I/O	SI1	DC16-A	M7313	AB16	DC16-B
Add second serial I/O	SI2	DC16-A	M7313	AB16	DC16-B
Data PROM option Interface	GPI	DB16-A	M7311	AB15	None
Decode EVOKES for SP1–16	EVOKE Decoder	PCS16-C	M7328	CD6	None
Decode EVOKES for SP17–32	EVOKE Decoder	PCS16-C	M7328	CD7	None

CHAPTER 3
INTERFACING

The PDP16-M is an asynchronous 16-bit parallel transfer machine. Communication between the PDP16-M and external equipment is accomplished via the following interface channels:

One Multiplexed (MUX) Input-Output Channel (TTL)

Two Serial Input-Output Channels (20-mA current loop or TTL)

Three Parallel 16-Bit Input-Output Channels (TTL)

Interfacing the PDP16-M with the outside world is a user task; therefore, detailed planning is required on the part of the user to ensure effective trouble-free interfacing. This chapter describes the PDP16-M input and output characteristics. The user must be thoroughly familiar with these characteristics in planning his interface requirements. Except for the optional serial I/O channel TTY current loops, all input and output signals are TTL levels and are brought to pins of specific slots on the logic assembly (Figure 3-1). Cables are inserted in these slots to bring the input and output signals to the outside world. All signals are high for assertion. Signals must be buffered with K or M-series modules if distances greater than 5 feet are to be driven or received by the interface slots. Mate-N-Lok connectors are provided on the SI Adapter module to interface with serial I/O channel TTY current loops.

Because the PDP16-M has a limited number of I/O channels just like any other controller and computer, the user must plan his interface carefully. If the interface is TTL compatible and the number of sensors, controls, and data paths for a given application do not exceed the I/O capability of the PDP16-M, direct TTL compatible interface connections may be made with only minimal planning for assigning the I/O signals to the I/O channels. However, if the number of sensors, controls, and data paths exceed the I/O capability, additional interface components may have to be designed or may have to be selected from the commercial market for concentrating (multiplexing) these signals.

Digital Equipment Corporation has off-the-shelf PDP-11 peripheral devices for concentrating digital I/O, analog inputs, and analog outputs. These devices may be interfaced directly with the PDP16-M if the DA16-F option is implemented (Paragraph 3.9). For those applications that require other than TTL compatible signals, DEC offers A, K, and M series modules that convert TTL levels to accommodate most input or drive requirements.

After the interface configuration is firm, the user can start his programming effort.

3.1 PARALLEL I/O DATA TRANSFERS

Parallel data transfers are accomplished using the general purpose interfaces (GPIs) or an external data bus configuration.

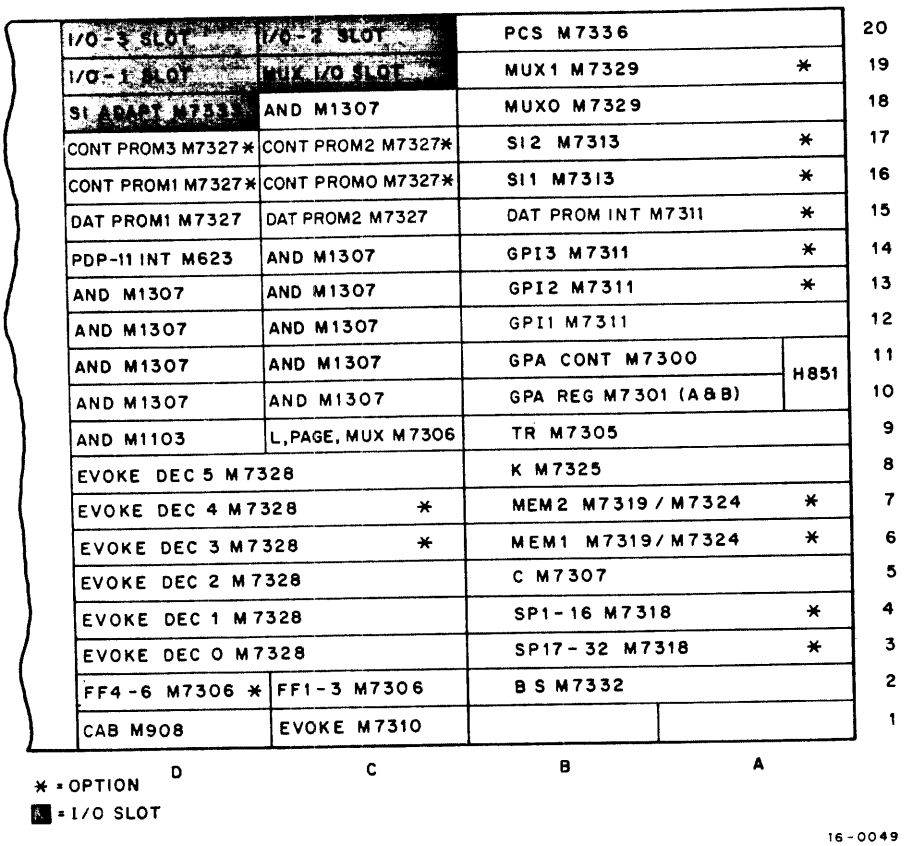


Figure 3-1 Logic Assembly I/O Slot Location

3.1.1 GPI Interfacing

Sixteen bit data words may be transferred directly to or from the PDP16-M via the three GPIs by utilizing the I/O commands GPI1=A, GPI2=A, A=GPI1, etc. Register A in the arithmetic unit (GPA) is used as both a source and destination for I/O commands. Figure 3-2 shows a simple I/O interface where analog data is first addressed, then strobed into the PDP16-M.

3.1.2 External Data Bus Interfacing

By implementing the external bus option (DA16-F), an external asynchronous bus may be developed. In this configuration the output half of GPI1 is normally used as an address register, the PDP16-M data bus is extended to the external interface by cable, and the DA16-F option is used to generate timing signals MSYN and SSYN. If both input and output transfers are required, a program flag (FF1) can be used to denote input or output. Figure 3-3 shows the physical and electronic implementation of a gauging system where external counters are preloaded, counted down by external pulse trains, and read back into the PDP16-M using the external bus configuration.

The external bus configuration can also be used to interface PDP-11 peripherals which are not direct memory access devices. Figure 3-4 shows an interface to the PC11 Paper-Tape Reader/Punch.

3.2 SERIAL I/O TRANSFERS

Two serial I/O interfaces may be implemented with the DC16-A options. The output of the DC16-A is 20 mil current loop and TTL level compatible; 20 mil current loop outputs are available on 8-pin Mate-N-Lok connectors mounted on the DC16-B Serial Interface Adapter option. The TTL levels are available on the MUX and FF I/O interface slot. These outputs may be converted to EIA, current mode, or other required levels using M or K series modules located on an external interface.

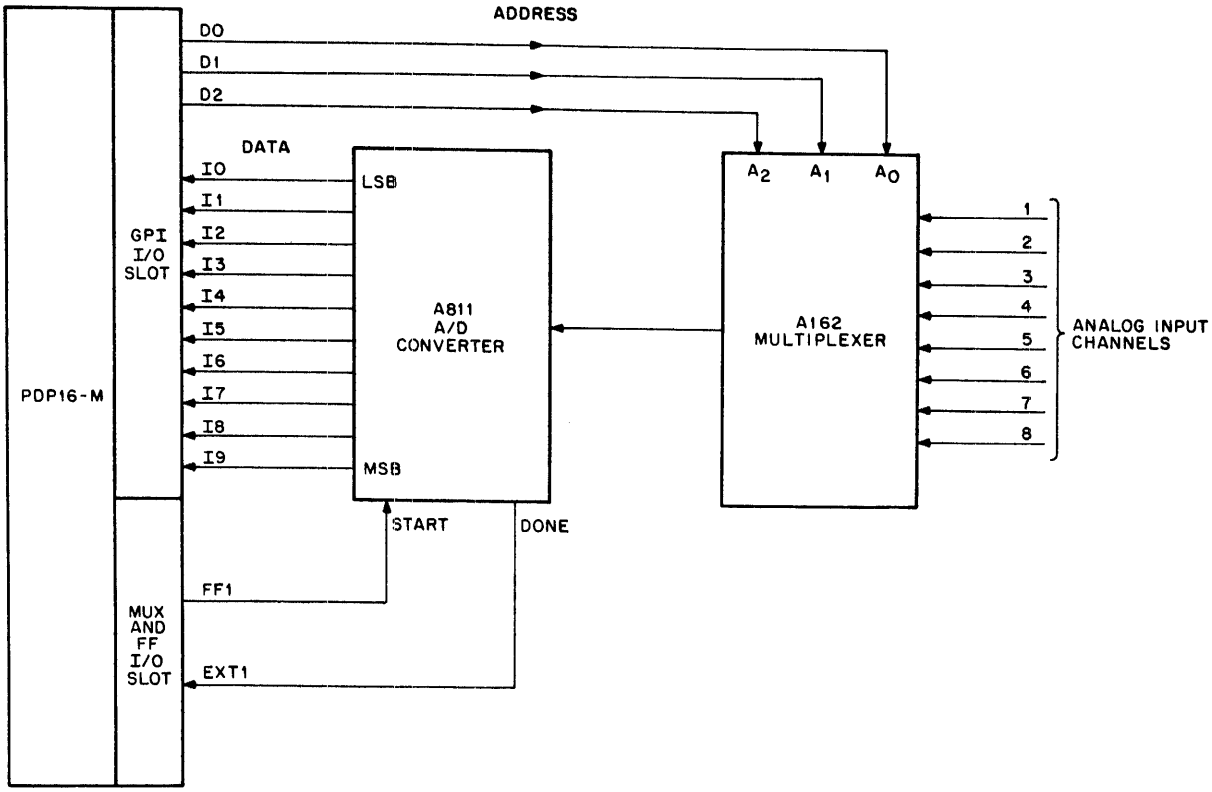


Figure 3-2 GPI Interfacing

3.3 MUX AND FF I/O INTERFACE

The multiplexer and flip-flop I/O slot contains the output signals of the six programmable flags (FF1–FF6), the four serial I/O signals, MSYN, SSYN, and up to 22 Boolean input signals. The Boolean inputs are labelled EXT1 – EXT22. These signals normally represent the state of external flags or switches and must be TTL compatible. These signals are referenced by the conditional branch instruction (IF statements) in the PDP16-M software.

3.4 MUX AND FF I/O SLOT C19

This slot provides the interface connections for the following signals:

a. Boolean Inputs (EXT1–22)

b. Boolean Outputs (FF1–6)

c. Serial I/O Channel TTL Inputs

d. Serial I/O Channel TTL Outputs

e. UNIBUS signals

MSYN
SSYN
C1 (FF1)

f. Continue Signal

The pin assignments and TTL loading data is given in Table 3-1.

Any one of four standard cables can be used for interfacing with the MUX and FF I/O slot. These cables are described in Paragraph 3.11.

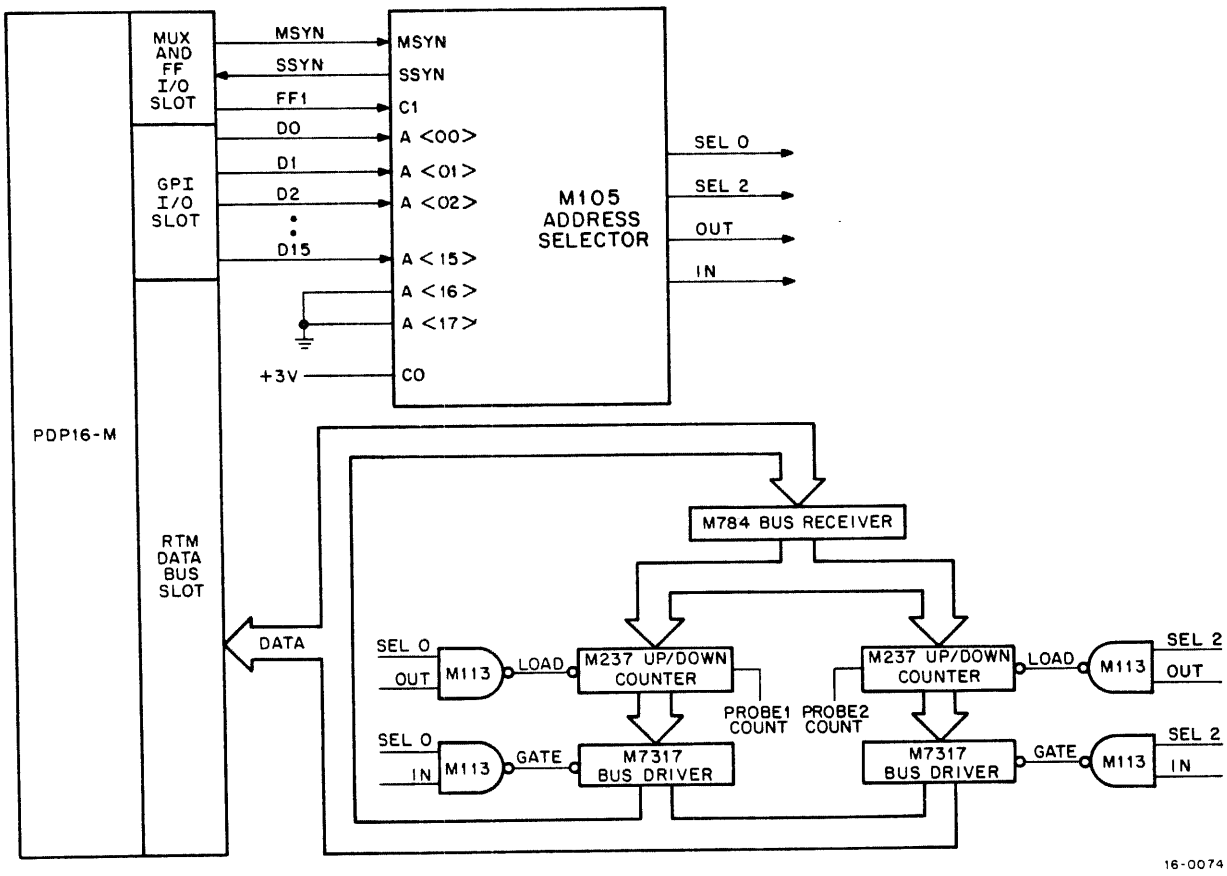


Figure 3-3 External Data Bus Interfacing

3.5 GPI1 I/O SLOT D19

This slot provides the interface connections for the following signals:

a. Input Data Line (I00–15)

b. Output Data Lines (D00–15)

c. Boolean Output/Flag (FF1)

The pin assignments and TTL loading data are given in Table 3-2. Any one of four standard cables can be used for interfacing with the GPI1 I/O slot. These cables are described in Paragraph 3.11.

3.6 GPI2 I/O SLOT C20

This slot provides the interface connections for the following signals:

a. Input Data Lines (I00–15)

b. Output Data Lines (D00–15)

c. Boolean Output/Flag (FF2)

The pin assignments and TTL loading data are given in Table 3-2. Any one of four standard cables can be used for interfacing with the GPI2 slot. These cables are described in Paragraph 3.11.

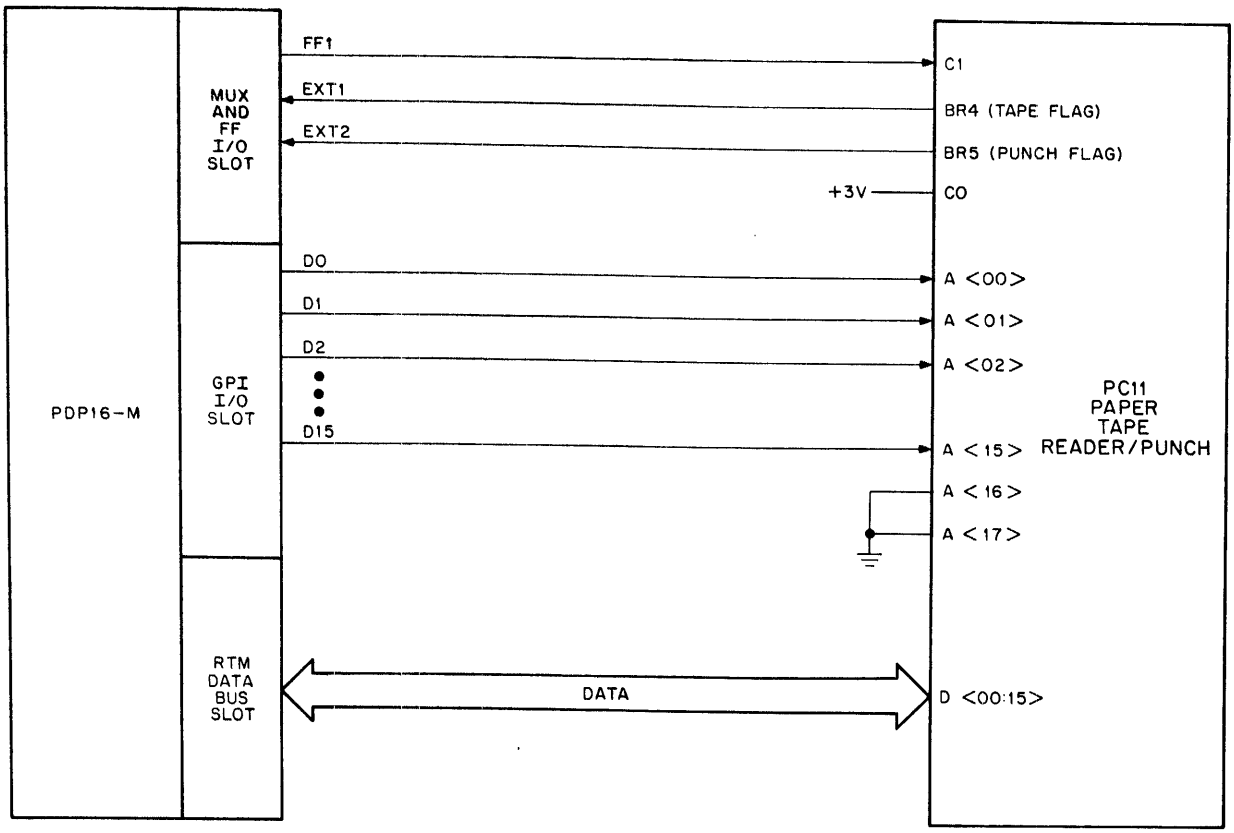


Figure 3-4 PDP-11 Peripheral Interfacing

Table 3-1
MUX and FF I/O Slot C19

Cable 1			Cable 2		
Pin	Signal	TTL Loading	Pin	Signal	TTL Loading
A1	EXT 1	1	A2	EXT 21	1
B1	EXT 2	1	B2	EXT 18	1
C1	EXT 3	1	C2	EXT 22	1
D1	EXT 4	1	D2	EXT 19	1
E1	EXT 5	1	E2	SI 1 SO (TTL)*	10
F1	EXT 6	1	F2	SI 1 SI (TTL)*	1
H1	EXT 7	1	H2	SI 2 SO (TTL)*	10
J1	EXT 8	1	J2	SI 2 SI (TTL)*	1
K1	EXT 9	1	K2	MSYN	—
L1	EXT 10	1	L2	EXT 20	1
M1	EXT 11	1	M2	CONTINUE	2
N1	EXT 12	1	N2	FF1	8
P1	EXT 13	1	P2	FF2	8
R1	EXT 14	1	R2	FF3	8
S1	EXT 15	1	S2	FF4	8
T1	GROUND	—	T2	FF5	8
U1	EXT 16	1	U2	FF6	8
V1	EXT 17	1	V2	SSYNC	2

*20 mil current loop (TTY) interfacing connectors are on SI Adapter.

Table 3-2
GPIn* I/O Slot

Cable 1			Cable 2		
Pin	Signal	TTL Loading	Pin	Signal	TTL Loading
A1	I00	1	A2		
B1	I01	1	B2	D00	10
C1	I02	1	C2		
D1	I03	1	D2	D01	10
E1	I04	1	E2	D02	10
F1	I05	1	F2	D03	10
H1	I06	1	H2	D04	10
J1	I07	1	J2	D05	10
K1	I08	1	K2	D06	10
L1	I09	1	L2	D07	10
M1	I10	1	M2	D08	10
N1	I11	1	N2	D09	10
P1	I12	1	P2	D09	10
R1	I13	1	R2	D10	10
S1	I14	1	S2	D12	10
T1	—	—	T2	D13	10
U1	I15	1	U2	D14	10
V1	FFn*	10	V2	D15	10

*n=1=slot D19
n=2=slot C20
n=3=slot D20

3.7 GPI3 I/O SLOT D20

This slot provides the interface connections for the following signals:

a. Input Data Lines (I00–15)

b. Output Data Lines (D00–15)

c. Boolean Output/Flag (FF3)

The pin assignments and TTL loading data are given in Table 3-2. Any one of four standard cables can be used for interfacing with the GPI3 I/O slot. These cables are described in Paragraph 3.11.

3.8 RTM DATA BUS SLOTS AB01 – AB17

These slots are interconnected to form the RTM data bus of the PDP16-M. Normally, these slots are occupied by the register transfer modules (functional elements of the PDP16-M). However, when interfacing with devices that require a data bus extension, a vacant slot is used to establish that data path. The pin assignments of the RTM data bus are given in Table 3-3.

Any one of four standard cables can be used for interfacing with the RTM data bus. These cables are described in Paragraph 3.11.

Table 3-3
RTM Data Bus Slots AB01 – AB17

Cable 1		Cable 2	
Pin	Signal	Pin	Signal
A1	Bit 00	A2	Not Used
B1	Bit 01	B2	Not Used
C1	Bit 02	C2	Not Used
D1	Bit 03	D2	Not Used
E1	Bit 04	E2	Not Used
F1	Bit 05	F2	Not Used
H1	Bit 06	H2	Not Used
J1	Bit 07	J2	Not Used
K1	Bit 08	K2	Not Used
L1	Bit 09	L2	Not Used
M1	Bit 10	M2	Not Used
N1	Bit 11	N2	Not Used
P1	Bit 12	P2	Not Used
R1	Bit 13	R2	Not Used
S1	Bit 14	S2	Not Used
T1		T2	Not Used
U1	Bit 15	U2	Not Used
V1		V2	Not Used

NOTE

BC02X or BC03H cable must be inserted in row A, not row B, of the specified slots. Row B carries the data bus interlocked timing signals DATA READY, DATA ACCEPT DONE, OVERFLOW, and POWER CLEAR.

3.9 PDP-11 PERIPHERAL INTERFACE CONNECTIONS

The PDP16-M has been prewired to interface with PDP-11 peripheral devices that do not need to become master devices. Only accumulator type transfers can transpire between the PDP16-M and PDP-11 peripherals. One prewired module slot, slot D14, is reserved on the logic assembly for installing the DA16-F option (module M623). This module contains AND gate bus drivers for interfacing with the data transfer interlock signals of the PDP16-M and the PDP-11 device. In addition to the M623 module, the following items are also required when connecting a PDP-11 device to a PDP16-M:

a. One H803 Connector Block

b. Three BC02X-05 or three BC03H-05 cables

c. One KTM16 Bus Terminator Option (M962)

NOTE

A BC11A Cable (preferably BC11A-02) is also required. This cable is supplied with the PDP-11 peripheral device.

The control signals, the address, and the data required for operating a PDP-11 peripheral are distributed between three slots on the PDP16-M logic assembly; three cables are, therefore, required. Since the pin assignments for the various signals on the PDP16-M I/O slots do not match those of the PDP-11 device input slot, an H803 Connector Block is employed to match the signals through wire wrapping. In addition, the H803 Connector Block is used to provide ground and +3V to certain PDP-11 device input lines. Figure 3-5 and Table 3-4 detail the pins on the connector block to be wirewrapped.

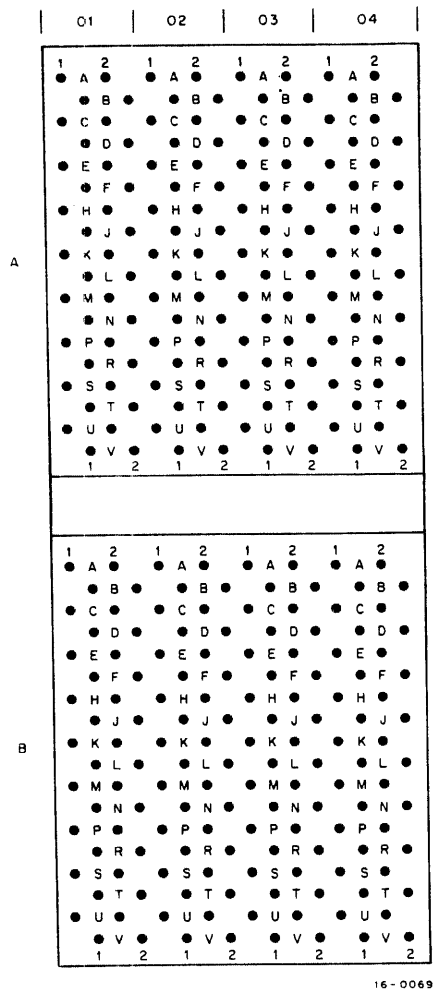


Figure 3-5 H803 Connector Block Pin Assignments

After the H803 Connector Block is wirewrapped, the cables can be installed as shown in Figure 3-6. The KTM16 option must be installed in slot A03 of H803 to terminate the data bus.

NOTE

A new PDP-11 peripheral interface option, the UNIBUS converter I/O interface module, will be available in the near future. This option will simplify the interfacing procedure.

Table 3-4
H803 Connector Block Wirewrap Connections

PDP16-M Signal Name	From	To	From	To	PDP-11 Signal Name
+5V	*A01A2	A04A2	A04A2	B04A2	+5V
GND	*A01C2	A04C2	A04C2	A04T1	GND

Table 3-4 (Cont)
H803 Connector Block Wirewrap Connections

PDP16-M Signal Name	From	To	From	To	PDP-11 Signal Name
GND	*A01T1	A04T1	A04T1	B04C2	GND
			B04C2	B04T1	GND
+5V	*B01A2			B04A2	+5V
GND	*B01C2			B04C2	GND
GND	*B01T1			B04T1	GND
+5V	A01A2	A03A2	A03A2	B03A2	+5V
+5V	B01A2			B03A2	+5V
GND	A01C2	A03C2	A03C2	B03C2	GND
GND	B01C2			B03C2	GND
GND	A01T1	A03T1	A03T1	B03T1	GND
GND	B01T1			B03T1	GND
DATA BIT 00	A01A1	A03A1	A03A1	A04C1	D00
DATA BIT 01	A01B1	A03B1	A03B1	A04D2	D01
DATA BIT 02	A01C1	A03C1	A03C1	A04D1	D02
DATA BIT 03	A01D1	A03D1	A03D1	A04E2	D03
DATA BIT 04	A01E1	A03E1	A03E1	A04E1	D04
DATA BIT 05	A01F1	A03F1	A04F1	A04F2	D05
DATA BIT 06	A01H1	A03H1	A04H1	A04F1	D06
DATA BIT 07	A01J1	A03J1	A03J1	A04H2	D07
DATA BIT 08	A01K1	A03K1	A03K1	A04H1	D08
DATA BIT 09	A01L1	A03L1	A03L1	A04J2	D09
DATA BIT 10	A01M1	A03M1	A03M1	A04J1	D10
DATA BIT 11	A01N1	A03N1	A03N1	A04K2	D11
DATA BIT 12	A01P1	A03P1	A03P1	A04K1	D12
DATA BIT 13	A01R1	A03R1	A03R1	A04L2	D13
DATA BIT 14	A01S1	A03S1	A03S1	A04L1	D14
DATA BIT 15	A01U1	A03U1	A03U1	A04M2	D15
+3V			B03A1	B04F1	AC LO
+3V			B03B1	Bo4F2	DC LO
D00	B01B2			B04H2	A00
D01	B01D2			B04H1	A01
D02	B01E2			B04J2	A02
D03	B01F2			B04J1	A03
D04	B01H2			B04K2	A04
D05	B01J2			B04K1	A05
D06	B01K2			B04L2	A06
D07	B01L2			B04L1	A07
D08	B01M2			B04M2	A08
D09	B01N2			B04M1	A09
D10	B01P2			B04N2	A10
D11	B01R2			B04N1	A11
D12	B01S2			B04P2	A12
D13	B01T2			B04P1	A13
D14	B01U2			B04R2	A14
D15	B01V2			B04R1	A15

Table 3-4 (Cont)
H803 Connector Block Wirewrap Connections

PDP16-M Signal Name	From	To	From		PDP-11 Signal Name
GND			B04T1	B04S2	A16
GND			B04S2	B04S1	A17
+3V			B03C1	B04U2	C0
MSYN	B02K2	B03D1	B03D1	B04V1	MSYN
SSYN	B02V2	B03E1	B03E1	B04U1	SSYN
FF1	B02N2			B04T2	C1
GND	B02T1			B04T1	GND

*Use 933 Bus Strips for these connections (power and ground). For all other connections use 24 AWG Bus Wire.

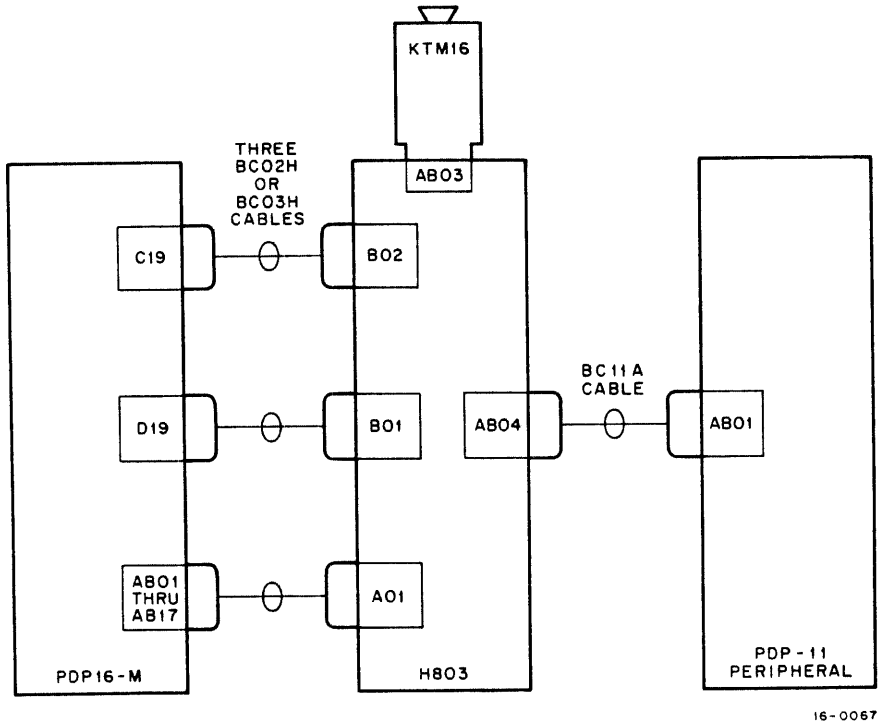


Figure 3-6 PDP-11 Peripheral Interface Cabling Diagram

3.10 SI1 AND SI2 INTERFACE CONNECTIONS

The serial I/O channel options are equipped to be interfaced with TTL or TTY compatible signals. The TTL compatible input and output connections are made available at the MUX and FF I/O slot (Paragraph 3.4). The TTY 20 mil current loop connections are made available through Mate-N-Lok connectors on the SI Adapter module. The standard teletype cable that comes with the asynchronous device can be connected directly to the Mate-N-Lok connectors.

Three prewired module slots are reserved on the PDP16-M logic assembly for implementing the serial I/O channels. They are:

Slot	Module	Name
AB16	M7313	SI1
AB17	M7313	SI2
D18	M7333	SI Adapter

The M7333 SI Adapter Module contains a set of split lugs and a Mate-N-Lok connector for each channel (Figure 3-7). The split lugs facilitate jumper installation for selecting the desired bit stream format. The number of DATA bits, STOP bits and channel baud rate can be selected to complement the terminal device. Use the following chart for installing the required jumpers.

Stop Bits	1	2				
SB	Yes	No				
Data Bits	**5**	**6**	**7**	**8**		
NB1	Yes	No	Yes	No		
NB2	Yes	Yes	No	No		
Baud Rate	**110**	**150**	**300**	**600**	**1200**	**2400**
110	Yes					
150		Yes				
300			Yes			
600				Yes		
1200					Yes	
2400						Yes

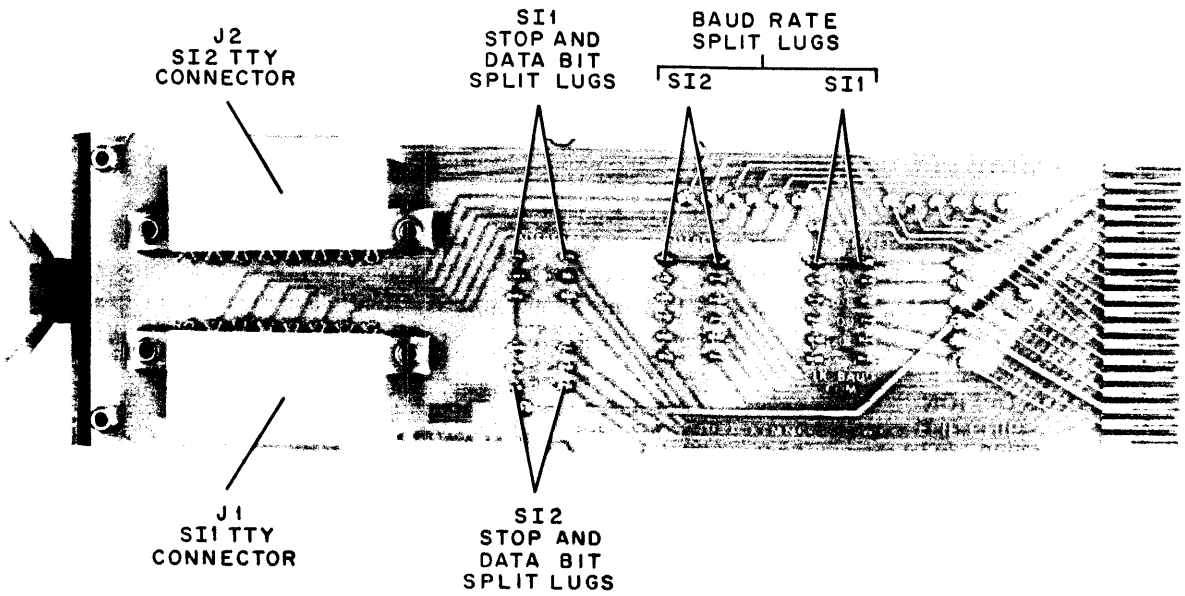


Figure 3-7 SI Adapter Module M7333, TTY Connectors, and Split Lug Identification

The Mate-N-Lok connector is the TTY 20 mil current loop interface connector for interfacing with the terminal device. TTL compatible serial I/O connections are made available on the MUX and FF I/O slot (C19).

3.11 INTERFACE CABLING

There are four different types of standard cables available for interfacing the outside world with the PDP16-M TTL I/O signals of the MUX and FF I/O and GPI I/O slots (Figure 3-8); two are Flexprint®-type cables and two are ribbon-type cables.

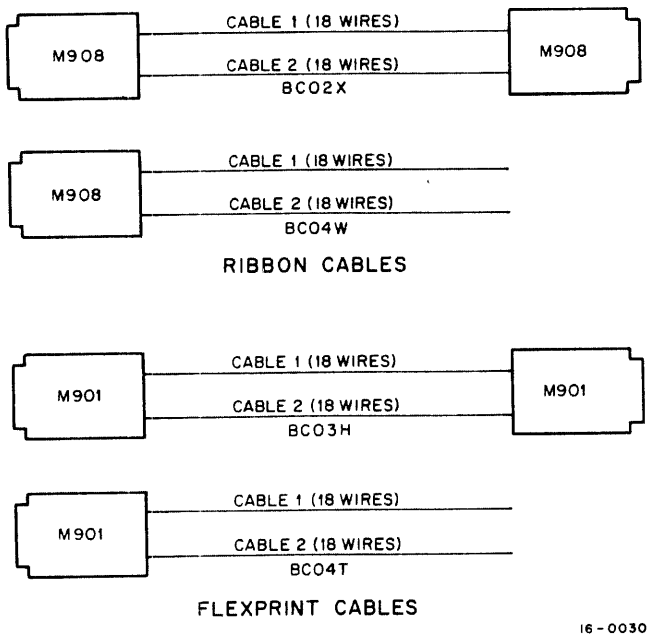


Figure 3-8 Interface Cables

The ribbon cables are easier to work with and have a low resistance which is suitable for long lines. The Flexprint cables require a special tool for stripping the wires. There are two types of ribbon and Flexprint cables: One is terminated with a single-height module at each end; the other is terminated with a single-height module at only one end. The type required by a given user depends entirely on the application.

3.12 MANUAL/AUTO RUN OPTION

Two methods for starting the PDP16-M program are available to the user: manual and automatic. In the manual mode, the user must press the START switch on the console; in the auto mode, the program is automatically started when the machine is turned on locally or remotely. The auto mode can be selected by the user simply by installing a jumper wire between pins S1 and U1 on slot D01. If this jumper wire is not installed, the program must be started manually.

3.13 CONTINUE OPTION

The HALT instruction can be used in the program to stop the program based on the outcome of a test or some other condition. The CONTINUE signal, which is available at the MUX and FF I/O slot, provides the control for continuing the program from where the program halted. A logic 0 (low) on the continue line will restart the program. Depressing the START switch will always cause the program to start at location 000 of the page where the halt occurred.

® Flexprint is a registered trademark of Sanders Associates, Inc.

CHAPTER 4
WRITING THE PROGRAM

The PDP16-M program is stored in a solid state, reprogrammable, read-only-memory (PROM). The basic memory module is 8 bits by 256 words. Up to four of these modules can be implemented in the PDP16-M. A special utility option (MR16-SL) is required for loading the PROM with the object code (Chapter 6). The object code (punched on a paper tape) is produced by assembling the source program on a PDP-8/E using the PAL16 Assembler. A PDP-8/E based Symbolic Editor provides the means for preparing the source program on-line and punching the ASCII source paper tape. Besides loading the PROM, the utility option permits the user to simulate and verify his program. The simulate function allows the user to exercise his control program for debugging purposes before loading the PROM. Once a PROM is loaded, the verify function can be used to check the contents of the PROM. The source program must be written in PAL16 Assembly language. This chapter describes the language in detail.

Five classes of instructions have been implemented for the PDP16-M. They are:

a. Register — Transfer (LET)

b. Unconditional Branch (GOTO)

c. Conditional Branch (IF)

d. Jump to Subroutine (CALL)

e. Return from Subroutine (EXIT)

4.1 REGISTER TRANSFER INSTRUCTION

The register transfer instruction is used to specify arithmetic, logical, data transfer, I/O, and control operations. Seventy unique register transfer instructions have been implemented in the basic PDP16-M. The general format of the instruction follows:

LET register get register-operation/comment

For example: LET A = A+1/INCREMENT A

The word LET is optional. If used, a space must appear between LET and A=A+1.

4.1.1 Arithmetic Group

The following arithmetic operations can be performed on the contents of the A and B registers:

```
A+1          B/2
A-1
A+B
A-B
A/2
AX2
```

Two's complement arithmetic is used for the add and subtract operations. The symbols / and X specify right and left shift operations, respectively. Shifting a register one bit to the right causes the contents of the register to be divided by two. Shifting the register one bit to the left multiplies the contents by two. Notice that the B register is equipped only with the right shift (/) operation. The result from the specified arithmetic operation can be transferred to either the A or the B register. For example:

```
A=A+B
B=A+B
```

Carries (overflow) from arithmetic operations can be saved. For any arithmetic instruction described above, the overflow (1 or 0) is saved simply by suffixing the instruction with (S). For example:

```
A=A+B(S) / SAVE SIGN
A=A-B(S) / SAVE SIGN
A=AX2(S) / SAVE MSB
A=A/2(S) / SAVE LSB
```

The overflow is saved until another save instruction is executed. This feature is extremely useful in coding multiply, divide, and other algorithms. The saved overflow can be tested for conditional branching or shifted into the LINK register for propagating the carry. For example:

```
A=AX2(S)
IF OVF,LABEL        /IF OVF IS ONE JUMP TO LABEL
L=OVF               /LINK GET OVF
```

The LINK register supplies the logic level for the LSB during left shift operations (A=AX2) and the MSB during right shift operations (A=A/2).

Besides propagating the carry, the LINK register is useful in generating constants. Constants can be generated by setting and resetting the LINK and shifting the register. For example, the constant 0042_8 (000 000 100 010) is generated by the following instructions:

```
A=0
L=1
A=AX2 / A contains octal 0001
L=0
A=AX2 / A contains octal 0002
A=AX2 / A contains octal 0004
A=AX2 / A contains octal 0010
L=1
A=AX2 / A contains octal 0021
L=0
A=AX2 / A contains octal 0042
```

If space is available in the control PROM and the constants required by the program never need to be changed, it is desirable to generate the constants and store them in the TR or the optional scratch pad (SP) rather than using the constant generator (C or K). Table 4-1 lists the complete arithmetic instruction set of PDP16-M.

Table 4-1
Arithmetic Instruction Set

A Register	
Overflow is not Saved	**Overflow is Saved**
A=0	
A=B	
A=A+1	A=A+1(S)
A=A-1	A=A-1(S)
A=A+B	A=A+B(S)
A=A-B	A=A-B(S)
A=A/2	A=A/2(S)
A=AX2	A=AX2(S)
A=B/2	A=B/2(S)
B Register	
B=0	
B=A	
B=A+1	B=A+1(S)
B=A-1	B=A-1(S)
B=A+B	B=A+B(S)
B=A-B	B=A-B(S)
B=A/2	B=A/2(S)
B=AX2	B=AX2(S)
B=B/2	B=B/2(S)
LINK Register	
L=0	
L=1	
L=LNOT	
L=OVF	

4.1.2 Logical Group

The following logical operations can be performed on the contents of the A and B registers:

ANOT
BNOT
AORB
AB
AXORB

The result from the specified logical operation can be transferred to either the A or the B register. For example:

```
A=AORB
B=AORB
```

4.1.3 Register Group

Besides the A and B registers, the transfer register (TR) is the only other register in the basic PDP16-M that can be used for temporary data storage. The register is byte and word addressable to facilitate data manipulation. The following instructions can be used to transfer data between the TR and A registers:

```
TR=0
TR=A        A=TR
TRU=A       A=TRU
TRL=A       A=TRL
```

The instructions specifying lower and upper byte transfers between the TR and the A register (TRL and TRU) transfer only the specified 8-bit byte between the respective 8-bit locations of the registers. The remaining 8 bits of the A register will be set to logic 0 during a byte transfer to the A register, and the remaining 8 bits of the TR will remain unchanged during a byte transfer to the TR. However, the contents of the source register will remain unchanged. Data cannot be transferred between the B and TR registers.

4.1.4 Constant Generator Group

Constants required by the program can be generated by the program or wired in the constant generator (C). If constants need to be changed from time to time it is desirable to wire them in the constant generator. This avoids having to change, reassemble, and reload the program. Up to four constants can be wired in the constant generator that is part of the basic PDP16-M. The following instructions have been implemented to transfer the constants from the constant generator to the B register:

```
B=C1
B=C2
B=C3
B=C4
```

The constants cannot be transferred directly to the A register.

4.1.5 I/O Group

The basic PDP16-M has one 16-bit parallel I/O channel (GPI1), six Boolean input channels (EXT1–6), and three Boolean output channels (FF1, FF2 and FF3). The Boolean output channels can also serve as program flags if not used as output channels since they can be tested using the IF instruction. The following instructions control the Boolean output channels/flags:

To Reset	To Set
FF1=0	FF1=1
FF2=0	FF2=1
FF3=0	FF3=1

The Boolean output channels can drive 8 TTL unit loads. A logic 1 (high) Boolean output is produced by setting the flip-flop. For example:

```
FF1=1
```

The Boolean input channels can be tested for the presence of a 1 or 0 TTL logic level using the IF instruction. For example:

```
IF EXT1,LABEL
```

If a logic 1 is sensed, the program will jump to the instruction identified by the label. A 0 logic level at the EXT1 input channel causes the instruction following the IF instruction to be executed. The 16-bit parallel I/O channel can be used in a variety of ways. The 16 bits can represent individual digital I/O points, a set point value, or a desired combination of address and data for controlling I/O multiplexers. The A and B registers serve as the source and destination registers for GPI1. The following instructions have been implemented to transfer output data to the GPI1:

```
GPI1=A
GPI1=B
```

Input data from the GPI1 can be transferred to the A or B registers using the following instructions:

```
A=GPI1
B=GPI1
```

The Boolean outputs (FF1–FF3) and the Boolean inputs (EXT 1 – EXT 6) can be used to synchronize and/or interlock the data transfers between the PDP16-M and external devices.

4.1.6 Command Group

The five control commands in the PDP16-M instruction set are:

```
PAGE0
PAGE1
MUX0
MUX1
HALT
```

Two pages of control memory, each containing 512 words (two control PROMs), can be implemented in the PDP16-M. The PAGE commands are needed to switch from one page to the other because only 512 words can be directly addressed. When the second multiplexer is implemented, the MUX commands are needed to switch from one multiplexer to the other. Each multiplexer offers 30 conditions that can be tested using the IF instruction. The HALT command causes the program to stop. The console START switch can be pressed to restart the program at location 0 of the page where the machine stopped. Depressing the START switch does not produce a power clear. (The power clear signal is generated during power up to reset all data and control registers.) Therefore, when the machine is restarted, all data and control registers will be in the state they were in when the HALT command was executed. That is, if the machine stopped in page 1 with multiplexer 1 selected and FF1 set, the program will continue starting with the instruction in location 0 of page 1 with multiplexer 1 selected and FF1 set.

If the START switch is depressed while the machine is running it may hang up unless the key switch is in the PANEL LOCK position. With the key switch in this position, the START switch is disabled. The program can also be restarted at the location following the HALT command by asserting the CONTINUE signal. The CONTINUE signal is made available at the MUX and FF I/O slot.

When any of the above five instructions are executed, the bus is automatically zeroed and the IF DZ,LABEL instruction will always be true.

4.1.7 Test Group

Every time a register transfer instruction is executed, an automatic test for positive, negative, or zero data (DP, DN, DZ) is made. The result from this test is retained until the next register transfer instruction is executed. Sometimes it is helpful to find out whether the previous data transfer was positive, negative, or zero. For this reason, two examine instructions have been implemented:

```
        EXA
        EXB
```

These two instructions can be used to retest the contents of the A and B registers prior to a conditional jump instruction. For example:

```
        A=A+B
        MUX0
        EXA
        IF DP,SEND
        :
SEND    GPI=A
```

4.2 GOTO INSTRUCTION

The GOTO instruction provides the user with a convenient way to transfer control from one part of his program to another. Control can be transferred unconditionally using the following statement:

```
        GOTO ADD
        .
        .
        .
ADD     LET A=A+B
```

This statement, which references a statement label, can be used anywhere in the program to transfer control to another part of the program within the same page. All labels must start with an alphabetic character (A–Z) and end with a delimiter. A space, rubout, tab, or a comma can be used as the delimiter. A label may contain from one to ten characters. To transfer control to a label that appears on the other page, linkage code must be written (Paragraph 4.5). Control can also be transferred unconditionally using the following statement.

```
GOTO ADD'-3
```

This statement, which references a label and a signed number preceded by an apostrophe, transfers control to the third memory location prior to the ADD label. Control can be transferred to a statement before or after the label by changing the sign of the number. This method of transferring control is useful when the coder runs out of symbol table space for labels.

The following statement can also be used for transferring control unconditionally:

```
GOTO .'+5
```

This statement, which references a period instead of a label and a signed number preceded by an apostrophe, transfers control to the fifth memory location after the GOTO instruction. Again the signed number can be positive or negative depending on which way the jump is to occur.

4.3 IF INSTRUCTION

The IF instruction is a conditional jump instruction. There are 21 hardwired conditions that can be tested in the basic PDP16-M. These conditions are:

Condition	Remarks
DZ, DP, DN	Data Word Sign (zero, pos, neg) of Last Data Transfer
OVF	Overflow
A<1,3,5,...15>	Odd A Register Bits
FF1, 2, 3	Boolean Outputs/Flags
EXT1,2,3,...6	Boolean Inputs
CLK	Clock

A conditional jump is specified by the following IF instruction:

```
          A=TR
          IF DP,ADD
          .
          .
          .
ADD       A=A+B
```

After the A register gets the contents of TR, the IF instruction causes a test of DP. If the data transferred was positive, the program will continue with the statement labeled ADD. This label cannot appear on the other page. If the data was not positive, the instruction following the IF statement will be executed. Notice that there is a space delimiter between IF and DP and a comma delimiter between DP and ADD. This format should always be followed when writing IF statements.

Bit 03 of the A register is tested using the following statement:

```
          IF A<3>,START
          .
          .
          .
START     GPI1=B
```

If bit 03 of the A register is set, the instruction labeled START is executed. If the bit is reset, the instruction following the IF instruction is executed. Notice that the bit to be tested is enclosed in left and right angle brackets. This format should always be used in writing IF statements for testing the bits of the A register. The remaining conditions declared above can also be tested by writing an IF statement as described. For example:

```
IF OVF,ONE
IF FF1,TWO
IF EXT1,THREE
IF CLK,FOUR
```

Remember that a space delimiter separates the word IF and the condition to be tested and the comma delimiter separates the condition to be tested and the statement label.

To transfer control to a label that appears on the other page, linkage code must be written(Paragraph 4.5).

The feature for specifying a jump location relative to the label or the current location, described for the GOTO instruction, also applies to the IF instruction. For example:

```
          A=TR
          IF DP,.'+5
          IF DN,ADD'-2
          .
          .
ADD       A=A+B
          .
          .
```

The first IF instruction will transfer control to the fifth memory location from its own location if the test is true. The second IF instruction transfers control to the second memory location before the label ADD. Again this feature permits the coder to minimize the number of labels in his program. When limited core storage (no more than 4K) is available in the PDP-8/E used for assembly, it is desirable to minimize the number of labels because there is room for only approximately 100 labels in the symbol table.

4.4 CALL AND EXIT INSTRUCTIONS

Repetitive functions can be coded as separate subroutines and called into operation when needed by the main program. The call statement brings the subroutine (on the current page) into operation and the EXIT statement causes a return to the main program. For example:

```
          CALL SUB1
          .
          .
          .
SUB1      A=GPI1
          B=C1
          A=AXORB
          IF DZ,STOP
          EXIT
STOP      HALT
```

Notice that a space delimits the subroutine label and the word call. This format must be used in writing CALL statements. This subroutine transfers a data word from the GPI1 to the A register.

The data is then compared with the constant stored in C1. If the data exhibits the same bit pattern as the constant, the subroutine will initiate a halt, otherwise it will return control to the main program. Whenever a subroutine is called, the return address is stored in a hardware stack. The stack can keep track of 16 return addresses; therefore, up to 16 subroutines can be called before returning to the main program. To call a subroutine that appears on the other page, page linkage code must be written (Paragraph 4.5.2).

4.5 ASSEMBLER DIRECTIVES

Besides the basic instruction set there are several assembler directives that have been implemented for the PDP16-M. These directives are useful for commenting the source program, for placing the page linkage code, segmenting the source tape, and writing the assembler initialization code. Assembler directives do not cause object code to be generated – they only control the operation of the assembler.

4.5.1 Comments

It is a good practice to comment the source program. Comprehensive comments make it easier to read and understand the program. Comments can appear anywhere in the source program. Typically, comments should be placed at the beginning of a routine to relate its function and after the individual program statements of the routine. The slash (/) is used to denote a comment follows. For example:

```
/N BIT RIGHT SHIFT
/
/LOAD A WITH BIT COUNT 0 TO 5
/LOAD B WITH DATA TO BE SHIFTED
/
SHIFTR    EXA              /TEST A (NOT NECESSARY IF JUST LOADED)
SHAGN     IF DZ,SHEND      /EXIT ROUTINE IF A=0
          B=B/2            /SHIFT DATA RIGHT 1 BIT
          A=A-1            /DECREMENT COUNT
          GOTO SHAGN       /REPEAT
SHEND     EXIT             /RETURN
          $
```

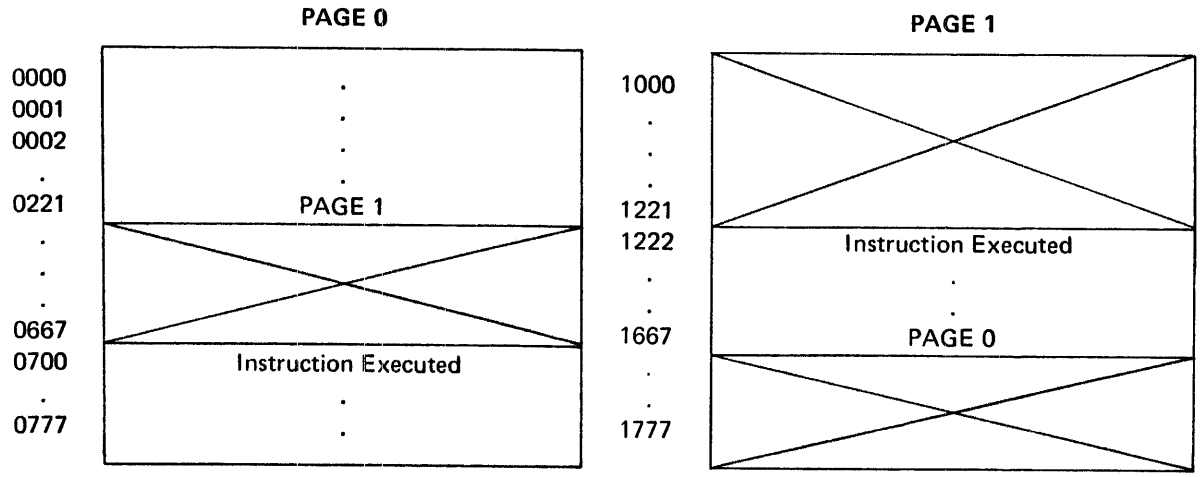
Notice that a comment does not have to follow the slash. Placing a slash at the left margin without any comment is simply a way to separate various program segments for readability.

4.5.2 Page Linkage Code

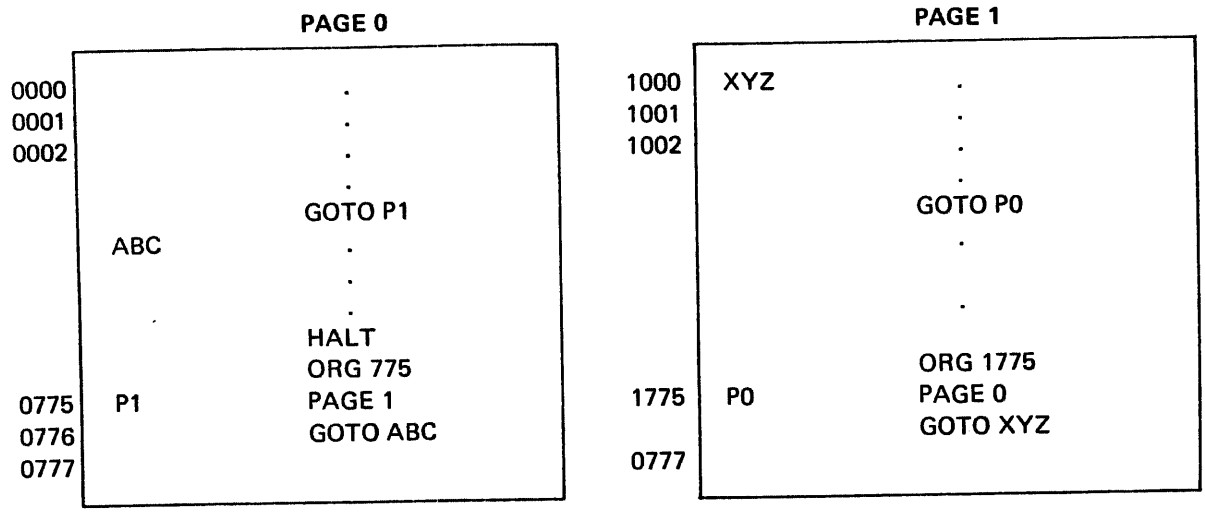
A jump (GOTO, IF, or CALL) into another page cannot be made directly because only 512 locations (the number of locations in one page) can be addressed directly. This is because only a 9-bit jump address (8-bit jump address word and the M-bit of the operation code) is stored in the control PROM. (Refer to Chapter 3 of *PDP16-M Maintenance Manual.*) The jump into another page must be made via source page linkage code. The following instructions are used to switch pages:

```
PAGE0
PAGE1
```

When either of these instructions are executed, the next sequential location in the specified page is executed. For example:



Using the PAGE instruction in the manner illustrated above creates holes in the page from which the switch was made. Holes such as these can be avoided by strategically placing the PAGE instructions or the source code. The ORGn (where n is an octal number from 0 to 1777) allows the coder to place the source and linkage code wherever he chooses. The ORGn statement causes the PC to be preset to the specified location (n) and causes all subsequent instructions to be placed in sequential location after the preset locations. For example:



Pages can be switched in the middle of a page. However, if this is done the coder must keep track of the number of instructions (locations being utilized in a given page) and then use the ORGn statement to preset the PC to the first location of each hole in order to utilize all the PROM storage space. Therefore, it is recommended that all page linkages be placed at the end of the two pages. If a given page is not completely filled with program statements the ORGn directive can be used to skip over the empty locations to maximize the storage space in the other page. The PAGE instruction must be labeled so that the GOTO or CALL instruction can reference the PAGE instructions. Also, the next location on the other page must contain the GOTO or CALL instruction to the desired label as applicable. To solve any instruction alignment problem, instructions EXA and EXB or the ORG directive can be used as no-operation instructions.

4.5.3 Source Program Segmentation

The Symbolic Editor occupies approximately 1000 locations of core in a PDP-8 and leaves all but the last page of core for the source program – allowing, in a 4K core system, for approximately 60 lines of heavily commented text or approximately 340 lines of text without comments (approximately 4200_{10} characters). The source program is stored in the text buffer area of core. When the text buffer is full, the Symbolic Editor rings the teletype bell. At this time the text buffer can be enlarged (*Introduction to Programming, 1972*) or the contents of the buffer can be punched on paper tape (Chapter 5). However, to satisfy the PDP16-M assembler a PAUSE or a $ sign directive must terminate each source program segment that is to be punched. The PAUSE directive tells the assembler during assembly that there is more to come and the $ sign directive notifies the assembler that the last tape has been read in. After the Symbolic Editor rings the bell, up to 200 additional characters can be added to the text buffer (Chapter 5). The last statement of the program segment to be punched must be the PAUSE directive or the $ sign directive.

4.5.4 Assembler Initialization Code

The assembler occupies approximately 2000 locations of core in a PDP-8/E. After the assembler is initialized with all PDP16-M instructions there is room left in the symbol table for approximately 100 program labels. After loading, the assembler is initialized by reading the definition tape. This tape is an ASCII tape prepared and punched using the

Symbolic Editor. The tape specifies all the PDP16-M instruction and condition (MUX0 and MUX1) mnemonics and the corresponding octal machine codes. (Refer to Appendix F.) The following assembler directives are used to define the mnemonics and the associated machine codes.

Directive	Remarks
INIT	Delete All Symbols
DI	Define Instruction
DC	Define Condition
FIX	Add Instruction and Condition Symbols Defined Above
PAUSE	Halt Reading Tape

A partial printout of the definition tape follows:

Mnemonic	Assembler Directive	Machine Code (octal)
	INIT	
A=0	DI	000
A=B	DI	001
A=A+1	DI	002
A=A-1	DI	003
A=A+B	DI	004
.	.	.
.	.	.
.	.	.
DATO	DI	275
TRU=A	DI	276
TRL=A	DI	277
EXT1	DC	01
EXT2	DC	02
EXT3	DC	03
.	.	.
.	.	.
.	.	.
L	DC	33
PWOK	DC	34
GND	DC	35
	FIX	
PAUSE		

If only a limited set of instructions or conditions are going to be used in coding the program a new definition tape can be created for initializing the assembler. For example:

```
            INIT
EXA         DI          036
B=B/2       DI          032
A=A-1       DI          033
DZ          DC           07
            FIX
PAUSE
/
/N BIT RIGHT SHIFT
/
/LOAD A WITH BIT COUNT 0 TO 15
/LOAD B WITH DATA TO BE SHIFTED
/
SHIFTR      EXA         /TEST A (NOT NECESSARY IF JUST
                        LOADED)
SHAGN       IF DZ,SHEND /EXIT ROUTINE IF A=0
            B=B/2       /SHIFT DATA RIGHT ONE BIT
            A=A-1       /DECREMENT COUNT
            GOTO SHAGN  /REPEAT
SHEND       EXIT        /RETURN
            $
```

This will create more space for program labels. In creating a new definition tape the user can define his own instruction and condition mnemonics. For example, the A=A+1 mnemonic can be defined as INCA (increment A). Or, the user can define the EXT1 Boolean input as CONTROL1. This feature allows the user to redefine all the instruction and condition mnemonics to simplify his programming task.

4.6 INSTRUCTION IMPLEMENTED THRU OPTIONS

The following options will extend the power and the instruction set of the basic PDP16-M.

Option	Function
MS16-C	Scratch Pad (SP)
MR16-D	Constant Generator (K)
MR16-E	Data PROM (RMAR–ROM)
MS16-D or E	Data R/W Mem (MAR–MEM)
DB16-A	Parallel I/O (GPI)
DA16-F	PDP-11 Peripheral Interface
DC16-A	Serial I/O (SI)
PCS16-D	Boolean Input (EXT)
KFL16	Boolean Output (FF)
PCS16-D	Multiplexer 1 (MUX1)

4.6.1 Scratch Pad (SP) Option MS16-C

Two MS16-C options can be implemented in the PDP16-M. Each MS16-C adds 16 16-bit high-speed registers. These registers can be used as storage buffers for program generated constants, logical masks, intermediate arithmetic results, or I/O data. The following instructions have been implemented to transfer data between the scratch pad registers and the A register:

Scratch Pad 1		Scratch Pad 2	
SP1=A	A=SP1	SP17=A	A=SP17
SP2=A	A=SP2	SP18=A	A=SP18
SP3=A	A=SP3	SP19=A	A=SP19
SP4=A	A=SP4	SP20=A	A=SP20
SP5=A	A=SP5	SP21=A	A=SP21
SP6=A	A=SP6	SP22=A	A=SP22
SP7=A	A=SP7	SP23=A	A=SP23
SP8=A	A=SP8	SP24=A	A=SP24
SP9=A	A=SP9	SP25=A	A=SP25
SP10=A	A=SP10	SP26=A	A=SP26
SP11=A	A=SP11	SP27=A	A=SP27
SP12=A	A=SP12	SP28=A	A=SP28
SP13=A	A=SP13	SP29=A	A=SP29
SP14=A	A=SP14	SP30=A	A=SP30
SP15=A	A=SP15	SP31=A	A=SP31
SP16=A	A=SP16	SP32=A	A=SP32

Data cannot be transferred directly between the scratch pad registers and the B register.

4.6.2 Constant Generator (K) Option MR16-D

One MR16-D option can be implemented in the PDP16-M. The option adds 24 16-bit read-only memory (ROM) locations for program constants. Each constant is set by stringing a single wire through and around a set of 16 ferrite cores. The following instructions have been implemented to transfer the constants from the constant generator to the B register:

B=K1	B=K9	B=K17
B=K2	B=K10	B=K18
B=K3	B=K11	B=K19
B=K4	B=K12	B=K20
B=K5	B=K13	B=K21
B=K6	B=K14	B=K22
B=K7	B=K15	B=K23
B=K8	B=K16	B=K24

The constants cannot be transferred directly to the A register.

4.6.3 Data PROM Option MR16-E

One or two MR16-E Data PROM options can be implemented in the PDP16-M. The options add 256 8-bit programmable read-only memory (PROM) locations for program constants, text information (ASCII characters – see Appendix B), code conversion tables, or matrix elements. The Data PROMs are identical to the

control PROM. Preparation of the source code and the object code, as well as the loading procedure for the PROM, are identical to that required for the control PROMs. (See Data PROM option MR16-E/F description in the *PDP16-M Maintenance Manual.*) The following instructions have been implemented to address the PROM and transfer the 8 or 16-bit data to the A or B register:

Load Address	Transfer Data
RMAR=A	A=ROM
RMAR=B	B=ROM

If only one PROM is to be implemented, it can be set up so that when the register transfer instruction (A=ROM or B=ROM) is executed, the data is transferred to the eight high-order bits or the eight low-order bits of the specified destination register. (See Data PROM option MR16-E/F description in the *PDP16-M Maintenance Manual.*) If the Data PROM is implemented to transfer its data into the eight low-order bits (0—7) of the destination register, the eight high-order bits will always contain 1s and vice versa (Figure 4-1). When both Data PROM options are implemented, a full 16-bit data word is transferred to the specified destination register in response to the register transfer instruction. In any case, since only 256 PROM locations will be available, only an 8-bit address (0—377) is required to address every location in the PROM(s). Arithmetic register transfer instructions can be used to generate a specific address or the base address for the constants, messages, tables, or matrices stored in the PROM. (Refer to example for generating constants in Paragraph 4.1.1.) The same procedure can be used for generating the desired address for the Data PROM. These addresses can be stored in the SP registers for future reference. Addresses and/or base addresses for these items can also be set into one of the constant generators (C or K). They can then be transferred from C or K to the RMAR register via the B register. For example:

```
B=C1          /GET ADDRESS
RMAR=B        /LOAD ADDRESS
A=ROM         /GET DATA
A=ANOT        /COMPLEMENT DATA
```

The data stored in the PROM is loaded in complement form by the MR16-SL Utility Interface Assembly. Therefore, a complement instruction must be used or the ASCII source tape must contain the data in complement form in order to get the desired data.

4.6.4 Data Read/Write Memory Option MS16-D and E

One or two MS16 Data Read/Write Memory options can be implemented in the PDP16-M in any desired combination. The MS16-D option provides 256 16-bit words of storage and the MS16-E option provides 1024 16-bit words of storage. Whatever the implemented combination, one option is designated MEM1 (slot A6) and the other is designated MEM2 (slot A7). The address buffers for the two memories are MAR1 and MAR2, respectively. The following is a listing of all normal combinations that can be implemented:

MEM1 (slot A6)	MEM2 (slot A7)	Total Memory (words)
MS16-D	—	256
MS16-D	MS16-D	512
MS16-E	—	1024
MS16-E	MS16-D	1280
MS16-E	MS16-E	2048

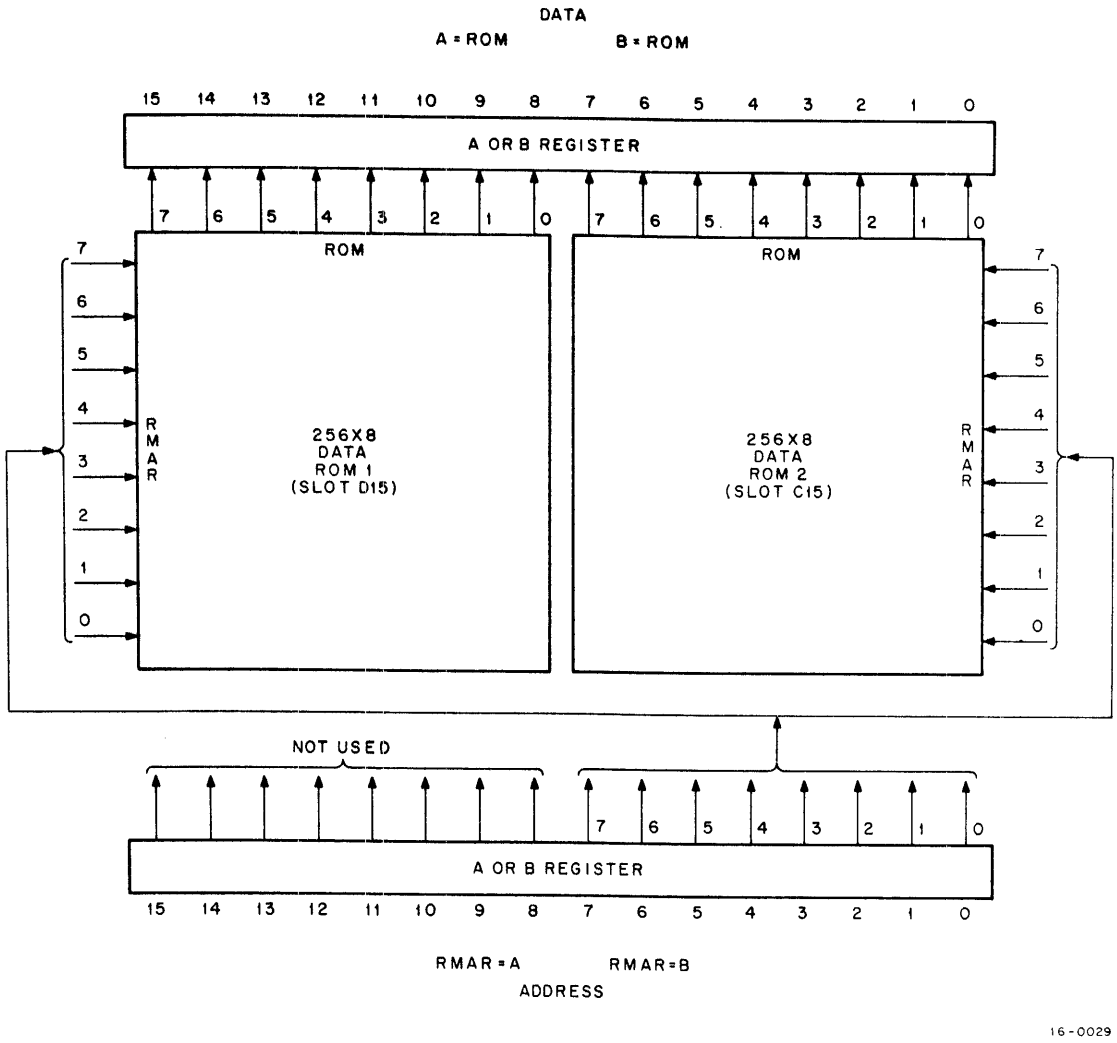


Figure 4-1 Address and Data Transfer Scheme

These memory options are useful when a requirement exists for accumulating computed data for subsequent output or for accumulating input data. Both parallel and serial data channels are available for transferring the data. The following instructions have been implemented to address the memories and to transfer the data between the memories and the A register:

Load Address	**Transfer Data**	
	IN	**OUT**
MAR1=A	MEM1=A	A=MEM1
MAR2=A	MEM2=A	A=MEM2

The B register cannot be used to address the memory to transfer the data.

Arithmetic register transfer instructions can be used to generate a specific address, the base address, and other parameters for storing and/or retrieving data. Address and parameters describing the location and size of the data block can be stored in the SP registers for future reference. The constant generator (C or K) can also be set up to provide address and size constants for fixed input/output data buffers.

4.6.5 Parallel I/O Option DB16-A

The basic PDP16-M is equipped with one 16-bit parallel I/O channel (Paragraph 4.1.5). Two additional parallel I/O channels can be added by implementing the DB16-A option. One DB16-A option is required for each channel. Slots C20 and D20 on the logic assembly are reserved for the parallel I/O channels. Slot C20 is assigned the mnemonic GPI2 and slot D20 is assigned the mnemonic GPI3. The following instructions have been implemented to transfer data to and from the channels via the A register:

Data Input	Data Output
A=GPI2	GPI2=A
A=GPI3	GPI3=A

The B register cannot be used as the source or destination register for these channels; however, this register can be used for this purpose with GPI1.

The Boolean outputs (FF1–FF3) and the Boolean inputs (EXT1–EXT6) can be used to synchronize and/or interlock the data transfers between the PDP16-M and external devices (Paragraph 4.1.5).

4.6.6 PDP-11 Peripheral Interface Option DA16-F

The DA16-F option must be implemented in order to interface the PDP16-M with low-speed PDP-11 peripheral devices. The following instructions are needed to communicate with the PDP-11 peripheral devices:

Instruction	Remarks
FF1=0	Prepares Output Section of Device
FF1=1	Prepares Input Section of Device
GPI1=A	Load Device Register Address
GPI1=B	Load Device Register Address
DATO	Send Contents of A Register to Device Register
DATI	Send Contents of Device Register to A Register

A thorough knowledge of the PDP-11 peripheral device is required before attempting to write routines for controlling the device. Typically, PDP-11 devices have status, control, and data registers. Each device register can be individually addressed, read, and/or loaded. Appendix A summarizes the register formats and addresses for some of the low-speed peripheral devices.

In generating addresses under program control or in installing addresses in the constant generators, the complement of the actual address must be created. This is because the GPI1 drives the address lines of the peripheral bus with TTL levels, not open-collector drivers. The data does not have to be complemented because the data lines are driven with open-collector drivers.

An example program for the UDC11 follows:

```
        :
        CALL READST
        :
READST  FF1=1           /SET CONTROL LINE C1
        B=C1            /GET STATUS REGISTER ADDRESS
        GPI1=B          /ADDRESS DEVICE STATUS REGISTER
        DATI            /READ STATUS INTO A REGISTER
        IF A <14>, SKIP /SKIP IF POWER FAIL
OUTPUT  FF1=0           /RESET CONTROL LINE C1
        A=0             /CLEAR A REGISTER
        B=C2            /GET OUTPUT MODULE ADDRESS
        GPI1=B          /ADDRESS OUTPUT MODULE
        A=A+1           /SET DATA BIT 00
        DATO            /SEND A REGISTER CONTENTS TO OUTPUT MODULE
INPUT   FF1=1           /SET CONTROL LINE C1
        B=C3            /GET INPUT MODULE ADDRESS
        GPI1=B          /ADDRESS INPUT MODULE
        DATI            /READ DATA INTO A REGISTER
        SP1=A           /STORE DATA
SKIP    EXIT            /RETURN TO MAIN PROGRAM
```

NOTE

C1 = $\overline{171776}$ = 006001
C2 = $\overline{171224}$ = 006553
C3 = $\overline{171226}$ = 006551

This program reads the UDC status word and if no power fail exists, a data word is sent to an output module and the data from an input module is read. If a power fail is sensed, the program will return control to the main program. The example program is not intended to serve any specific function other than to illustrate how to use PDP16-M instructions to program PDP-11 peripheral devices.

4.6.7 Serial I/O Option DC16-A

One or two serial interface channels can be added to a PDP16-M. They are added by implementing the DC16-A and B options. One DC16-A option is required for each channel. Slots AB16 and AB17 on the logic assembly are reserved for the serial I/O channels. Slot AB16 is assigned the mnemonic SI1 and slot AB17 is assigned the mnemonic SI2. Any TTY compatible devices such as teleprinter terminals, displays, or modems for communication lines can be connected to the serial I/O channels. The following instructions have been implemented to transfer data between the PDP16-M and the devices and the I/O channels:

Instruction	Remarks
TAPE1	Read one character from Channel 1
TAPE2	Read one character from Channel 2
IF KF1,LABEL	KF1 is set after character from Channel 1 is read and assembled
IF KF2,LABEL	KF2 is set after character from Channel 2 is read and assembled
A=SI1	Transfer character from Channel 1 (SI1) to A <0–7>
A=SI2	Transfer character from Channel 2 (SI2) to A <0–7>
SI1=A	Transfer character from A <0–7> to Channel 1 device
SI2=A	Transfer character from A <0–7> to Channel 2 device
IF PF,LABEL	PF1 is set after character is displayed, printed, or punched
IF PF,LABEL	PF2 is set after character is displayed, printed, or punched

The code required for transferring characters between the serial I/O device and the PDP16-M is illustrated below:

```
/TRANSMIT A CHARACTER
            A=SP1               /GET CHARACTER
            CALL OUTPUT         /CALL OUTPUT SUBROUTINE
            .
            .
            .
OUTPUT      SI1=A               /TRANSFER CHARACTER
L2          IF PF1,L1           /TEST PUNCH FLAG AND JUMP IF SET
            GOTO L2             /TEST FLAG AGAIN
L1          EXIT                /RETURN
/READ A CHARACTER
            CALL INPUT          /CALL INPUT SUBROUTINE
            .
            .
            .
INPUT       TAPE1               /READ A CHARACTER
M2          IF KF1,M1           /TEST KEYBOARD FLAG AND JUMP IF SET
            GOTO M2             /TEST FLAG AGAIN
M1          A=SI1               /TRANSFER CHARACTER
            EXIT                /RETURN
```

Notice that the code for inputting and outputting characters is imbedded in subroutines so that the code can be called into operation each time a character is to be transferred. A summary of ASCII character codes is given in Appendix B.

4.6.8 Boolean Output Option KFL16

Three additional Boolean output channels/program flags can be added to the PDP16-M (Paragraph 4.1.5). They are added by implementing the KFL16 option. The following instructions control the additional Boolean output channels:

To Reset	To Set
FF4=0	FF4=1
FF5=0	FF5=1
FF6=0	FF6=1

Each channel can drive up to 8 TTL unit loads. A logic 1 (high) Boolean output is produced by setting the flip-flop. For example:

```
FF4=1
```

The Boolean output channels can also serve as program flags if not used as output channels because they can be tested using the IF instruction. For example:

```
IF FF4,LABEL
IF FF5,LABEL
IF FF6,LABEL
```

These instructions cause a test of the specified flip-flop. If the flip-flop is set, the instruction with the declared label will be executed. If the flip-flop is not set, the next sequential instruction is executed. The flip-flops are set or reset under program control to specify that a particular condition was satisfied (program flag).

4.6.9 Boolean Input Option PCS16-D

Sixteen additional Boolean input channels can be added to the PDP16-M (Paragraph 4.1.5) by implementing the PCS16-D option. The following instruction provides the means for testing the state (logic 1 or 0) of each input channel:

```
IF EXTn,LABEL
```

where n=the numeral 7 through 22.

If a logic 1 is sensed, the program will jump to the instruction identified by the label. A 0 logic level at the EXTn input channel causes the instruction following the IF instruction to be executed.

4.6.10 IF Instruction Option PCS16-D

Besides adding the sixteen additional Boolean inputs (EXT1–22), the PCS16-D option adds 12 additional hardwired conditions that can be tested with the IF conditional jump instruction (Paragraph 4.3). These conditions are:

Conditions	Remarks
L	One Bit LINK Register
A <0,2,4,...14>	Even Bits of the A Register
B<0>	LSB of B Register
B<15>	MSB of B Register
PWOK	AC Power OK

4.7 SAMPLE PROGRAMS

To further illustrate the simplicity of coding algorithms using PAL16, the following additional general purpose subroutines are presented:

a. Reading Paper Tape from ASR 33

b. Print message on ASR 33

c. Multiply two 16-bit numbers

4.7.1 Read Paper Tape

The following subroutine will read a paper tape from the Model ASR 33 Teletype®. The ASR 33 is connected to serial I/O Channel 1 (SI1).

```
/          READING PAPER TAPE FROM ASR 33
/
/             IGNORE BLANK TAPE LEADER. READ PAPER TAPE
/             AND LOAD IT INTO SEQUENTIAL LOCATIONS IN A
/             256X16 MEMORY. START IN LOC. 0 AND STOP
/             WHEN YOU REACH BLANK TAPE TRAILER.
/
READ       B=0
           A=0
           MAR1=A
READT      CALL PTR            /READ CHARACTER
           IF DZ,READT         /JUMP IF BLANK TAPE
READN      MEM1=A
           A=B
           A=A+1
           B=A
           MAR1=A
           CALL PTR
           IF DP,READN
           EXIT
/
PTR        TAPE1
PTRW       IF KF1,PTRR
           GOTO PTRW
PTRR       A=SI1
           EXIT
```

4.7.2 Print Message on ASR 33

The following subroutine will print a message on the ASR 33 keyboard. The ASR 33 is connected to serial I/O Channel 1 (SI1).

```
/          PRINT MESSAGE ON ASR 33
/
/             ASCII CODES ARE STORED IN 256X8 PROM.
/             END OF MESSAGE IS CODE 377
/
/          B REGISTER HAS START ADDRESS OF TEXT
/
TEXT       RMAR=B              /INITIALIZE PROM ADDRESS
           A=ROM               /FETCH CODE
           A=ANOT              /COMPLEMENT
TEXTW      IF PF1,TEXTP        /WAIT FOR TTY
           GOTO TEXTW
TEXTP      SI1=A               /PRINT CHARACTER
           A=A+1               /INCREMENT
           A=A/2               /SHIFT RIGHT
           IF A<7>,TEXTC       /TEST FOR END OF MESSAGE CODE
           A=B
           B=A+1               /UPDATE TEXT POINTER
           GOTO TEXTW
TEXTC      EXIT
```

® Teletype is a registered trademark of the Teletype Corporation.

4.7.3 Multiply

The following subroutine will multiply two 16-bit signed numbers and yield a 32-bit result.

```
/           PDP16 MULTIPLY ROUTINE
/
/           TWO 16 BIT SIGNED NUMBERS YIELD A 32 BIT RESULT
/
/             SP2=NUMBER 1
/             B=NUMBER 2
/           THE B REGISTER WILL REMAIN UNCHANGED.
/
/
MULT        A=0                 /CLEAR A REG
            SP1=A               /CLEAR SP1
            L=1                 /GENERATE SHIFT COUNT = 15
            A=AX2
            A=AX2
            A=AX2
            A=AX2
            SP3=A               /STORE SHIFT COUNT
/
/           START MULTIPLY
/
            A=SP2               /LOAD NUMBER 1
            A=A/2(S)            /FETCH LSB IN OVF
            SP2=A
MPCON       A=SP1               /LOAD RESULT UPPER HALF
            IF OVF,MPADD        /JUMP IF LSB WAS A 1
MPSHF       A=AX2(S)            /DOUBLE PRECISION ARITHMETIC SHIFT
            L=OVF               /SET L = TO SIGN
            A=A/2
            A=A/2(S)            /SHIFT AND SAVE OVERFLOW
            L=OVF
            SP1=A               /STORE RESULT UPPER
            A=SP2               /LOAD RESULT LOWER
            A=A/2(S)            /SHIFT AND SAVE LSB
            SP2=A               /STORE RESULT LOWER
            A=SP3               /LOAD SHIFT COUNT
            A=A-1               /DECREMENT COUNT
            SP3=A               /SAVE NEW COUNT
            IF DP,MPCON         /REPEAT 15 TIMES
            IF DZ,MPSIGN        /JUMP IF SIGN BIT
            EXIT
/
MPADD       A=A+B               /ADD TO RESULT UPPER
                                GOTO MPSHF
MPSIGN      IF OVF,MPSUB        /JUMP IF NEG SIGN
            GOTO MPCON
/
MPSUB       A=SP1               /LOAD RESULT UPPER
            A=A-B               /SUBTRACT FROM RESULT UPPER
            GOTO MPSHF          /GO TO SHIFT
```

CHAPTER 5
PROGRAM PREPARATION AND ASSEMBLY

The PDP16-M control program (or data arrays for the data PROM) is prepared and assembled on a PDP-8/E. The source paper tape, which is required for assembly, is prepared under the control of the Symbolic Editor. The source tape is read and translated by the PAL16 Assembler. After successful assembly the object tape can be punched. The object tape contains the machine code of the control program that is loaded into the PDP16-M control PROM.

The PDP-8/E to be used for program preparation and assembly need only be equipped with 4K of core and an ASR 33. The ASR 33 contains a low-speed reader and a low-speed punch.

Before either the Symbolic Editor or the PAL16 Assembler can be loaded, the Binary Loader must be in core.

5.1 BINARY LOADER

The Binary Loader (BIN) is a short utility program which, when in core, instructs the computer to read binary-coded data punched on paper tape and store it in core memory. This loader is used to load the Symbolic Editor and PAL16 Assembler. It is also used to load the utility program (Chapter 6).

The Binary Loader is supplied to the user on punched paper tape in RIM-coded format. This tape is loaded into core by the RIM Loader (Table 5-1).

There are two RIM Loaders: one for the low-speed reader (LSR) and another for the high-speed reader (HSR). The appropriate RIM Loader is toggled in core with the computer switch register as detailed in Figure 5-1.

Table 5-1
RIM Loader Programs

Location	Instruction	
	Low-Speed Reader	High-Speed Reader
7756	6032	6014
7757	6031	6011
7760	5357	5357
7761	6036	6016
7762	7106	7106
7763	7006	7006
7764	7510	7510
7765	5357	5374
7766	7006	7006

Table 5-1 (Cont)
RIM Loader Programs

Location	Instruction	
	Low-Speed Reader	High-Speed Reader
7767	6031	6011
7770	5367	5367
7771	6034	6016
7772	7420	7420
7773	3776	3776
7774	3376	3376
7775	5356	5357
7776	0000	0000

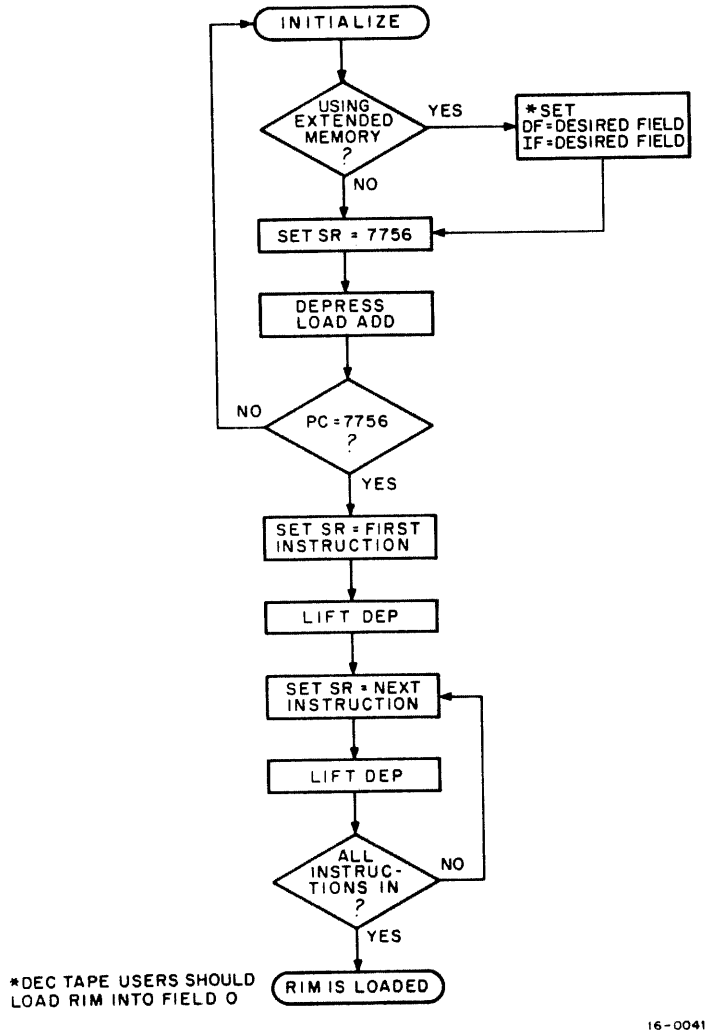


Figure 5-1 Loading the RIM Loader

After RIM is loaded, it is a good practice to verify that all instructions were stored properly. This can be done by performing the procedure illustrated in Figure 5-2. The flowchart also shows how to correct an incorrectly stored instruction. With the correct RIM Loader in core, the Binary Loader paper tape can be read and stored in core by following the procedure illustrated in Figure 5-3. After the Binary Loader paper tape is successfully read, the loader resides in the last page of core, occupying absolute locations 7625 through 7752 and 7777.

The Binary Loader was purposely placed on the last page of core so that it would always be available for use – the Symbolic Editor and the PAL16 Assembler do not use the last page of core. To load either the Symbolic Editor or the PAL16 Assembler binary paper tapes perform the procedure illustrated in Figure 5-4. Since the Binary Loader remains in the last page of core, unaffected by the loading procedure, it can be used again using the same procedure to switch between the Symbolic Editor and the assembler, or vice versa.

5.2 SYMBOLIC EDITOR

The Symbolic Editor is a service program which allows the programmer to write and prepare symbolic programs and to generate a symbolic program tape of his programs. Editor is very flexible in that the programmer can type his symbolic program on-line from the teletype keyboard, thus storing it directly into core memory. Then, using certain Editor commands, the programmer can have his program listed (printed) on the teleprinter for visual inspection.

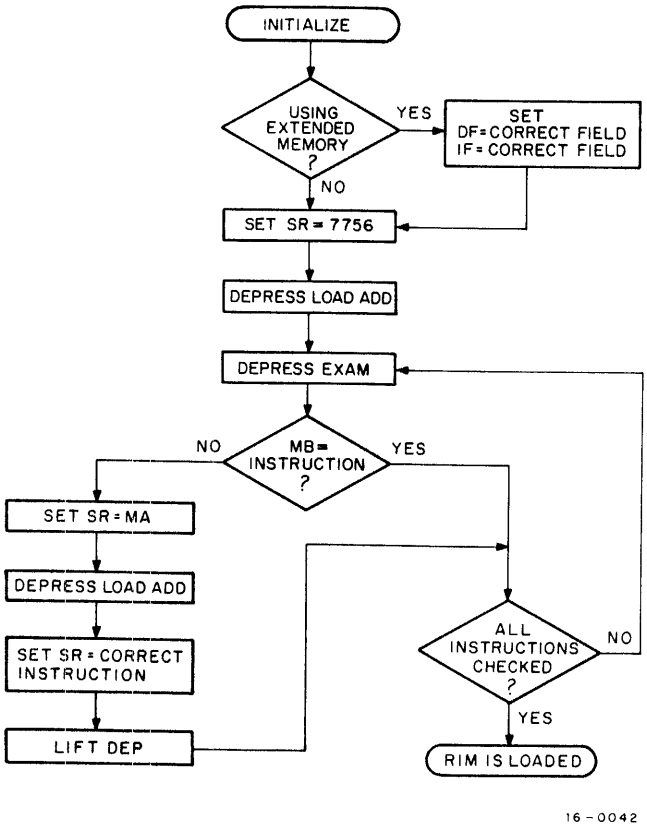


Figure 5-2 Checking the RIM Loader

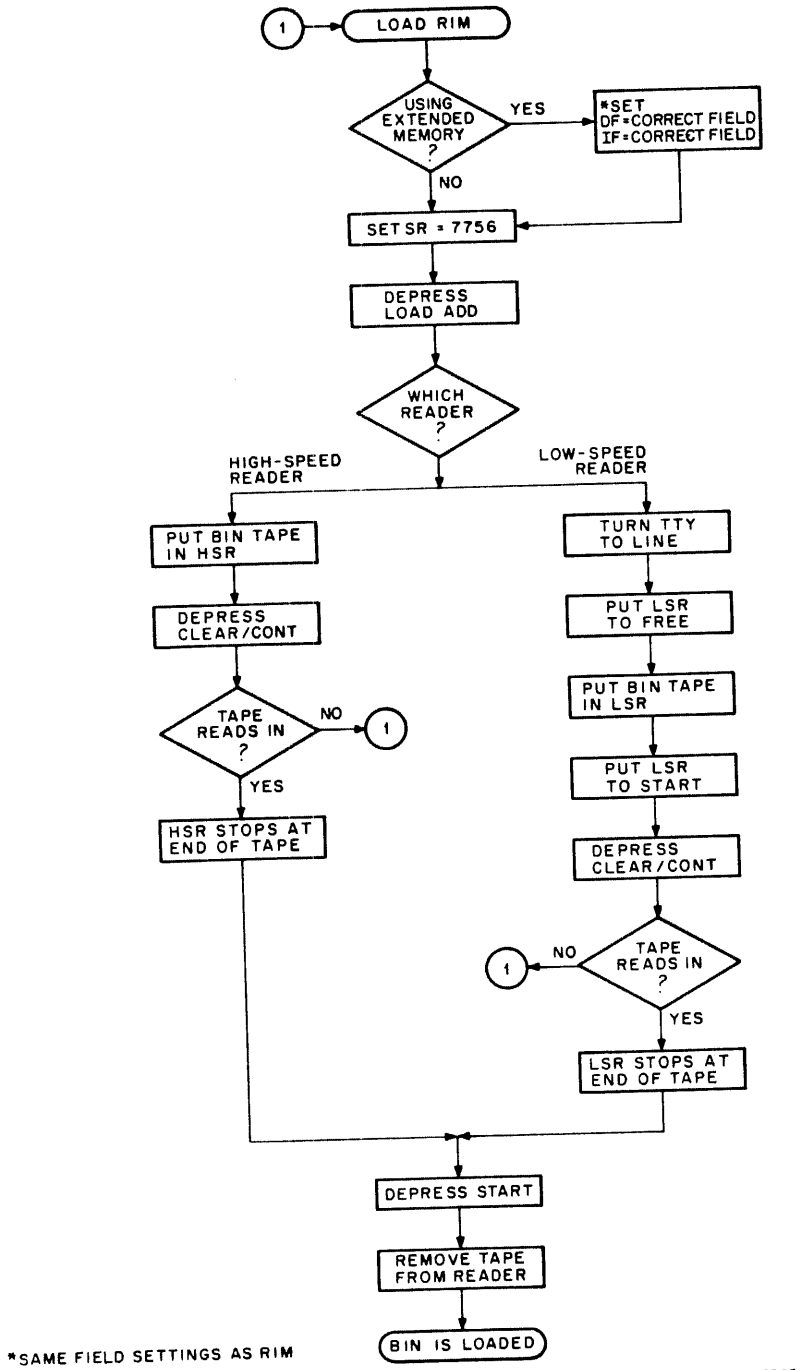


Figure 5-3 Loading the BIN Loader

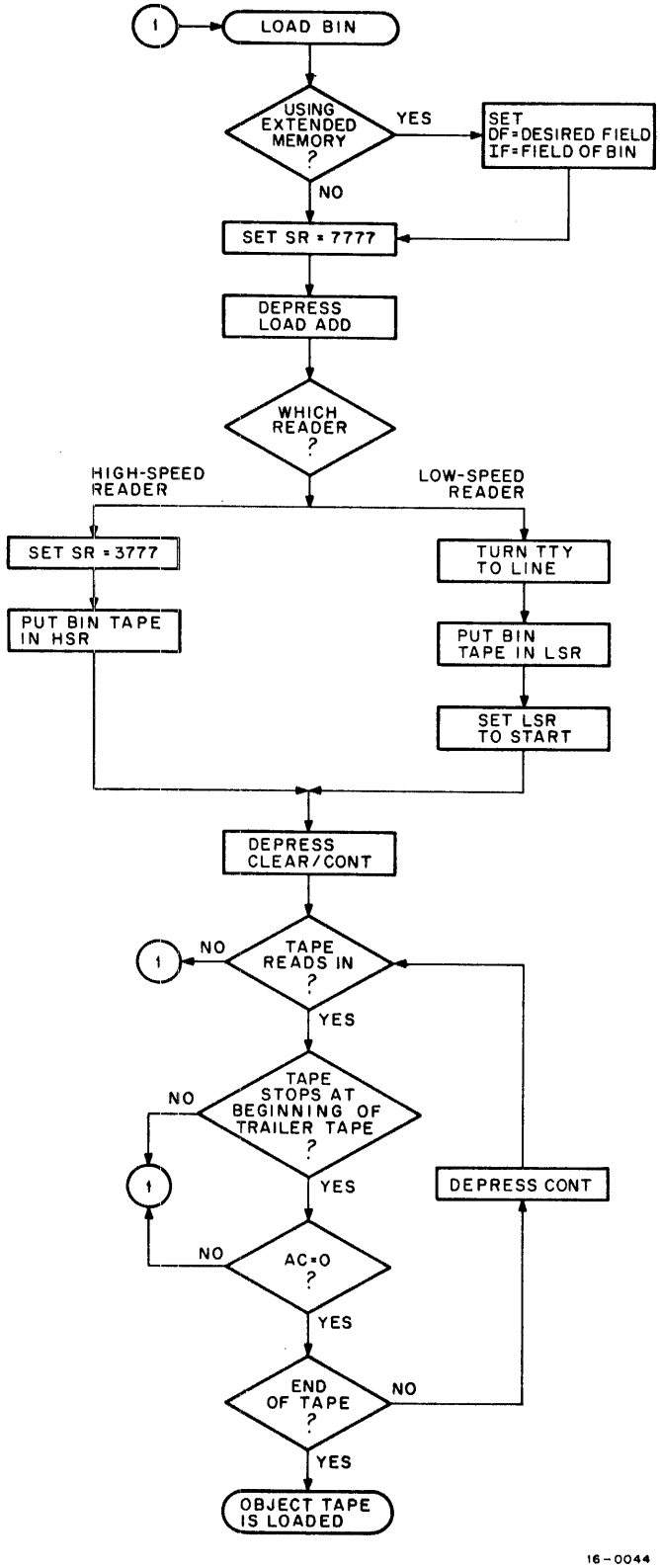


Figure 5-4 Loading A Binary Tape Using BIN

Editor also allows the programmer to add, correct, or delete any portion of his symbolic program. When the programmer is satisfied that his program is correct and ready to be assembled, Editor can be commanded to generate a symbolic program tape of the stored program.

The Symbolic Editor program is issued on punched paper tape in binary-coded format. Therefore, it is loaded into core memory using the BIN Loader. When in core, Editor is activated by setting the switch register (SR) to 0200 (the starting address) and depressing the LOAD ADD (load address switch) and then the START switch, Editor responds with a carriage return/line feed sequence on the teletype.

Initially, Editor is in command mode, (that is, it is ready to accept commands from the programmer); anything typed by the programmer is interpreted as a command to Editor. Editor accepts only legal commands, and if the programmer types something else, Editor ignores the command and types a question mark (?).

When not in command mode, Editor is in text mode; that is, all characters typed from the keyboard or tapes read in on the tape reader are interpreted as text to be put into the text buffer in the manner specified by a preceding Editor command. Figure 5-5 illustrates how the programmer can transfer Editor from one mode to another.

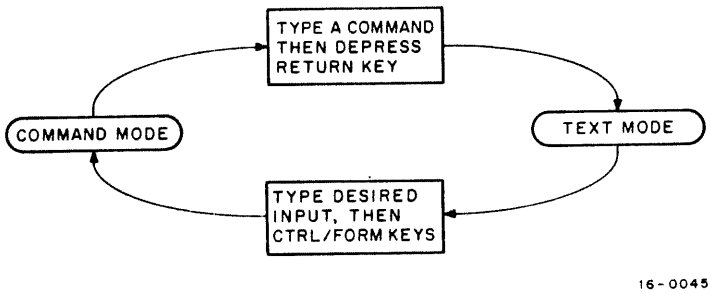


Figure 5-5 Transition Between Editor Modes

Seven of Editor's basic commands are briefly described below:

COMMAND	MEANING
A	Append incoming text from the keyboard into the text buffer immediately following the text currently stored in the buffer.
R	Read incoming text from the tape reader and append it to the text currently stored in the buffer.
L	List entire text buffer; the programmer can specify one line or a group of lines.
C	Change a line; the programmer precedes the command with the decimal line number or line numbers of the lines to be changed.
I	Insert into text buffer; the programmer specifies the decimal line number in his program where the inserted text is to begin.
D	Delete from text buffer; the programmer specifies the line or group of lines to be deleted.
P	Punch text buffer; the programmer can specify one line, a group of lines, or the entire text buffer.

All commands are executed, except the P command, when the RETURN key is depressed. To execute the P command, press the RETURN key on the teletype, turn on the punch, and press the CONT (continue) switch on the computer console.

The above commands are only the seven basic commands. A summary of all commands is provided in Table 5-4.

5.2.1 Writing a Program

Now that you have some idea of what you can do with the Symbolic Editor and what it can do for you, we will write and edit a short program, explaining each step in the comments to the right of the printout.

The example program is a print text routine for serial I/O channel SI1. The program is written in PAL16, to be assembled using the PAL16 Assembler described later in this chapter.

The programmer loads Editor using the BIN Loader (Figure 5-4). Editor is then activated by loading the starting address (0200 – octal) and depressing the LOAD ADD, CLEAR, and CONT switches. After Editor responds with a carriage return/line feed, the programmer types A and RETURN key. Editor is now in text mode; that is, subsequent characters typed are added in the text buffer. The programmer now types the symbolic program. (Block indenting is facilitated using the CTRL/TAB key which Editor has programmed to indent in 8-character increments.)

```
A
/           TEXT PRINT ROUTINE
/
/              ASCII CODES ARE STORED IN 256X8 PROM
/              END OF MESSAGE IS CODE 377
/
/           B REGISTER HAS START ADDRESS OF TEXT
/
TEXT        RMAR=B              /INITIALIZE PROM ADDRESS
            A=ROM
            A=ROM               /FETCH CODE
            A=ANOT              /COMPLEMENT
TEXTW       IF PF1,TEXTP        /WAIT FOR TTY
            GOTO TEXTW
TEXTP       SI1=A               /PRINT CHARACTER
            A=A+1               /INCREMENT
            A=A/2               /SHIFT RIGHT
            IF A<7>,TEXTC       /TEST FOR END OF MESSAGE CODE
            A=B
            B=A+2               /UPDATE TEXT POINTER
            GOTO TEXTW
TEXTC       EXIT
            $
```

Visual inspection reveals errors in lines 9, 19, and 20 (Editor maintains a line number count in decimal, with the first line typed being 1 and the last line being 22). Line 9 can be removed using the D (delete) command, and lines 19 and 20 can be corrected using the C (change) command. However, Editor is presently in text mode, and in order to issue another command, Editor must be transferred to command mode. This is done when the programmer types CTRL/FORM (depress and hold down the CTRL key while typing the FORM key).

CTRL/FORM (nonprinting)	The programmer types CTRL/FORM; Editor responds with CR/LF and rings the teleprinter bell, indicating that it is in command mode.
9D	The programmer types 9D and the RETURN key; Editor responds with a CR/LF and the line is deleted.
18, 19C	The programmer types 18, 19C and the RETURN key, informing Editor that lines 18 and 19 (formerly 19 and 20) are to be changed. Editor responds with a CR/LF, transfers to text mode, and waits for the programmer to change the lines.
B=A+1 /UPDATE TEXT POINTER GOTO TEXT	The programmer types B=A+1 /UPDATE TEXT POINTER and GOTO TEXT

The symbolic program should now be correct. However, it is good programming practice to check the program after editing; this can be done using the L (list) command, but since only original lines 9, 19, and 20 were changed, it is not necessary to have the whole program listed. The programmer can command Editor to list lines 9 through 20.

CTRL/FORM (nonprinting)	The programmer types CTRL/FORM to return Editor to command mode; Editor responds with CR/LF, rings the bell, and waits for the next command.
9, 20L	The programmer types 9, 20L and the RETURN key; Editor types lines 9 through 20.

```
            A=ROM               /FETCH CODE
            A=ANOT              /COMPLEMENT
TEXTW       IF PF1,TEXTP        /WAIT FOR TTY
            GOTO TEXTW
TEXTP       SI1=A               /PRINT CHARACTER
            A=A+1               /INCREMENT
            A=A/2               /SHIFT RIGHT
            IF A<7>,TEXTC       /TEST FOR END OF MESSAGE
            A=B
            B=A+1               /UPDATE TEXT POINTER
            GOTO TEXT
TEXTC       EXIT
```

The changes were accepted properly. The symbolic program is correct and ready to be punched on paper tape.

5.2.2 Search Feature

A very convenient feature available with Editor is the search feature which allows the programmer to search a line of text for a specified character. When the programmer types a line number followed by S, Editor waits for the user to

type in the character for which it is to search. The search character is not echoed (printed on the teleprinter). When Editor locates and types the search character, typing stops and Editor waits for the programmer to either type new text and terminate the line with a RETURN key or to use one of the following special keys:

Special Key	Function
←	to delete the entire line to the left,
RETURN	to delete the entire line to the right,
RUBOUT	to delete from right to left one character for each RUBOUT typed (a / is echoed for each RUBOUT typed),
LINE FEED	to insert a carriage return/line feed (CR/LF) thus dividing the line into two,
CTRL/FORM	to search for the next occurrence of the search character, and/or
CTRL/BELL	to change the search character to the next character typed by the programmer.

5.2.3 Input/Output Control

Switch register options are used with input and output commands to control the reading and punching of paper tape. The options available to the programmer are shown in Table 5-2.

Table 5-2
Input/Output Control

SR Bit	Position	Function
0	0	Input text as is
	1	Convert all occurrences of 2 or more spaces to a tab
1	0	Output each tab as 8 spaces
	1	Tab is punched as tab/rubout
2	0	Output as specified
	1	Suppress output*
10	0	Low-speed punch and teleprinter
	1	High-speed punch
11	0	Low-speed reader
	1	High-speed reader

*Bit 2 allows the user to interrupt any output command and return immediately to command mode; when desired, merely set bit 2 to 1.

5.2.4 Generating a Program Tape

Before issuing the P (punch) command, Editor must be in command mode. Figure 5-6 illustrates the procedures required to generate a symbolic program tape using the Editor.

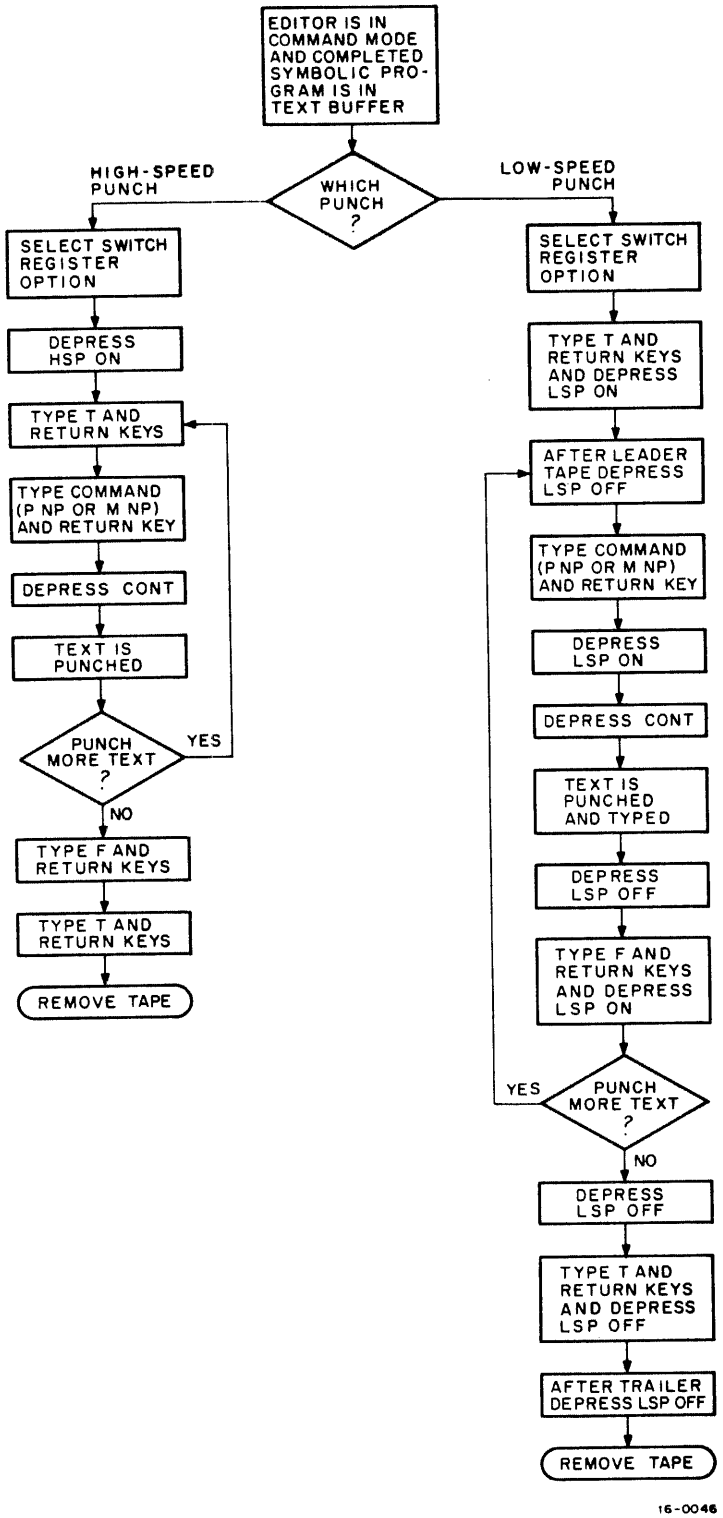


Figure 5-6 Generating a Symbolic Tape Using Editor

CTRL/FORM (nonprinting)	The programmer types CTRL/FORM.
?	Editor responds with a question mark, indicating that Editor was already in command mode.
P	The programmer commands Editor to punch the entire text buffer by typing P and the RETURN key.

When Editor recognizes a P command it waits for the programmer to specify the low- or high-speed punch. If the programmer wants the program punched and typed, he sets SR bit 10 to 0 and the program will be punched on the low-speed punch and simultaneously typed on the teleprinter. If the programmer wants only a program tape and if he has a high-speed punch available, he sets SR bit 10 to 1 and the program will be punched on the high-speed punch. For the purposes of this discussion a printed program listing is desired, so the low-speed punch is specified. The programmer turns on the low-speed punch and depresses the CONT switch on the computer console. Editor begins punching and typing the contents of the entire text buffer.

An image of the stored symbolic program has been punched and typed by Editor.

5.2.5 Loading a Program Tape

Set console switches as indicated in the section on input/output control options (Paragraph 5.2.3), depending on options desired.

Place the symbolic tape of the program to be corrected in the appropriate paper-tape reader. At the keyboard, type the READ command (R) followed by a carriage return. If using the teletype reader, turn it on now. The symbolic tape will be read into the text buffer.

The Editor will continue reading the tape until a form feed code is encountered. If the tape contains no form feed code, and the teletype reader is being used for input, type the CTRL/FORM key combination after the tape has been read in. Upon recognizing the form feed character, Editor enters the command mode and rings a bell to indicate that it is ready for the first command.

CAUTION

When using the teletype reader, if the form feed code is encountered before the symbolic tape has completely read in (as indicated by the ringing of the bell), turn off the paper-tape reader. Otherwise, characters on tape will be interpreted as commands to Editor. The section of tape read in up to the form feed code should then be edited first before proceeding with the remainder of the tape.

5.2.6 Restart Procedure

If the programmer stops the computer, for example, purposely or accidentally turning the computer off, he may restart Editor at location 0200 or 0177 without disturbing the text in the buffer. Editor can also be restarted at location 0176; however, all text currently in the buffer is wiped out. Therefore, the programmer can restart at location 0176 to re-initialize for a new program.

5.2.7 Error Detection

Editor checks all commands for nonexistent information and incorrect formatting. When an error is detected, Editor types a question mark (?) and ignores the command. However, if an argument is provided for a command that doesn't require one, the argument is ignored and the command is executed properly.

Editor does not recognize extraneous and illegal control characters; therefore, a tape containing these characters can be cleared up or corrected by merely reading the tape into Editor and punching out a new tape.

5.2.8 Summary of Special Keys and Commands

Using special keyboard keys and commands, the programmer controls Editor's operation. Certain keys have special meaning to Editor, of which some can be used in either command or text mode. The mode of operation determines the function of each key. The special keys and their functions are shown in Table 5-3.

Table 5-3
Summary of Special Keys

Key	Command Mode	Text Mode
RETURN	Execute preceding command	Enter line in text buffer
←	Cancel preceding command (Editor responds with a ? followed by a carriage return and line feed)	Cancel line to the left margin
RUBOUT	same as ←	Delete to the left character for each depression; a backslash is echoed [not used in Read (R) command]
CTRL/FORM	Respond with question mark and remain in command mode	Return to command mode and ring teleprinter bell
.(period)	Value equal to decimal value of current line (may be used alone or with + or − and a number, for example, .+8)	Legal text character
/	Value equal to number of last line in buffer; used as an argument	Legal text character
LINE FEED	List next line	Used in Search (S) command to insert CR/LF into line
ALTMODE	List next line	
>	List next line	
<	List previous line	
=	Used with . or/to obtain their value	
:	Same as = (gives value of legitimate argument)	
CTRL/TAB		Produces a tab, which on output is interpreted as 10 spaces or a tab/rubout, depending on SR option

Editor commands are given when in command mode. There are three basic types of commands: input, editing, and output. Table 5-4 contains a summary of Editor commands and their function.

Table 5-4
Summary of Commands

Type	Command	Function
Input	A	Append incoming text from keyboard into text buffer
	R	Append incoming text from tape reader into text buffer
Editing	L	List entire text buffer
	nL	List line n
	m,nL	List lines m through n inclusively
	nC	Change line n
	m,nC	Change lines m through n inclusively
	I	Insert before first line
	nI	Insert before line n
	K	Delete entire text buffer
	nD	Delete line n
	m,nD	Delete lines m through n inclusively
	m,n$jM	Move lines m through n to before line j
	G	Print next tagged line (if none, Editor types ?)
	nG	Print next tagged line after line n (if none, ?)
	S	Search buffer for character specified after RETURN key and allow modification (search character is not echoed on printer)
	nS	Search line n, as above
	m,nS	Search lines m through n inclusively, as above
Output	P	Punch entire text buffer
	nP	Punch line n
	m,nP	Punch lines m through n inclusively

Table 5-4 (Cont)
Summary of Commands

Type	Command	Function
Output (Cont)	T	Punch about 6 inches of leader/trailer tape
	F	Punch a FORM FEED onto tape
	N	Do P, F, K, and R commands

Notes: 1. m and n are decimal numbers, and m is smaller than n; j is a decimal number.

2. The P and N commands halt Editor to allow the programmer to select I/O control; press CONT to execute these commands.

3. Commands are executed when the RETURN key is depressed, excluding the P and N commands.

5.3 PAL16 ASSEMBLER

The PAL16 Symbolic Assembler (PAL stands for Program Assembly Language) is a system program used to translate symbolic programs written in the PAL16 language into binary-coded (machine code) programs. PAL16 is a two-pass assembler that is run on the PDP-8 family of computers. In a two-pass assembler, the symbolic source program tape must be processed by the assembler two times to produce the binary object tape. PAL16 accepts symbolic program tapes from either the low-speed reader or the high-speed reader and produces the binary tapes on either the low-speed or high-speed punch. A brief description of the two passes is given below:

5.3.1 Pass 1

The assembler reads the symbolic program tape; stores labels with an assigned program location; checks for defined and proper operation codes, valid label fields, duplicate labels, and valid labels; and generates the label table. No error messages are typed during pass 1.

5.3.2 Pass 2

The assembler rereads the symbolic program tape, takes the indicated labels found in pass 1 and creates the object machine code. The assembler also checks for undefined and duplicate labels. During this pass the object code is punched and/or a listing of the object and source program is produced.

During assembly, the programmer communicates with PAL16 via the switch register on the computer console. Switches are set and reset to specify the high or low-speed reader for reading the source tapes, the high or low-speed punch for punching the object tape, and the TTY or line printer for listing the program. A summary of available switch register options is given in Table 5-5.

If the assembler finds errors in the source program, error messages are printed directly after the program listing and before the symbol (label) table.

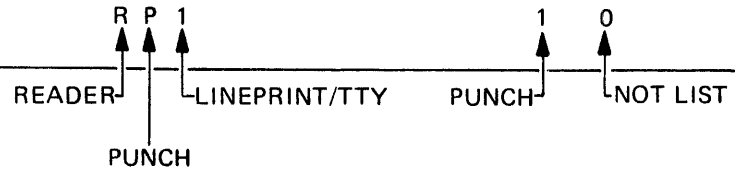
5.3.3 Assembling a Program

Paragraph 5.2 described how to prepare a PAL16 symbolic program and produce the paper tape using the Symbolic Editor. After the paper tape is punched the program can be assembled using PAL16. A listing of a sample symbolic program follows:

```
/          TEXT PRINT ROUTINE
/
/             ASCII CODES ARE STORED IN 256X8 PROM.
/             END OF MESSAGE IS CODE 377
/
/          B REGISTER HAS START ADDRESS OF TEXT
/
TEXT       RMAR=B              /INITIALIZE PROM ADDRESS
           A=ROM               /FETCH CODE
           A=ANOT              /COMPLEMENT
TEXTW      IF PF1,TEXTP
           GOTO TEXTW
TEXTP      SI1=A               /PRINT CHARACTER
           A=A+1               /INCREMENT
           A=A/2               /SHIFT RIGHT
           IF A<7>,TEXTC       /TEST FOR END OF MESSAGE
           A=B
           B=A+1               /UPDATE TEXT POINTER
           GOTO TEXT
TEXTC      EXIT
           $
```

Table 5-5
PAL16 Assembler Switch Register Option

Function	Switch Register Setting																
	0	1	2	.	.	.	.	.	.	.	.	.	.	10	11
Load Definition or Source Tape	R	P	0											0	0
Assemble and List on TTY	R	P	0											0	0
Assemble and List on Line Printer	R	P	1											0	0
Assemble and Punch	R	P	0											1	1
Assemble Punch and List on TTY	R	P	0											1	0
Assemble Punch and List on Line Printer	R	P	1											1	0



Notes: R = 0 for HSR and 1 for LSR

P = 0 for HSP and 1 for LSP

First, PAL16 must be loaded into core memory, and since PAL16 is on punched paper tape in binary format, it is loaded into core memory using the BIN Loader. Refer to Paragraph 5.1 for the loading procedure.

After PAL16 is loaded into core memory, it must be initialized before it can be used to assemble a symbolic program. Figure 5-7 illustrates the procedures required for initializing the assembler and for assembling a symbolic program using the low-speed reader/punch, the high-speed reader/punch, and the TTY or line printer.

An example of the assembly procedure using the low-speed reader/punch (LSR and LSP) follows (Figure 5-7).

Initializing and Starting	Load PAL16 into core memory using BIN
	Set SR=0200 and depress LOAD ADDR
	Turn TTY to LINE and place definition tape (assembler initialize code) in LSR
	Set SR=4000, set LSR to START and depress CLEAR/CONT
Entering Pass 1	Place symbolic source program tape in LSR
	Set SR=0200 and depress LOAD ADDR
	Set SR=4000 and depress CLEAR/CONT
Entering Pass 2	Place symbolic source program tape in LSR again
	Set SR=2002, set LSR to START, depress LSP to ON and depress CLEAR/CONT
	The octal/symbolic program and symbol table is listed (see below) and the object code is punched on paper tape

```
                /       TEXT PRINT ROUTINE
                /
                /          ASCII CODES ARE STORED IN 256X8 PROM.
                /          END OF MESSAGE IS CODE 377
                /
                /       B REGISTER HAS START ADDRESS OF TEXT
                /
0000 246    TEXT        RMAR=B              /INITIALIZE PROM ADDRESS
0001 135                A=ROM               /FETCH CODE
0002 011                A=ANOT              /COMPLEMENT
0003 364    TEXTW       IF PF1,TEXTP        /WAIT FOR TTY
0004 007
0005 300                GOTO TEXTW
0006 003
0007 050    TEXTP       SI1=A               /PRINT CHARACTER
0010 002                A=A+1               /INCREMENT
0011 012                A=A/2               /SHIFT RIGHT
0012 334                IF A<7>,TEXTC       /TEST FOR END OF MESSAGE CODE
```

```
0013 020
0014 001                    A=B
0015 257                    B=A+1          /UPDATE TEXT POINTER
0016 300                    GOTO TEXT
0017 000
0020 377    TEXTC           EXIT

TEXT        0000
TEXTW       0003
TEXTP       0007
TEXTC       0020
```

The paper tape contains the binary object code (in ASCII format) that is loaded into the control PROM using the load function of the utility program. Refer to Chapter 6. The octal/symbolic program listing and symbol table produced during pass 2 is used when debugging the program. If errors in the source program are encountered by the assembler, appropriate error messages are printed just before the symbol table. Each error message is preceded by a four digit line number (in octal) where the error was encountered.

5.3.4 Error Messages

The error messages that are printed are mostly self-explanatory. A brief description of each error message follows:

XXXX UNDEFINED OP-CODE

This message is printed when an operation code is used that is not defined in the definition tape. For example:

```
A=ANOT      /where A=ANOT is not defined in definition tape.
```

XXXX INVALID LABEL

This message is printed when a label greater than 10 characters in length or a label starting with a non-alphabetic character is used. For example:

```
1LOOP
```

XXXX UNDEFINED LABEL

This message is printed when a test condition for the IF statement is not defined in the definition tape or when the destination label in a GOTO or an IF statement are not defined. For example:

```
IF DN,NEG
```

where DN or NEG is not defined. DN must be defined in the definition tape and NEG must be defined in the source program and must appear on the left margin.

XXXX DUPLICATE LABEL

This message is printed when two labels with the same name are used.

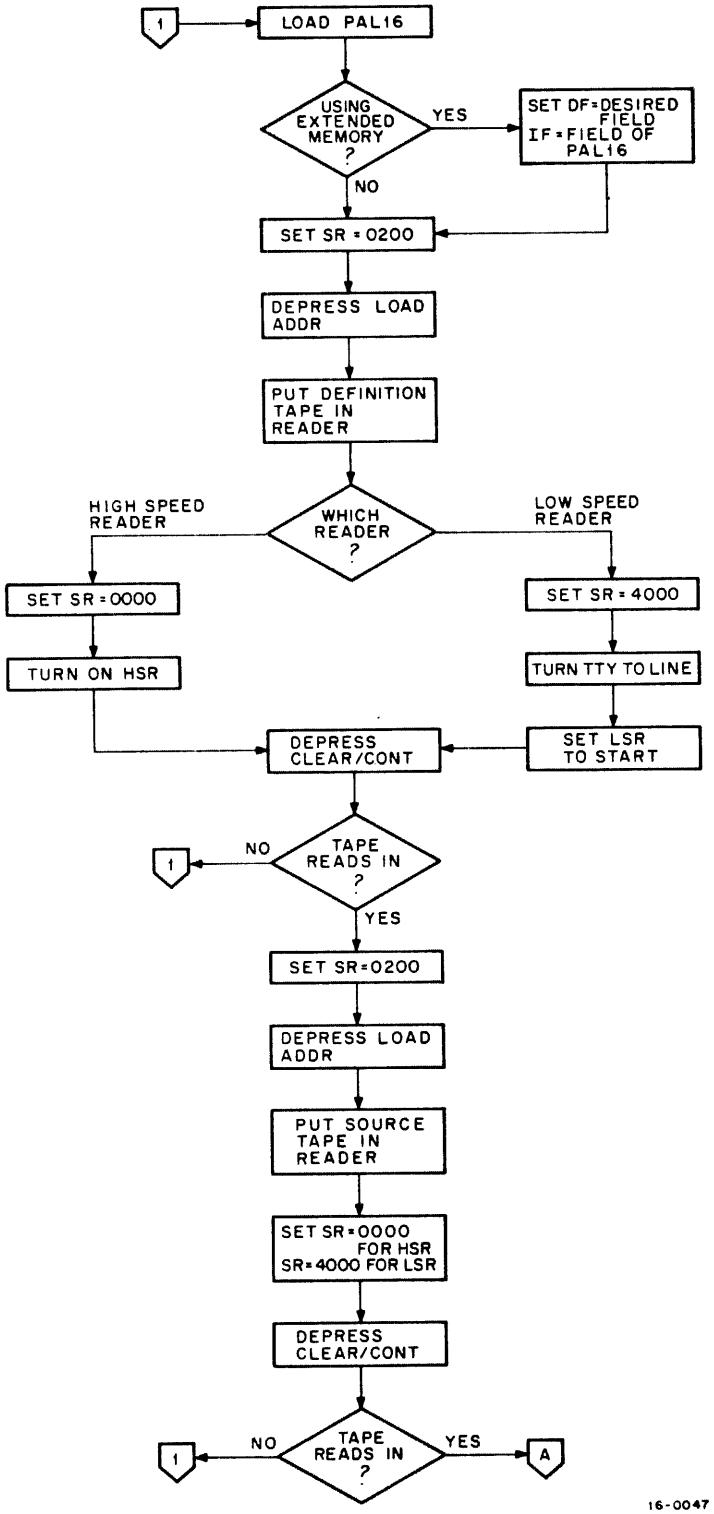


Figure 5-7 Assembling with PAL16 (Sheet 1 of 2)

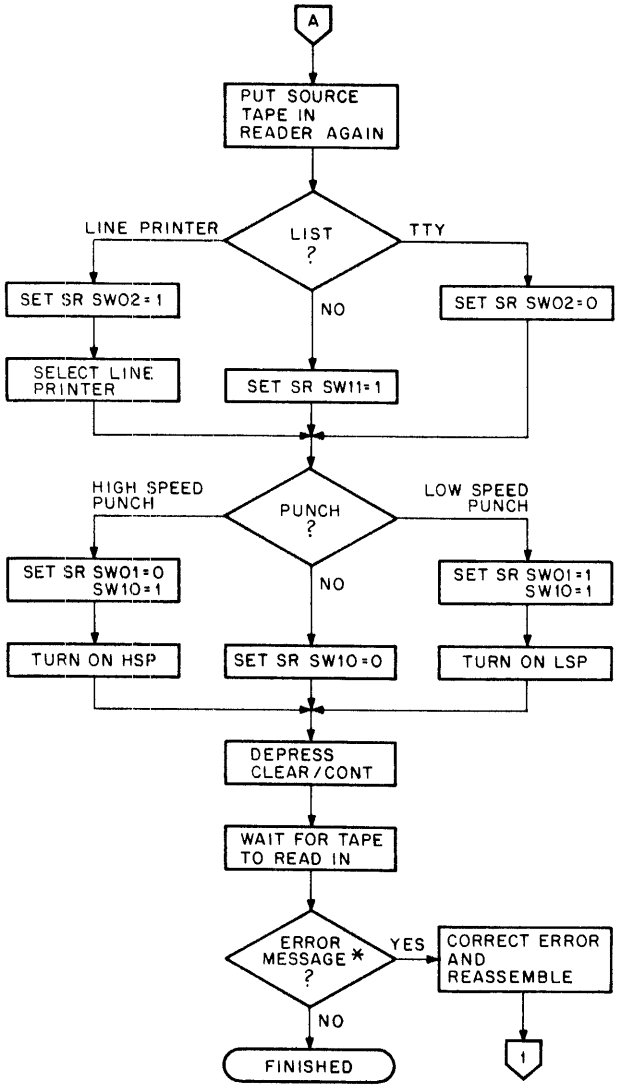


Figure 5-7 Assembling with PAL16 (Sheet 2 of 2)

XXXX MISSING OPERAND

This message is printed when no destination label is given to the GOTO or IF statements. For example:

```
IF DP, OR GOTO
```

XXXX INVALID STATEMENT FORMAT

This message is printed when the operation code starts on the left margin (first space) or if a label appears with no operation code or operand.

XXXX INVALID CHARACTER

This message is printed when other than characters represented by ASCII codes 212 through 337 are used in the source program. Therefore the ALT MODE key and all control command if used in the source program will cause this error message to be printed.

XXXX NON-OCTAL DIGIT

This message is printed if a non-octal digit is used in preparing the definition table tape. For example:

```
A=AB    DI    008
```

XXXX CONSTANT TOO LARGE

This message is printed when the evaluation number in the definition table exceeds the limit or if an ORG statement is out if range. For example:

```
B=SI1    DI    300
ORG 3777
```

XXXX NO LABEL ON EQU STATEMENT

This message is printed when the evaluation number in the definition table is missing. For example:

```
B=SI1    DI
```

XXXX ADDRESSING ERROR

This message is printed if a CALL, GOTO, or IF instruction runs off the end of the page or references a label located in the other page.

CHAPTER 6
UTILITY OPTION (TENTATIVE)

After a program is written and assembled error-free, the program and the application interface must be exercised and debugged. A new program will not always run the first time. Usually, inconspicuous bugs are characteristic of new programs and newly interfaced equipment. Only after the program and the interfacing equipment are debugged should the PROM be loaded. The utility option designated MR16-SL, offered by Digital Equipment Corporation in conjunction with a PDP-8/E computer and supporting utility software, offers the means with which the user can exercise, debug, and load his program (Appendix H). In addition to exercising and loading, this option offers useful functions such as verifying that a PROM does contain accurate object code and listing the contents of a PROM. The MR16-SL Utility Option comprises the following:

a. M8307 Utility Option Module (PROM simulator/loader control) that plugs directly into a vacant OMNIBUS slot of the PDP-8/E.

b. MR16-SL Function Panel that mounts in a standard 19-inch rack.

c. Three BC08R-XX Cables for interconnecting the M8307 Utility Option module and the function panel.

d. Utility program tape (binary) containing the following:

    a. PROGRAM Subroutine

    b. VERIFY Subroutine

    c. SIMULATE Subroutine

    d. Support Subroutines

6.1 INSTALLATION

Any standard PDP-8/E computer equipped with 4K of core memory and an ASR 33 (LT33-DC) can serve as the utility computer (Figure 6-1). Machines equipped with an ASR 35 and a high-speed reader/punch in place of the ASR 33 can also be used. To install an MR16-SL Utility Option in a PDP-8/E computer, proceed as follows:

1. Mount the MR16-SL Function Panel in close proximity to the PDP-8/E main frame.

2. Install MR16-SL Utility Option Module (M8307) in a vacant OMNIBUS slot of the PDP-8/E.

3. Connect the three supplied cables.

4. Connect 62V (0.1A) power supply (user supplied) to + and − tabs located on the function panel.

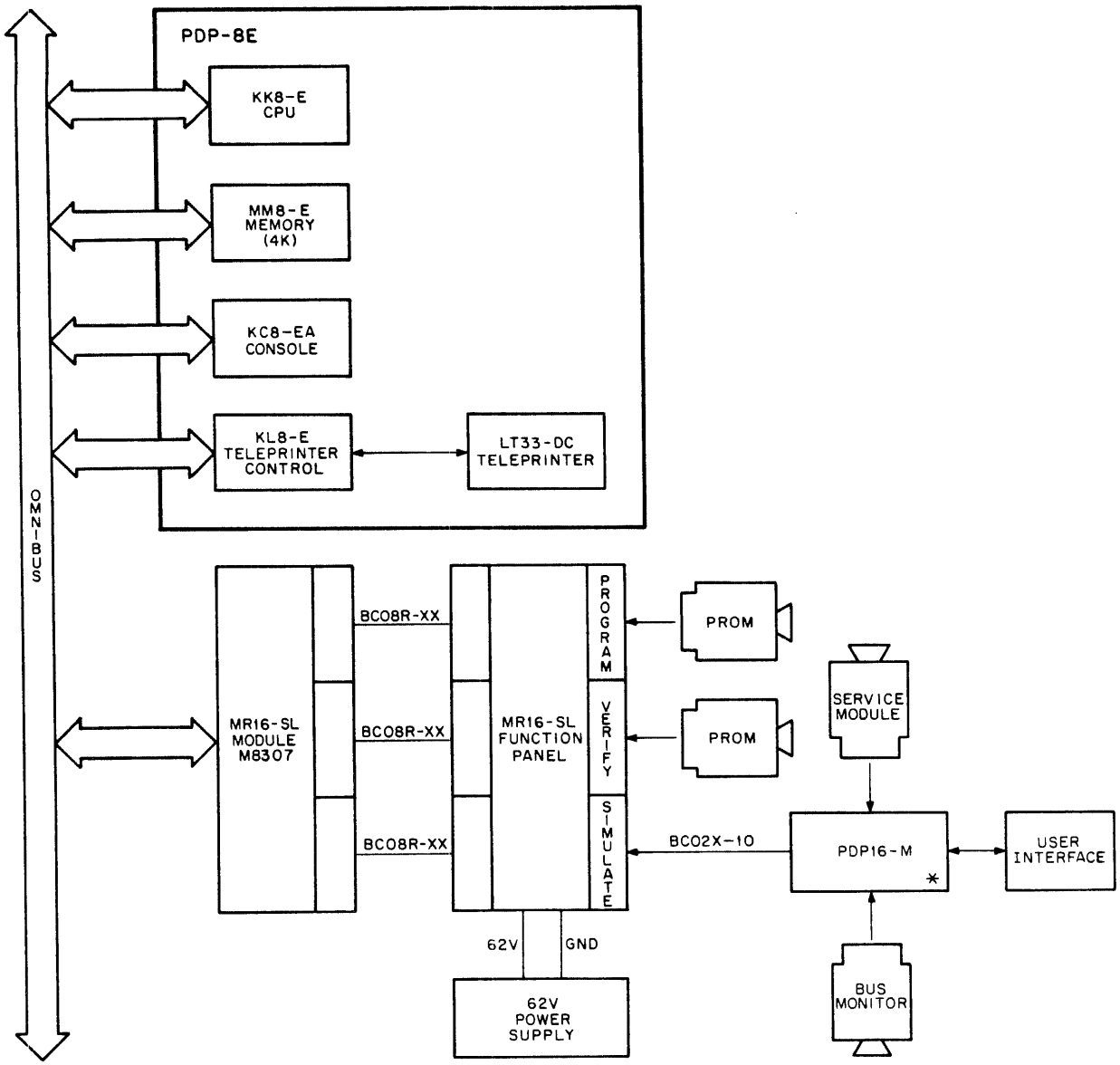


Figure 6-1 Utility Computer Configuration

Installing the utility option in a PDP-8/E does not dedicate the machine to just utility functions. The machine can still be used for program development as described in Chapter 5.

6.2 OPTION DESCRIPTION

The MR16-SL is a programmable device interface (Figure 6-2) for debugging, verifying, and loading PDP16-M application programs. A special utility program controls the operation of the device interface. The function panel houses three connectors; one for each of the specified utility functions. The connectors are placarded: PROGRAM, VERIFY, and SIMULATE. To program (load) or verify (check) a PROM, the PROM must be inserted in the

corresponding connector. To exercise a program for debugging purposes, the PDP16-M is connected to the SIMULATE connector. After the utility program is loaded into PDP-8/E core memory and started, it will query the operator as to which function is desired by typing an F on the teleprinter. The operator can then select the desired function by typing the appropriate command.

Since the MR16-SL is fully supported by the utility program, the user need not concern himself with programming the MR16-SL. Therefore, detailed description and programming information for the MR16-SL are not provided in this manual.

6.3 OPERATIONAL DETAILS

To use the utility option, the user must first load the utility program into PDP-8/E core memory and start the program at octal location 6000. The Binary Loader must be used to load the utility program tape (Paragraph 5.1). After the program is loaded and started, the user can select any one of eight utility functions by typing the appropriate command. The commands are:

Z – Zero

F – Fetch

L – Load

M – Modify

S – Simulate

B – Program

C – Check

T – Type

The zero, fetch, load, and modify functions all deal with loading a PDP16-M application program into core memory. The application program must be loaded into core memory of the PDP-8/E before a PROM can be loaded or checked and before the program can be exercised on a PDP16-M for debugging purposes. To debug a program and its application interface, the S command must be declared to select the simulate function. The B command stands for PROGRAM and must be declared to load a PROM. To check the contents of a PROM against the application program in PDP-8/E core memory, the C command must be declared. Finally, the T command, which stands for TYPE, can be declared to list the content of the PROM on the teleprinter.

The Z function is provided for zeroing the first 6000_8 locations of core memory. These locations are used as the application program storage buffer. The application program is loaded into this buffer using the F, L, or M function. The F function is used to read the program from a PROM; the L and M functions are used to read the program from a paper tape. Loading a tape using the M function will cause the contents of the tape to be echoed on the TTY. Therefore, unless a printout is desired, the L function should be used to load the tape. The buffer must be zeroed before loading the application program so that any extraneous data that may be in these locations will not be loaded into the PROM or checked. The B function is used to load a Control or Data PROM and the C function is used to check the contents of the PROM against the contents of the application program buffer.

Either the low-speed or high-speed paper-tape reader can be used for loading a paper tape.

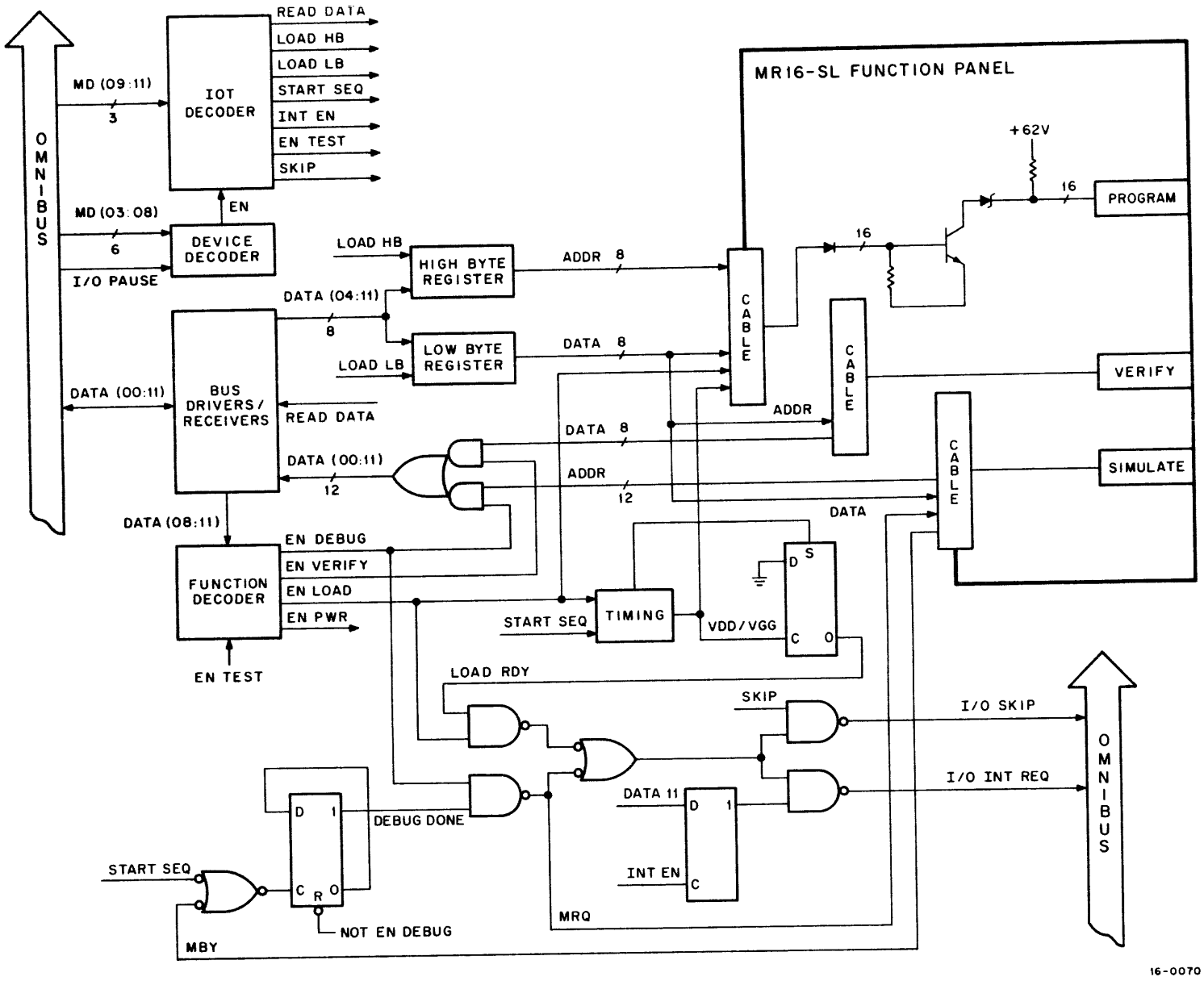


Figure 6-2 MR16-SL Utility Option

Besides loading the application program, the M function can also be used for modifying the application program in the buffer or inputting a program from the TTY keyboard. Only octal numbers, the % sign, and the $ sign have any meaning for the modify function. A % sign followed by an octal number, a RETURN and a LINE FEED sets the program counter (address) to the value of the octal number. Typing three octal digits, a RETURN, and a LINE FEED, stores the binary value of the octal number in the memory location pointed to by the program counter. After an object program resides in the buffer, it can be executed on the application PDP16-M by selecting the S function, it can be loaded into a PROM by selecting the B function, or a PROM loaded with the same program can be checked by selecting the C function. Finally, the T function, which stands for TYPE, can be selected to list the contents of a PROM.

6.3.1 Zero, Fetch, Load and Modify

These functions may be used to load the application program object code into PDP-8/E core memory for subsequent simulation, PROM loading, or PROM checking. After the utility program is loaded and started (SA=6000_8) by depressing the PDP-8/E START switch, the program types:

```
F
```

The user may then select the zero, fetch, load, or modify function by typing the corresponding command character. When first starting, it is recommended to zero the buffer by typing Z in response to the F query. For example:

```
FZ
```

6.3.1.1 Load – To load an object tape (punched by the PAL16 Assembler), simply place the tape in the reader (high or low speed) and type L for load. If the tape is in the high-speed reader it will read in. If the low-speed reader is used, push LSR switch to START; after the tape is read, turn the LSR to OFF. Whether the LSR or the HSR is used to load the tape, the CONT switch on the utility computer console must be depressed to reactivate the utility program. The program will then type F, requesting another function. The user can now modify the program, simulate the program, load a PROM, or check a PROM by selecting the appropriate function and performing the necessary setup procedure. Refer to respective function descriptions.

6.3.1.2 Modify – The modify function allows the user to change the program stored in core memory or to manually input a program via the teletype keyboard. To select this function, the user types M in response to F. The utility computer is now ready to accept the input string. Only octal numbers, the % sign, and the $ sign have any meaning for the modify function. A % sign followed by an octal number, a RETURN, and a LINE FEED sets the PC (address) to the value of the octal number. Typing three octal digits, a RETURN, and a LINE FEED stores the binary value of the octal number in the memory location pointed to by the PC.

NOTE

When starting, the PC is always set to 0000; therefore, the % sign must be used to select memory locations at random.

A $ sign followed by a carriage RETURN and a LINE FEED must terminate each input string. After the $ sign and the control keys are typed, the program responds by typing F to request the next function. An example of an input string acceptable to the modify routine follows:

Address (not typed)	Input String	
	FM	
0	12	CR LF
1	14	CR LF
2	300	CR LF
3	1	CR LF
4	123	CR LF
5	%100	CR LF
100	2	CR LF
101	5	CR LF
102	332	CR LF
103	$	CR LF
	F	

The modify subroutine looks at only the last three data digits. Therefore, if a typing error is made, the correct three digits can be typed on the same line without having to reset the PC.

6.3.1.3 Fetch – The fetch function allows the user to transcribe a program from a PROM to PDP-8/E core memory. To use this function, the user must first insert the PROM to be transcribed into the VERIFY connector of the function panel (Figure 6-1). The user can then type F in response to the F query to select the fetch function. The user can now modify the program, simulate the program, load a PROM, or check a PROM by selecting the appropriate function and performing the necessary setup procedure. Refer to respective function description.

6.3.2 Simulate

This function is used to exercise the program for debugging purposes. To use this function, the user must first connect the PDP16-M on which the program is to be run to the SIMULATE connector on the function panel (Figure 6-1). Optionally, Service Module M7335 and Bus Monitor Module M7322 can be installed in the PDP16-M to implement the data and address readout, single step, and break-point debugging features. (Refer to Chapter 5 of the *PDP16-M Maintenance Manual*.) The PDP16-M must also be equipped with all the options that are exercised by the application program.

After the application program is loaded into core memory using the F, L, or M function, it can be executed on the application PDP16-M by typing the S command in response to the next F query. When the L function is used in loading the application program, the PDP-8/E will halt after the $ sign, CR, and LF at the end of the paper tape are read. The utility program will return by typing F on the teleprinter to request the next function only when CONT on the PDP-8/E console is depressed. This feature gives the operator time to turn off the low-speed reader. After turning off the reader and depressing CONT, the user simply types S in response to the F query and depresses the START switch on the PDP16-M to start the program. The debugging features offered by the maintenance modules are very useful in debugging the program and the application interface. Therefore, these modules should be used when debugging a program.

6.3.3 Program

This function is used to load PROMs. To use this function the user must first install a refreshed PROM in the PROGRAM connector on the function panel (Figure 6-1).

NOTE

A PROM is erased by exposir ' to ultraviolet light.

After the application program is loaded into core memory using the F, L, or M function, the PROM can be loaded by typing the B command in response to the next F query. The utility program then responds with ROM?. For example:

```
FB
ROM?
```

The user they types 0, 1, 2, or 3, depending on which PROM is to be loaded. Normally, the PROMs are loaded sequentially from 0 to 3. After the PROM is loaded, the utility program will type F to request another function. For example:

```
FB
ROM?0
F
```

The above listing indicates that the PROM was loaded successfully and the utility program is requesting another function. The same procedure is repeated to load other PROMs.

6.3.4 Check

This function is used to verify that a PROM is programmed correctly. To use this function the user must first install the PROM to be checked in the VERIFY connector on the function panel. The user must then load the master application program into core. After the application program is loaded using the F or M function, the PROM can be checked by typing the C command in response to the next F query. The utility program then responds with ROM?. For example:

```
FC
ROM?
```

The user they types 0, 1, 2, or 3, depending on which PROM is to be checked. After the contents of the PROM are checked against the contents of core memory, the utility program will type OK or X followed by F. For example:

```
FC
ROM? 0
OK
F
```

The above listing indicates that the PROM does contain an accurate application program. The same procedure is used to check other PROMs. If an X is typed instead of OK, the PROM does not contain a good application program and must be reloaded.

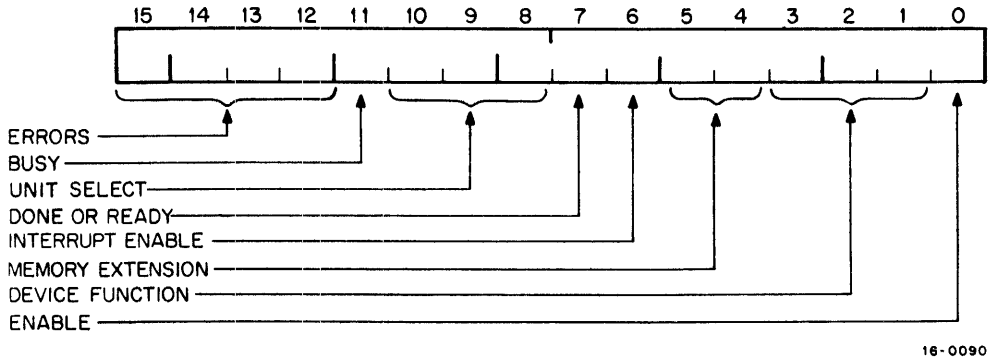
6.3.5 Type

This function is used to list the contents of a PROM. The instruction codes and addresses stored in the PROM are typed out as three octal digits. To use this function the user must first install the PROM whose contents are to be listed in the VERIFY connector on the function panel. The user then simply types T in response to the F query. After the contents of the PROM are listed, the utility program again types F to request the next function.

APPENDIX A
PDP-11 PERIPHERAL SUMMARY

A.1 CONTROL AND STATUS REGISTERS

Each peripheral has one or more control and status registers that contain all the information necessary to communicate with that device. The general form, shown below, does not necessarily apply to every device, but is presented as a guide.



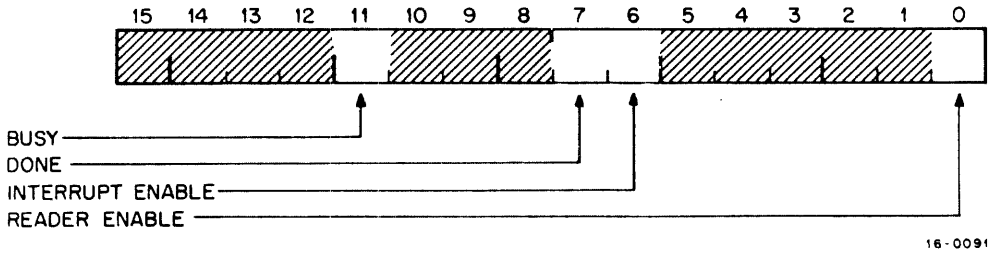
A.2 DATA REGISTERS

The data registers may contain up to 16 bits of data depending on the type of device. TTY compatible devices transfer only eight bits of data at a time.

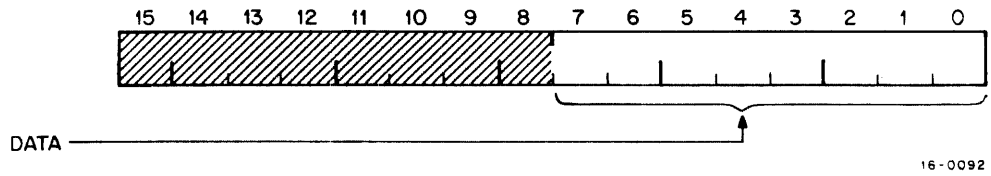
A.3 ASR 33 TELETYPE

Register	Address
Reader Status Register (TKS)	777560
Reader Buffer Register (TKB)	777562
Punch Status Register (TPS)	777564
Punch Buffer Register (TPB)	777566

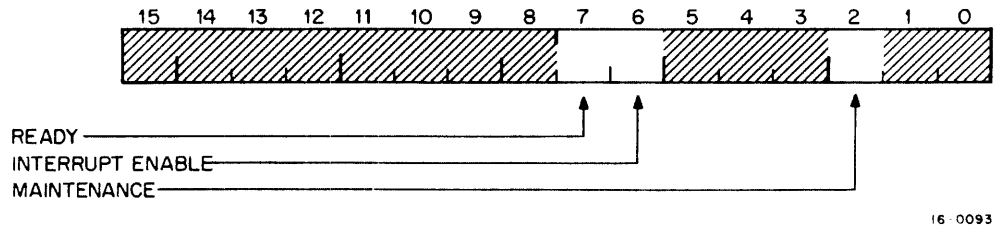
Reader Status Register (TKS)



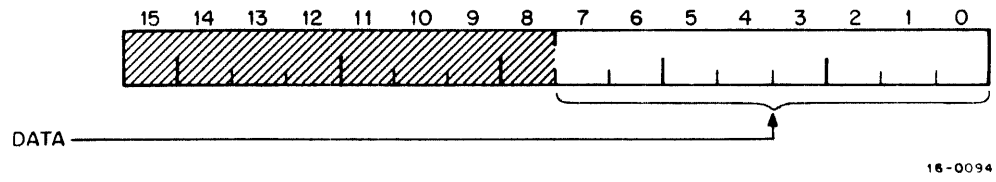
Reader Buffer (TKB)



Punch Status Register (TPS)



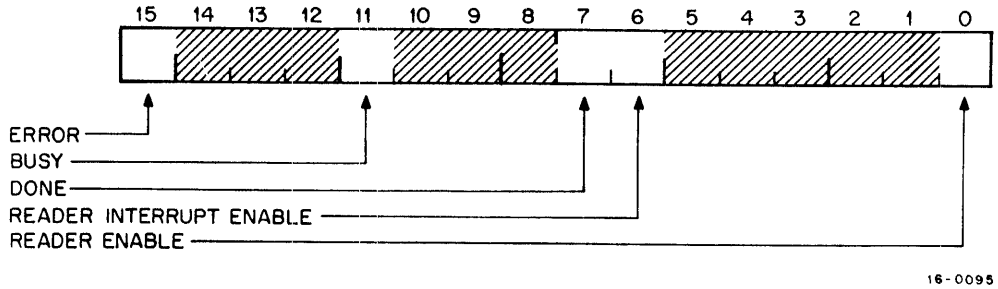
Punch Buffer Register (TPB)



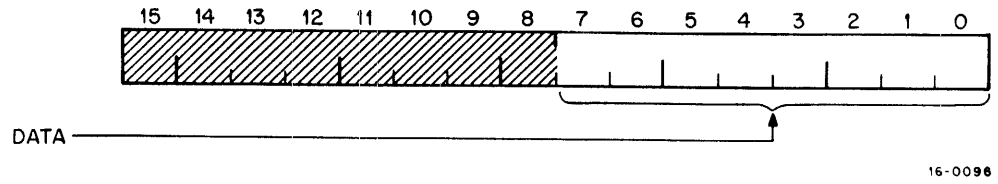
A.4 PC11 HIGH SPEED READER PUNCH

Register	Address
Papertape Reader Status Register (PRS)	777550
Papertape Reader Buffer (PRB)	777552
Papertape Punch Status (PPS)	777554
Papertape Punch Buffer (PPB)	777556

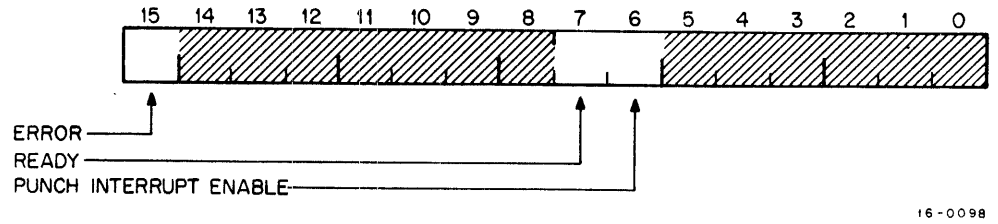
Papertape Reader Status Register (PRS)



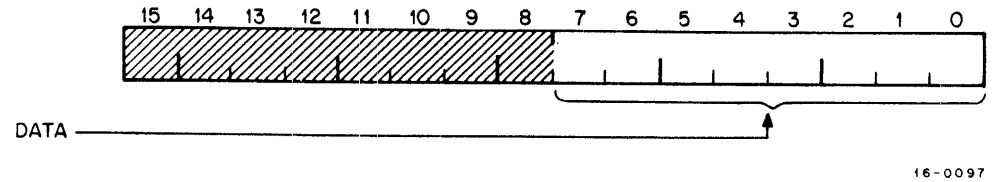
Papertape Reader Buffer (PRB)



Papertape Punch Status (PPS)



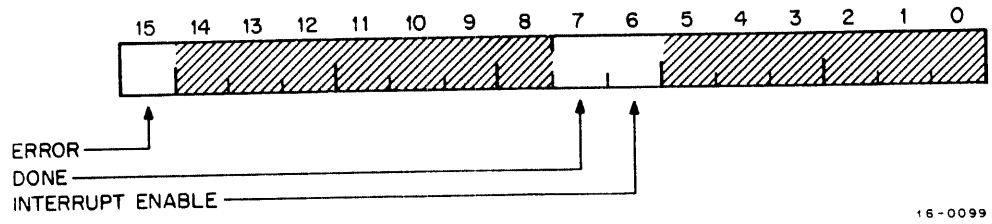
Papertape Punch Buffer Register (PPB)



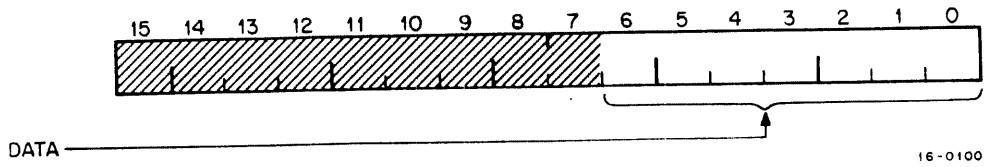
A.5 LP11 LINE PRINTER

Register	Address
Line Printer Status (LPS)	777514
Line Printer Data Buffer (LPB)	777516

Line Printer Status Register (LPS)



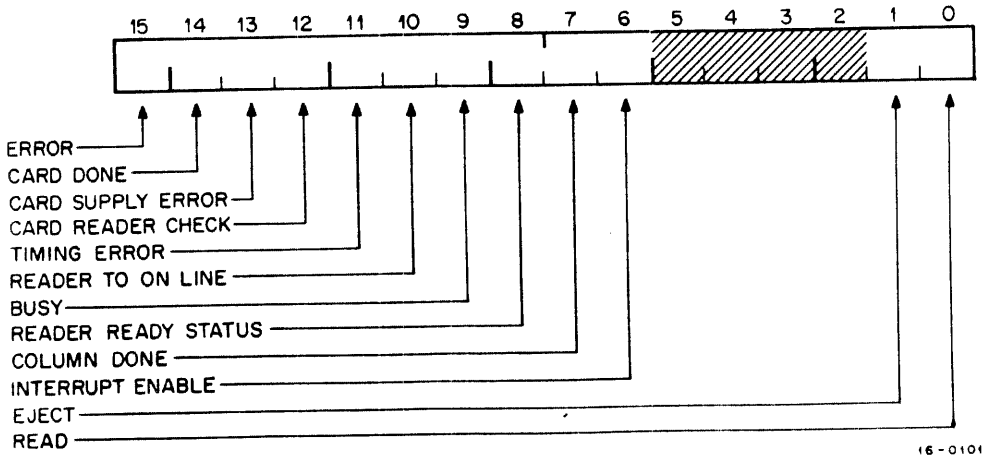
Line Printer Data Buffer Register (LPB)



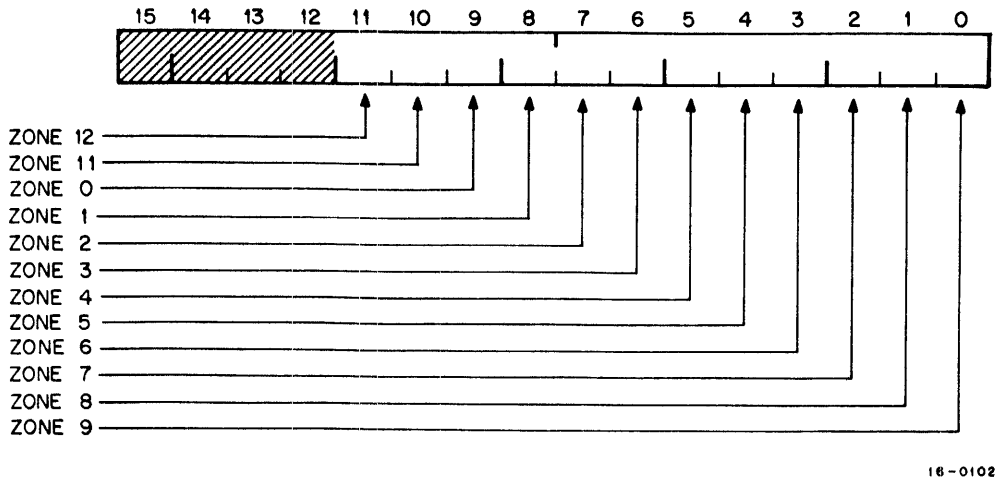
A.6 CR11 AND CM11 CARD READER

Register	Address
Card Reader Status (CRS)	777160
Card Reader Data Buffer (CRB1)	777162
Card Reader Data Buffer (CRB2)	777164

Card Reader Status Register (CRS)



Card Reader Data Buffer Register (CRB1, CRB2)

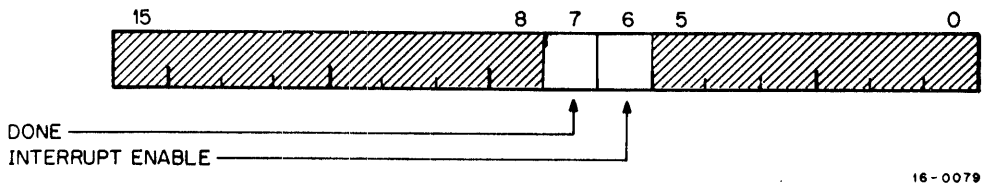


No information can be loaded into the Card Reader Data Buffer (CRB1) by any program; the content of this register can only be read.

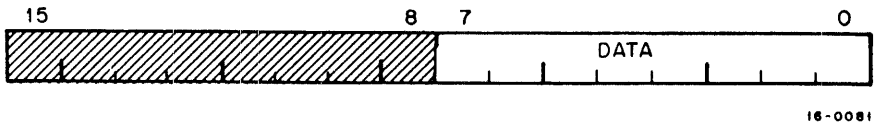
A.7 LA30 DECWRITER

Register	Address
Keyboard Status Register (KBS)	777560
Keybuffer Register (KBB)	777562
Printer Status Register (PRS)	777564
Printer Buffer Register (PRB)	777566

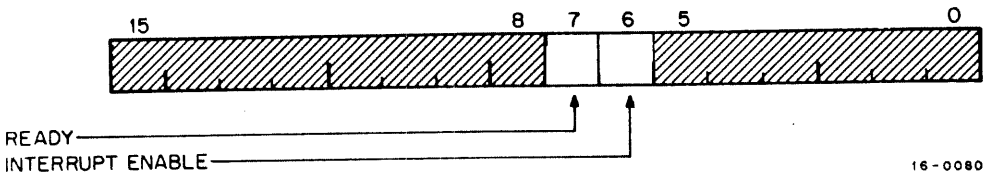
Keyboard Status Register (KBS)



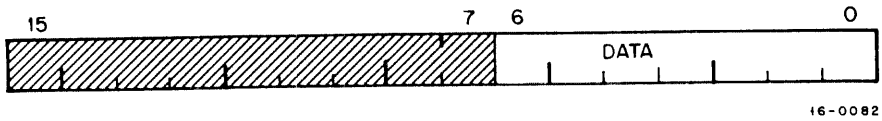
Keyboard Buffer Register (KBB)



Printer Status Register (PRS)



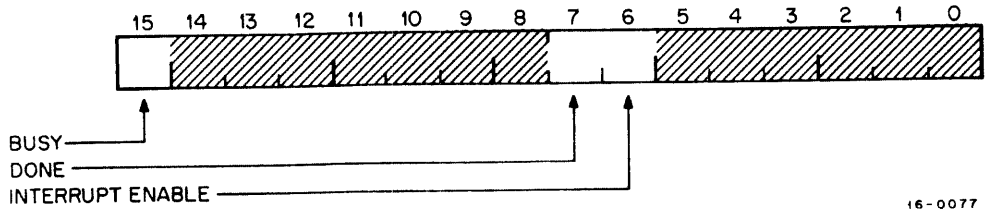
Printer Buffer (PRB)



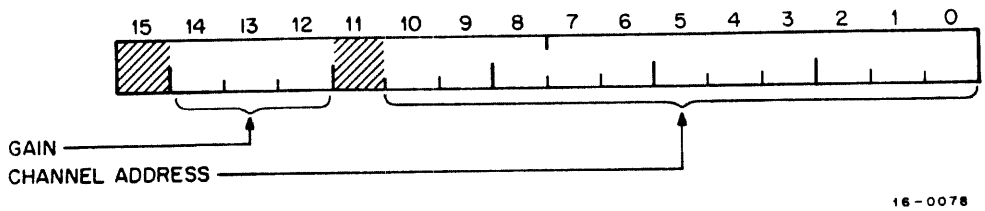
A.8 AFC11 FLYING CAPACITOR

Register	Address
Control and Status Register (AFCS)	772570
Data Buffer Register (AFBR)	772572
Multiplexer Channel/Gain Register (AFCG)	772574
Maintenance Register (AFMR)	772576

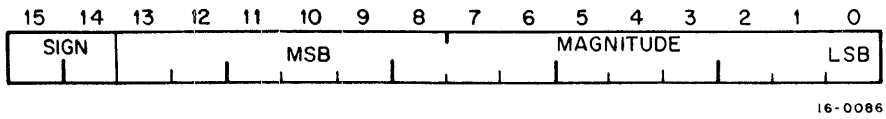
Control and Status Register (AFCS)



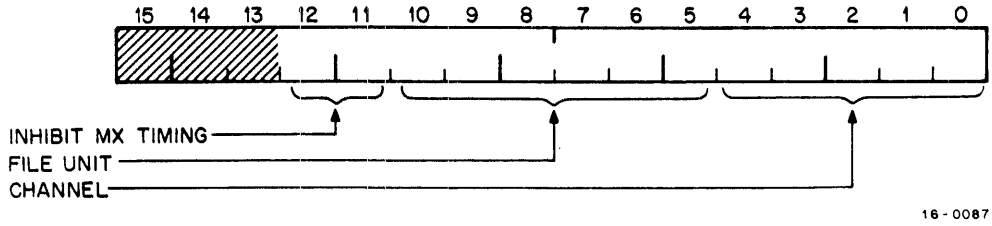
Multiplexer Channel/Gain Register (AFCG)



Data Buffer Register (AFBR)



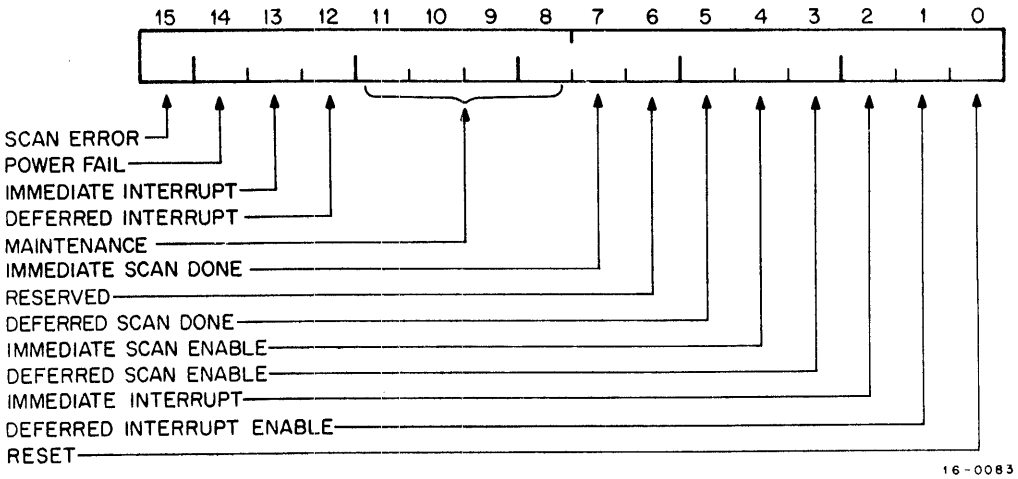
Maintenance Register (AFMR)



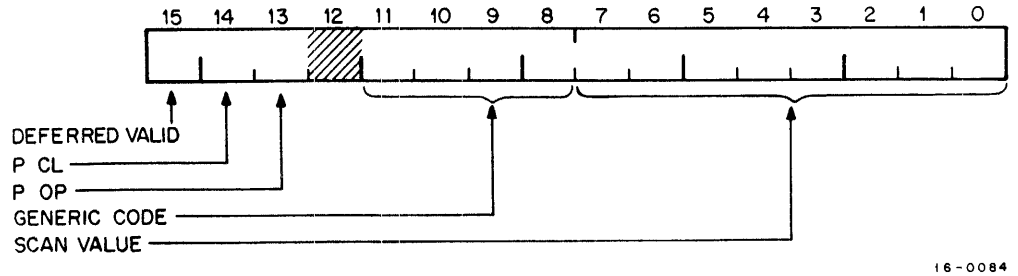
A.9 UDC11 DIGITAL I/O

Register	Address
Control Status Register (UDCS)	771776
Scan Register (UDSR)	771774
Data Registers	771XXX

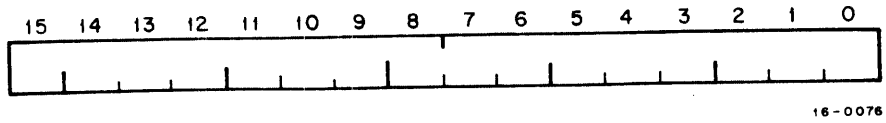
Control and Status Register (UDCR)



Scan Register (UDSR)



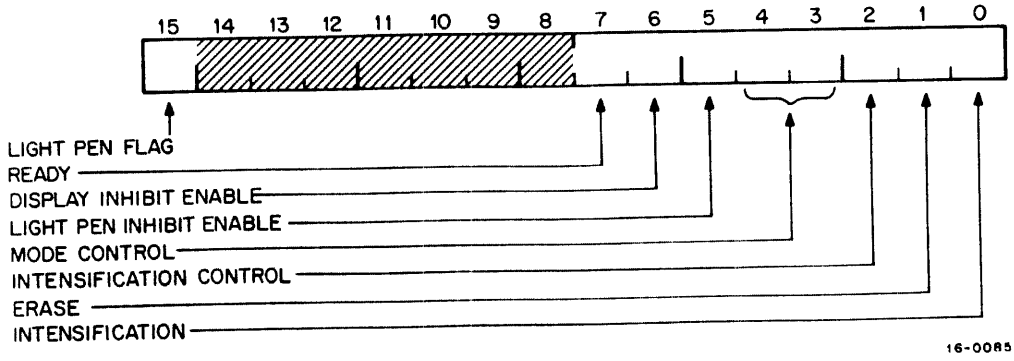
Data Registers



A.10 AA11-D D/A CONVERTER

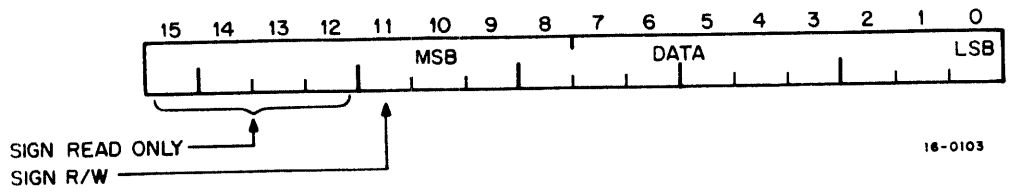
Register	Address
Command and Status Register (CSR)	776756
Data Register DAC1	776760
Data Register DAC2	775762
Data Register DAC3	776704
Data Register DAC4	776766

Command and Status Register (CSR)



Data Registers (DAC)

DAC1 and 2 may be used either in conjunction with the scope or for D/A channels. DAC3 and 4 may be used for additional D/A channels.

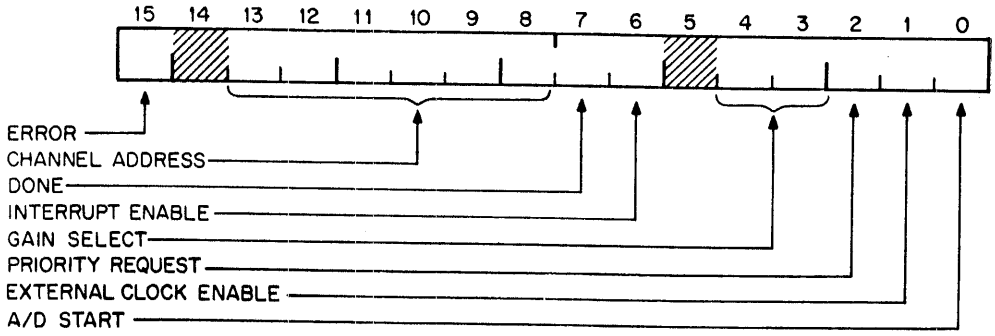


A.11 AD01-D A/D CONVERTER

Register	Address
Control and Status Register	776770
Data Register	776772

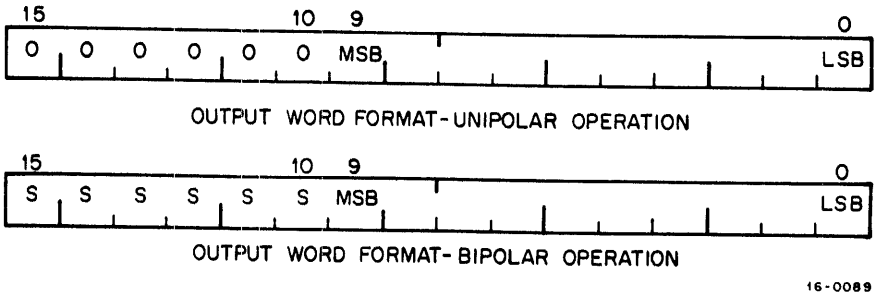
Control and Status Register

Transfer of a 16-bit control word from the PDP-11 to the control and status register (ADCS 776770) establishes the operating conditions of the AD01-D.



Data Register

The A/D Converter Data Register (ADDB-776772) transfers data to the PDP-11 in the following format.



Bits 15 to 10 are tied together, and are 0 in the standard unipolar configuration. With the sign bit option, bits 15 to 10 indicate the sign of the input voltage.

Output Notation Table

Analog Input Voltage	Unipolar	Bipolar
-10.0		176000
5.0		177000
0.0	000000	000000
+ 5.0	001000	001000
+ 9.9902	001777	001777

*For 10V full scale input range; divide by appropriate gain factor for other input ranges.

APPENDIX B
ASCII AND EBCDIC CHARACTER SET ENCODINGS

USASCII Data Transmission Code

Bit Positions 4	3	2	1	7 6 5 = 0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
0	0	0	0	NUL	DLE	SP	0	@	P	'	p
0	0	0	1	SOH	DCL	!	1	A	Q	a	q
0	0	1	0	STX	DC1	"	2	B	R	b	r
0	0	1	1	ETX	DC3	#	3	C	S	c	s
0	1	0	0	EOT	DC4	$	4	D	T	d	t
0	1	0	1	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	ACK	SYN	&	6	F	V	f	v
0	1	1	1	BEL	ETB	'	7	G	W	g	w
1	0	0	0	BS	CAN	(	8	H	X	h	x
1	0	0	1	HT	EM	)	9	I	Y	i	y
1	0	1	0	LF	SUB	*	:	J	Z	j	z
1	0	1	1	VT	ESC	+	;	K	[	k	{
1	1	0	0	FF	FS	,	<	L	\	l	|
1	1	0	1	CR	GS	-	+	M	]	m	}
1	1	1	0	SO	RS	.	>	N	^	n	~
1	1	1	1	SI	US	/	?	O	_	o	DEL

USASCII Data Transmission Code
(Key)

NUL = All zeros
SOH = Start of heading
STX = Start of text
ETX = End of text
EOT = End of transmission
ENQ = Enquiry
ACK = Acknowledgement
BEL = Bell or attention signal
BS = Back space
HT = Horizontal tabulation
LF = Line feed
VT = Vertical tabulation
FF = Form feed
CR = Carriage return
SO = Shift out
SI = Shift in
DLE = Data link escape
DC1 = Device control 1
DC2 = Device control 2
DC3 = Device control 3
DC4 = Device control 4
NAK = Negative acknowledgement
SYN = Synchronous/idle
ETB = End of transmitted block
CAN = Cancel (error in data)
EM = End of medium
SUB = Start of special sequence
ESC = Escape
FS = Information file separator
GS = Information group separator
RS = Information record separator
US = Information unit separator
DEL = Delete

EBCDIC Code

Bit Positions				0,1	00				01				10				11			
4	5	6	7	2,3	00	01	10	11	00	01	10	11	00	01	10	11	00	01	10	11
0	0	0	0		NUL	DLE	DS		SP	&	–									0
0	0	0	1		SOH	DC1	SOS				/		a	j			A	J		1
0	0	1	0		STX	DC2	FS	SYN					b	k	s		B	K	S	2
0	0	1	1		ETX	DC3							c	l	t		C	L	T	3
0	1	0	0		PF	RES	BYP	PN					d	m	u		D	M	U	4
0	1	0	1		HT	NL	LF	RS					e	n	v		E	N	V	5
0	1	1	0		LC	BS	ETB	UC					f	o	w		F	O	W	6
0	1	1	1		DEL	IL	ESC	EOT					g	p	x		G	P	X	7
1	0	0	0			CAN							h	q	y		H	Q	Y	8
1	0	0	1			EM							i	r	z		I	R	Z	9
1	0	1	0		SMM	CC	SM		¢	!		:								
1	0	1	1		VT					$	,	#								
1	1	0	0		FF	IFS		DC4	<	*	%	@								
1	1	0	1		CR	IGS	ENQ	NAK	(	)	_	'								
1	1	1	0		SO	IRS	ACK		+	;	>	=								
1	1	1	1		SI	IUS	BEL	SUB	|	¬	?	"								

EBCDIC Code (Key)

NUL	Null
PF	Punch Off
HT	Horizontal Tab
LC	Lower Case
DEL	Delete
RES	Restore
NL	New Line
BS	Backspace
IL	Idle
CC	Cursor Control
DS	Digit Select
SOS	Start of Significance
FS	Field Separator
BYP	Bypass
LF	Line Feed
EOB	End of Block (or ETB, End of Transmission Block)
PRE	Prefix (or ESC, Escape)
SM	Set Mode
PN	Punch On
RS	Reader Step
UC	Upper Case
EOT	End of Transmission
SP	Space
SOH	Start of Heading
STX	Start of Text
ETX	End of Text
ENQ	Enquire
ACK	Acknowledge
BEL	Bell
VT	Vertical Tabulation
FF	Form Feed
SO	Shift Out
SI	Shift In
SMM	Start of Manual Message
DLE	Data Link Escape
DC1	Device Control 1
DC2	Device Control 2
DC3	Device Control 3
DC4	Device Control 4
NAK	Negative Acknowledge
SYN	Synchronous Idle
CAN	Cancel
EM	End of Medium
SUB	Substitute
IGS	Information Group Separator
IRS	Information Record Separator
IUS	Information Unit Separator
IFS	Information Field Separator

APPENDIX C
CONVERSION TABLES

Scales of Notation

2^x IN DECIMAL

x	2^x	x	2^x	x	2^x
0.001	1.00069 33874 62581	0.01	1.00695 55500 56719	0.1	1.07177 34625 36293
0.002	1.00138 72557 11335	0.02	1.01395 94797 90029	0.2	1.14869 83549 97035
0.003	1.00208 16050 79633	0.03	1.02101 21257 07193	0.3	1.23114 44133 44916
0.004	1.00277 64359 01078	0.04	1.02811 38266 56067	0.4	1.31950 79107 72894
0.005	1.00347 17485 09503	0.05	1.03526 49238 41377	0.5	1.41421 35623 73095
0.006	1.00416 75432 38973	0.06	1.04246 57608 41121	0.6	1.51571 65665 10398
0.007	1.00486 38204 23785	0.07	1.04971 66836 23067	0.7	1.62450 47927 12471
0.008	1.00556 05803 98468	0.08	1.05701 80405 61380	0.8	1.74110 11265 92248
0.009	1.00625 78234 97782	0.09	1.06437 01824 53360	0.9	1.86606 59830 73615

$10^{\pm n}$ IN OCTAL

10^n	n	10^{-n}	10^n	n	10^{-n}
1	0	1.000 000 000 000 000 000 000 00	112 402 762 000	10	0.000 000 000 006 676 337 66
12	1	0.063 146 314 631 463 146 31	1 351 035 564 000	11	0.000 000 000 000 537 657 77
144	2	0.005 075 341 217 270 243 66	16 432 451 210 000	12	0.000 000 000 000 043 136 32
1 750	3	0.000 406 111 564 570 651 77	221 411 634 520 000	13	0.000 000 000 000 003 411 35
23 420	4	0.000 032 155 613 530 704 15	2 657 142 036 440 000	14	0.000 000 000 000 000 264 11
303 240	5	0.000 002 476 132 610 706 64	34 327 724 461 500 000	15	0.000 000 000 000 000 022 01
3 641 100	6	0.000 000 206 157 364 055 37	434 157 115 760 200 000	16	0.000 000 000 000 000 001 63
46 113 200	7	0.000 000 015 327 745 152 75	5 432 127 413 542 400 000	17	0.000 000 000 000 000 000 14
575 360 400	8	0.000 000 001 257 143 561 06	67 405 553 164 731 000 000	18	0.000 000 000 000 000 000 01
7 346 545 000	9	0.000 000 000 104 560 276 41			

$n \log_{10} 2$, $n \log_2 10$ IN DECIMAL

n	$n \log_{10} 2$	$n \log_2 10$	n	$n \log_{10} 2$	$n \log_2 10$
1	0.30102 99957	3.32192 80949	6	1.80617 99740	19.93156 85693
2	0.60205 99913	6.64385 61898	7	2.10720 99696	23.25349 66642
3	0.90308 99870	9.96578 42847	8	2.40823 99653	26.57542 47591
4	1.20411 99827	13.28771 23795	9	2.70926 99610	29.89735 28540
5	1.50514 99783	16.60964 04744	10	3.01029 99566	33.21928 09489

ADDITION AND MULTIPLICATION TABLES

Addition

Multiplication

Binary Scale

$$0 + 0 = 0 \qquad 0 + 1 = 1 \qquad 1 + 0 = 1 \qquad 1 + 1 = 10$$

$$0 \cdot 0 = 0 \qquad 0 \cdot 1 = 0 \qquad 1 \cdot 0 = 0 \qquad 1 \cdot 1 = 1$$

Octal Scale

+	01	02	03	04	05	06	07
1	02	03	04	05	06	07	10
2	03	04	05	06	07	10	11
3	04	05	06	07	10	11	12
4	05	06	07	10	11	12	13
5	06	07	10	11	12	13	14
6	07	10	11	12	13	14	15
7	10	11	12	13	14	15	16

×	02	03	04	05	06	07
2	04	06	10	12	14	16
3	06	11	14	17	22	25
4	10	14	20	24	30	34
5	12	17	24	31	36	43
6	14	22	30	36	44	52
7	16	25	34	43	52	61

MATHEMATICAL CONSTANTS IN OCTAL SCALE

π = 3.11037 552421$_8$	e = 2.55760 521305$_8$	γ = 0.44742 147707$_8$
π^{-1} = 0.24276 301556$_8$	e^{-1} = 0.27426 530661$_8$	$\ln \gamma$ = −0.43127 233602$_8$
$\sqrt{\pi}$ = 1.61337 611067$_8$	$\sqrt{e}$ = 1.51411 230704$_8$	$\log_2 \gamma$ = −0.62573 030645$_8$
$\ln \pi$ = 1.11206 404435$_8$	$\log_{10} e$ = 0.33626 754251$_8$	$\sqrt{2}$ = 1.32404 746320$_8$
$\log_2 \pi$ = 1.51544 163223$_8$	$\log_2 e$ = 1.34252 166245$_8$	$\ln 2$ = 0.54271 027760$_8$
$\sqrt{10}$ = 3.12305 407267$_8$	$\log_2 10$ = 3.24464 741136$_8$	$\ln 10$ = 2.23273 067355$_8$

Powers of Two

2^n	n	2^{-n}
1	0	1.0
2	1	0.5
4	2	0.25
8	3	0.125
16	4	0.062 5
32	5	0.031 25
64	6	0.015 625
128	7	0.007 812 5
256	8	0.003 906 25
512	9	0.001 953 125
1 024	10	0.000 976 562 5
2 048	11	0.000 488 281 25
4 096	12	0.000 244 140 625
8 192	13	0.000 122 070 312 5
16 384	14	0.000 061 035 156 25
32 768	15	0.000 030 517 578 125
65 536	16	0.000 015 258 789 062 5
131 072	17	0.000 007 629 394 531 25
262 144	18	0.000 003 814 697 265 625
524 288	19	0.000 001 907 348 632 812 5
1 048 576	20	0.000 000 953 674 316 406 25
2 097 152	21	0.000 000 476 837 158 203 125
4 194 304	22	0.000 000 238 418 579 101 562 5
8 388 608	23	0.000 000 119 209 289 550 781 25
16 777 216	24	0.000 000 059 604 644 775 390 625
33 554 432	25	0.000 000 029 802 322 387 695 312 5
67 108 864	26	0.000 000 014 901 161 193 847 656 25
134 217 728	27	0.000 000 007 450 580 596 923 828 125
268 435 456	28	0.000 000 003 725 290 298 461 914 062 5
536 870 912	29	0.000 000 001 862 645 149 230 957 031 25
1 073 741 824	30	0.000 000 000 931 322 574 615 478 515 625
2 147 483 648	31	0.000 000 000 465 661 287 307 739 257 812 5
4 294 967 296	32	0.000 000 000 232 830 643 653 869 628 906 25
8 589 934 592	33	0.000 000 000 116 415 321 826 934 814 453 125
17 179 869 184	34	0.000 000 000 058 207 660 913 467 407 226 562 5
34 359 738 368	35	0.000 000 000 029 103 830 456 733 703 613 281 25
68 719 476 736	36	0.000 000 000 014 551 915 228 366 851 806 640 625
137 438 953 472	37	0.000 000 000 007 275 957 614 183 425 903 320 312 5
274 877 906 944	38	0.000 000 000 003 637 978 807 091 712 951 660 156 25
549 755 813 888	39	0.000 000 000 001 818 989 403 545 856 475 830 078 125
1 099 511 627 776	40	0.000 000 000 000 909 494 701 772 928 237 915 039 062 5
2 199 023 255 552	41	0.000 000 000 000 454 747 350 886 464 118 957 519 531 25
4 398 046 511 104	42	0.000 000 000 000 227 373 675 443 232 059 478 759 765 625
8 796 093 022 208	43	0.000 000 000 000 113 686 837 721 616 029 739 379 882 812 5
17 592 186 044 416	44	0.000 000 000 000 056 843 418 860 808 014 869 689 941 406 25
35 184 372 088 832	45	0.000 000 000 000 028 421 709 430 404 007 434 844 970 703 125
70 368 744 177 664	46	0.000 000 000 000 014 210 854 715 202 003 717 422 485 351 562 5
140 737 488 355 328	47	0.000 000 000 000 007 105 427 357 601 001 858 711 242 675 781 25
281 474 976 710 656	48	0.000 000 000 000 003 552 713 678 800 500 929 355 621 337 890 625
562 949 953 421 312	49	0.000 000 000 000 001 776 356 839 400 250 464 677 810 668 945 312 5
1 125 899 906 842 624	50	0.000 000 000 000 000 888 178 419 700 125 232 338 905 334 472 656 25
2 251 799 813 685 248	51	0.000 000 000 000 000 444 089 209 850 062 616 169 452 667 236 328 125
4 503 599 627 370 496	52	0.000 000 000 000 000 222 044 604 925 031 308 084 726 333 618 164 062 5
9 007 199 254 740 992	53	0.000 000 000 000 000 111 022 302 462 515 654 042 363 166 809 082 031 25
18 014 398 509 481 984	54	0.000 000 000 000 000 055 511 151 231 257 827 021 181 583 404 541 015 625
36 028 797 018 963 968	55	0.000 000 000 000 000 027 755 575 615 628 913 510 590 791 702 270 507 812 5
72 057 594 037 927 936	56	0.000 000 000 000 000 013 877 787 807 814 456 755 295 395 851 135 253 906 25
144 115 188 075 855 872	57	0.000 000 000 000 000 006 938 893 903 907 228 377 647 697 925 567 626 953 125
288 230 376 151 711 744	58	0.000 000 000 000 000 003 469 446 951 953 614 188 823 848 962 783 813 476 562 5
576 460 752 303 423 488	59	0.000 000 000 000 000 001 734 723 475 976 807 094 411 924 481 391 906 738 281 25
1 152 921 504 606 846 976	60	0.000 000 000 000 000 000 867 361 737 988 403 547 205 962 240 695 953 369 140 625
2 305 843 009 213 693 952	61	0.000 000 000 000 000 000 433 680 868 994 201 773 602 981 120 347 976 684 570 312 5
4 611 686 018 427 387 904	62	0.000 000 000 000 000 000 216 840 434 497 100 886 801 490 560 173 988 342 285 156 25
9 223 372 036 854 775 808	63	0.000 000 000 000 000 000 108 420 217 248 550 443 400 745 280 086 994 171 142 578 125
18 446 744 073 709 551 616	64	0.000 000 000 000 000 000 054 210 108 624 275 221 700 372 640 043 497 085 571 289 062 5
36 893 488 147 419 103 232	65	0.000 000 000 000 000 000 027 105 054 312 137 610 850 186 320 021 748 542 785 644 531 25
73 786 976 294 838 206 464	66	0.000 000 000 000 000 000 013 552 527 156 068 805 425 093 160 010 874 271 392 822 265 625
147 573 952 589 676 412 928	67	0.000 000 000 000 000 000 006 776 263 578 034 402 712 546 580 005 437 135 696 411 132 812 5
295 147 905 179 352 825 856	68	0.000 000 000 000 000 000 003 388 131 789 017 201 356 273 290 002 718 567 848 205 566 406 25
590 295 810 358 705 651 712	69	0.000 000 000 000 000 000 001 694 065 894 508 600 678 136 645 001 359 283 924 102 783 203 125
1 180 591 620 717 411 303 424	70	0.000 000 000 000 000 000 000 847 032 947 254 300 339 068 322 500 679 641 962 051 391 601 562 5
2 361 183 241 434 822 606 848	71	0.000 000 000 000 000 000 000 423 516 473 627 150 169 534 161 250 339 820 981 025 695 800 781 25
4 722 366 482 869 645 213 696	72	0.000 000 000 000 000 000 000 211 758 236 813 575 084 767 080 625 169 910 490 512 847 900 390 625

Octal-Decimal Integer Conversion Table

0000 to 0777 (Octal)	0000 to 0511 (Decimal)

Octal	Decimal
10000	4096
20000	8192
30000	12288
40000	16384
50000	20480
60000	24576
70000	28672

	0	1	2	3	4	5	6	7
0000	0000	0001	0002	0003	0004	0005	0006	0007
0010	0008	0009	0010	0011	0012	0013	0014	0015
0020	0016	0017	0018	0019	0020	0021	0022	0023
0030	0024	0025	0026	0027	0028	0029	0030	0031
0040	0032	0033	0034	0035	0036	0037	0038	0039
0050	0040	0041	0042	0043	0044	0045	0046	0047
0060	0048	0049	0050	0051	0052	0053	0054	0055
0070	0056	0057	0058	0059	0060	0061	0062	0063
0100	0064	0065	0066	0067	0068	0069	0070	0071
0110	0072	0073	0074	0075	0076	0077	0078	0079
0120	0080	0081	0082	0083	0084	0085	0086	0087
0130	0088	0089	0090	0091	0092	0093	0094	0095
0140	0096	0097	0098	0099	0100	0101	0102	0103
0150	0104	0105	0106	0107	0108	0109	0110	0111
0160	0112	0113	0114	0115	0116	0117	0118	0119
0170	0120	0121	0122	0123	0124	0125	0126	0127
0200	0128	0129	0130	0131	0132	0133	0134	0135
0210	0136	0137	0138	0139	0140	0141	0142	0143
0220	0144	0145	0146	0147	0148	0149	0150	0151
0230	0152	0153	0154	0155	0156	0157	0158	0159
0240	0160	0161	0162	0163	0164	0165	0166	0167
0250	0168	0169	0170	0171	0172	0173	0174	0175
0260	0176	0177	0178	0179	0180	0181	0182	0183
0270	0184	0185	0186	0187	0188	0189	0190	0191
0300	0192	0193	0194	0195	0196	0197	0198	0199
0310	0200	0201	0202	0203	0204	0205	0206	0207
0320	0208	0209	0210	0211	0212	0213	0214	0215
0330	0216	0217	0218	0219	0220	0221	0222	0223
0340	0224	0225	0226	0227	0228	0229	0230	0231
0350	0232	0233	0234	0235	0236	0237	0238	0239
0360	0240	0241	0242	0243	0244	0245	0246	0247
0370	0248	0249	0250	0251	0252	0253	0254	0255

	0	1	2	3	4	5	6	7
0400	0256	0257	0258	0259	0260	0261	0262	0263
0410	0264	0265	0266	0267	0268	0269	0270	0271
0420	0272	0273	0274	0275	0276	0277	0278	0279
0430	0280	0281	0282	0283	0284	0285	0286	0287
0440	0288	0289	0290	0291	0292	0293	0294	0295
0450	0296	0297	0298	0299	0300	0301	0302	0303
0460	0304	0305	0306	0307	0308	0309	0310	0311
0470	0312	0313	0314	0315	0316	0317	0318	0319
0500	0320	0321	0322	0323	0324	0325	0326	0327
0510	0328	0329	0330	0331	0332	0333	0334	0335
0520	0336	0337	0338	0339	0340	0341	0342	0343
0530	0344	0345	0346	0347	0348	0349	0350	0351
0540	0352	0353	0354	0355	0356	0357	0358	0359
0550	0360	0361	0362	0363	0364	0365	0366	0367
0560	0368	0369	0370	0371	0372	0373	0374	0375
0570	0376	0377	0378	0379	0380	0381	0382	0383
0600	0384	0385	0386	0387	0388	0389	0390	0391
0610	0392	0393	0394	0395	0396	0397	0398	0399
0620	0400	0401	0402	0403	0404	0405	0406	0407
0630	0408	0409	0410	0411	0412	0413	0414	0415
0640	0416	0417	0418	0419	0420	0421	0422	0423
0650	0424	0425	0426	0427	0428	0429	0430	0431
0660	0432	0433	0434	0435	0436	0437	0438	0439
0670	0440	0441	0442	0443	0444	0445	0446	0447
0700	0448	0449	0450	0451	0452	0453	0454	0455
0710	0456	0457	0458	0459	0460	0461	0462	0463
0720	0464	0465	0466	0467	0468	0469	0470	0471
0730	0472	0473	0474	0475	0476	0477	0478	0479
0740	0480	0481	0482	0483	0484	0485	0486	0487
0750	0488	0489	0490	0491	0492	0493	0494	0495
0760	0496	0497	0498	0499	0500	0501	0502	0503
0770	0504	0505	0506	0507	0508	0509	0510	0511

1000 to 1777 (Octal)	0512 to 1023 (Decimal)

	0	1	2	3	4	5	6	7
1000	0512	0513	0514	0515	0516	0517	0518	0519
1010	0520	0521	0522	0523	0524	0525	0526	0527
1020	0528	0529	0530	0531	0532	0533	0534	0535
1030	0536	0537	0538	0539	0540	0541	0542	0543
1040	0544	0545	0546	0547	0548	0549	0550	0551
1050	0552	0553	0554	0555	0556	0557	0558	0559
1060	0560	0561	0562	0563	0564	0565	0566	0567
1070	0568	0569	0570	0571	0572	0573	0574	0575
1100	0576	0577	0578	0579	0580	0581	0582	0583
1110	0584	0585	0586	0587	0588	0589	0590	0591
1120	0592	0593	0594	0595	0596	0597	0598	0599
1130	0600	0601	0602	0603	0604	0605	0606	0607
1140	0608	0609	0610	0611	0612	0613	0614	0615
1150	0616	0617	0618	0619	0620	0621	0622	0623
1160	0624	0625	0626	0627	0628	0629	0630	0631
1170	0632	0633	0634	0635	0636	0637	0638	0639
1200	0640	0641	0642	0643	0644	0645	0646	0647
1210	0648	0649	0650	0651	0652	0653	0654	0655
1220	0656	0657	0658	0659	0660	0661	0662	0663
1230	0664	0665	0666	0667	0668	0669	0670	0671
1240	0672	0673	0674	0675	0676	0677	0678	0679
1250	0680	0681	0682	0683	0684	0685	0686	0687
1260	0688	0689	0690	0691	0692	0693	0694	0695
1270	0696	0697	0698	0699	0700	0701	0702	0703
1300	0704	0705	0706	0707	0708	0709	0710	0711
1310	0712	0713	0714	0715	0716	0717	0718	0719
1320	0720	0721	0722	0723	0724	0725	0726	0727
1330	0728	0729	0730	0731	0732	0733	0734	0735
1340	0736	0737	0738	0739	0740	0741	0742	0743
1350	0744	0745	0746	0747	0748	0749	0750	0751
1360	0752	0753	0754	0755	0756	0757	0758	0759
1370	0760	0761	0762	0763	0764	0765	0766	0767

	0	1	2	3	4	5	6	7
1400	0768	0769	0770	0771	0772	0773	0774	0775
1410	0776	0777	0778	0779	0780	0781	0782	0783
1420	0784	0785	0786	0787	0788	0789	0790	0791
1430	0792	0793	0794	0795	0796	0797	0798	0799
1440	0800	0801	0802	0803	0804	0805	0806	0807
1450	0808	0809	0810	0811	0812	0813	0814	0815
1460	0816	0817	0818	0819	0820	0821	0822	0823
1470	0824	0825	0826	0827	0828	0829	0830	0831
1500	0832	0833	0834	0835	0836	0837	0838	0839
1510	0840	0841	0842	0843	0844	0845	0846	0847
1520	0848	0849	0850	0851	0852	0853	0854	0855
1530	0856	0857	0858	0859	0860	0861	0862	0863
1540	0864	0865	0866	0867	0868	0869	0870	0871
1550	0872	0873	0874	0875	0876	0877	0878	0879
1560	0880	0881	0882	0883	0884	0885	0886	0887
1570	0888	0889	0890	0891	0892	0893	0894	0895
1600	0896	0897	0898	0899	0900	0901	0902	0903
1610	0904	0905	0906	0907	0908	0909	0910	0911
1620	0912	0913	0914	0915	0916	0917	0918	0919
1630	0920	0921	0922	0923	0924	0925	0926	0927
1640	0928	0929	0930	0931	0932	0933	0934	0935
1650	0936	0937	0938	0939	0940	0941	0942	0943
1660	0944	0945	0946	0947	0948	0949	0950	0951
1670	0952	0953	0954	0955	0956	0957	0958	0959
1700	0960	0961	0962	0963	0964	0965	0966	0967
1710	0968	0969	0970	0971	0972	0973	0974	0975
1720	0976	0977	0978	0979	0980	0981	0982	0983
1730	0984	0985	0986	0987	0988	0989	0990	0991
1740	0992	0993	0994	0995	0996	0997	0998	0999
1750	1000	1001	1002	1003	1004	1005	1006	1007
1760	1008	1009	1010	1011	1012	1013	1014	1015
1770	1016	1017	1018	1019	1020	1021	1022	1023

Octal-Decimal Integer Conversion Table (Cont)

	0	1	2	3	4	5	6	7
2000	1024	1025	1026	1027	1028	1029	1030	1031
2010	1032	1033	1034	1035	1036	1037	1038	1039
2020	1040	1041	1042	1043	1044	1045	1046	1047
2030	1048	1049	1050	1051	1052	1053	1054	1055
2040	1056	1057	1058	1059	1060	1061	1062	1063
2050	1064	1065	1066	1067	1068	1069	1070	1071
2060	1072	1073	1074	1075	1076	1077	1078	1079
2070	1080	1081	1082	1083	1084	1085	1086	1087
2100	1088	1089	1090	1091	1092	1093	1094	1095
2110	1096	1097	1098	1099	1100	1101	1102	1103
2120	1104	1105	1106	1107	1108	1109	1110	1111
2130	1112	1113	1114	1115	1116	1117	1118	1119
2140	1120	1121	1122	1123	1124	1125	1126	1127
2150	1128	1129	1130	1131	1132	1133	1134	1135
2160	1136	1137	1138	1139	1140	1141	1142	1143
2170	1144	1145	1146	1147	1148	1149	1150	1151
2200	1152	1153	1154	1155	1156	1157	1158	1159
2210	1160	1161	1162	1163	1164	1165	1166	1167
2220	1168	1169	1170	1171	1172	1173	1174	1175
2230	1176	1177	1178	1179	1180	1181	1182	1183
2240	1184	1185	1186	1187	1188	1189	1190	1191
2250	1192	1193	1194	1195	1196	1197	1198	1199
2260	1200	1201	1202	1203	1204	1205	1206	1207
2270	1208	1209	1210	1211	1212	1213	1214	1215
2300	1216	1217	1218	1219	1220	1221	1222	1223
2310	1224	1225	1226	1227	1228	1229	1230	1231
2320	1232	1233	1234	1235	1236	1237	1238	1239
2330	1240	1241	1242	1243	1244	1245	1246	1247
2340	1248	1249	1250	1251	1252	1253	1254	1255
2350	1256	1257	1258	1259	1260	1261	1262	1263
2360	1264	1265	1266	1267	1268	1269	1270	1271
2370	1272	1273	1274	1275	1276	1277	1278	1279

	0	1	2	3	4	5	6	7
2400	1280	1281	1282	1283	1284	1285	1286	1287
2410	1288	1289	1290	1291	1292	1293	1294	1295
2420	1296	1297	1298	1299	1300	1301	1302	1303
2430	1304	1305	1306	1307	1308	1309	1310	1311
2440	1312	1313	1314	1315	1316	1317	1318	1319
2450	1320	1321	1322	1323	1324	1325	1326	1327
2460	1328	1329	1330	1331	1332	1333	1334	1335
2470	1336	1337	1338	1339	1340	1341	1342	1343
2500	1344	1345	1346	1347	1348	1349	1350	1351
2510	1352	1353	1354	1355	1356	1357	1358	1359
2520	1360	1361	1362	1363	1364	1365	1366	1367
2530	1368	1369	1370	1371	1372	1373	1374	1375
2540	1376	1377	1378	1379	1380	1381	1382	1383
2550	1384	1385	1386	1387	1388	1389	1390	1391
2560	1392	1393	1394	1395	1396	1397	1398	1399
2570	1400	1401	1402	1403	1404	1405	1406	1407
2600	1408	1409	1410	1411	1412	1413	1414	1415
2610	1416	1417	1418	1419	1420	1421	1422	1423
2620	1424	1425	1426	1427	1428	1429	1430	1431
2630	1432	1433	1434	1435	1436	1437	1438	1439
2640	1440	1441	1442	1443	1444	1445	1446	1447
2650	1448	1449	1450	1451	1452	1453	1454	1455
2660	1456	1457	1458	1459	1460	1461	1462	1463
2670	1464	1465	1466	1467	1468	1469	1470	1471
2700	1472	1473	1474	1475	1476	1477	1478	1479
2710	1480	1481	1482	1483	1484	1485	1486	1487
2720	1488	1489	1490	1491	1492	1493	1494	1495
2730	1496	1497	1498	1499	1500	1501	1502	1503
2740	1504	1505	1506	1507	1508	1509	1510	1511
2750	1512	1513	1514	1515	1516	1517	1518	1519
2760	1520	1521	1522	1523	1524	1525	1526	1527
2770	1528	1529	1530	1531	1532	1533	1534	1535

2000 to 2777 (Octal)	1024 to 1535 (Decimal)

Octal	Decimal
10000	4096
20000	8192
30000	12288
40000	16384
50000	20480
60000	24576
70000	28672

	0	1	2	3	4	5	6	7
3000	1536	1537	1538	1539	1540	1541	1542	1543
3010	1544	1545	1546	1547	1548	1549	1550	1551
3020	1552	1553	1554	1555	1556	1557	1558	1559
3030	1560	1561	1562	1563	1564	1565	1566	1567
3040	1568	1569	1570	1571	1572	1573	1574	1575
3050	1576	1577	1578	1579	1580	1581	1582	1583
3060	1584	1585	1586	1587	1588	1589	1590	1591
3070	1592	1593	1594	1595	1596	1597	1598	1599
3100	1600	1601	1602	1603	1604	1605	1606	1607
3110	1608	1609	1610	1611	1612	1613	1614	1615
3120	1616	1617	1618	1619	1620	1621	1622	1623
3130	1624	1625	1626	1627	1628	1629	1630	1631
3140	1632	1633	1634	1635	1636	1637	1638	1639
3150	1640	1641	1642	1643	1644	1645	1646	1647
3160	1648	1649	1650	1651	1652	1653	1654	1655
3170	1656	1657	1658	1659	1660	1661	1662	1663
3200	1664	1665	1666	1667	1668	1669	1670	1671
3210	1672	1673	1674	1675	1676	1677	1678	1679
3220	1680	1681	1682	1683	1684	1685	1686	1687
3230	1688	1689	1690	1691	1692	1693	1694	1695
3240	1696	1697	1698	1699	1700	1701	1702	1703
3250	1704	1705	1706	1707	1708	1709	1710	1711
3260	1712	1713	1714	1715	1716	1717	1718	1719
3270	1720	1721	1722	1723	1724	1725	1726	1727
3300	1728	1729	1730	1731	1732	1733	1734	1735
3310	1736	1737	1738	1739	1740	1741	1742	1743
3320	1744	1745	1746	1747	1748	1749	1750	1751
3330	1752	1753	1754	1755	1756	1757	1758	1759
3340	1760	1761	1762	1763	1764	1765	1766	1767
3350	1768	1769	1770	1771	1772	1773	1774	1775
3360	1776	1777	1778	1779	1780	1781	1782	1783
3370	1784	1785	1786	1787	1788	1789	1790	1791

	0	1	2	3	4	5	6	7
3400	1792	1793	1794	1795	1796	1797	1798	1799
3410	1800	1801	1802	1803	1804	1805	1806	1807
3420	1808	1809	1810	1811	1812	1813	1814	1815
3430	1816	1817	1818	1819	1820	1821	1822	1823
3440	1824	1825	1826	1827	1828	1829	1830	1831
3450	1832	1833	1834	1835	1836	1837	1838	1839
3460	1840	1841	1842	1843	1844	1845	1846	1847
3470	1848	1849	1850	1851	1852	1853	1854	1855
3500	1856	1857	1858	1859	1860	1861	1862	1863
3510	1864	1865	1866	1867	1868	1869	1870	1871
3520	1872	1873	1874	1875	1876	1877	1878	1879
3530	1880	1881	1882	1883	1884	1885	1886	1887
3540	1888	1889	1890	1891	1892	1893	1894	1895
3550	1896	1897	1898	1899	1900	1901	1902	1903
3560	1904	1905	1906	1907	1908	1909	1910	1911
3570	1912	1913	1914	1915	1916	1917	1918	1919
3600	1920	1921	1922	1923	1924	1925	1926	1927
3610	1928	1929	1930	1931	1932	1933	1934	1935
3620	1936	1937	1938	1939	1940	1941	1942	1943
3630	1944	1945	1946	1947	1948	1949	1950	1951
3640	1952	1953	1954	1955	1956	1957	1958	1959
3650	1960	1961	1962	1963	1964	1965	1966	1967
3660	1968	1969	1970	1971	1972	1973	1974	1975
3670	1976	1977	1978	1979	1980	1981	1982	1983
3700	1984	1985	1986	1987	1988	1989	1990	1991
3710	1992	1993	1994	1995	1996	1997	1998	1999
3720	2000	2001	2002	2003	2004	2005	2006	2007
3730	2008	2009	2010	2011	2012	2013	2014	2015
3740	2016	2017	2018	2019	2020	2021	2022	2023
3750	2024	2025	2026	2027	2028	2029	2030	2031
3760	2032	2033	2034	2035	2036	2037	2038	2039
3770	2040	2041	2042	2043	2044	2045	2046	2047

3000 to 3777 (Octal)	1536 to 2047 (Decimal)

Octal-Decimal Integer Conversion Table (Cont)

4000 to 4777 (Octal)	2048 to 2559 (Decimal)

Octal	Decimal
10000	4096
20000	8192
30000	12288
40000	16384
50000	20480
60000	24576
70000	28672

	0	1	2	3	4	5	6	7
4000	2048	2049	2050	2051	2052	2053	2054	2055
4010	2056	2057	2058	2059	2060	2061	2062	2063
4020	2064	2065	2066	2067	2068	2069	2070	2071
4030	2072	2073	2074	2075	2076	2077	2078	2079
4040	2080	2081	2082	2083	2084	2085	2086	2087
4050	2088	2089	2090	2091	2092	2093	2094	2095
4060	2096	2097	2098	2099	2100	2101	2102	2103
4070	2104	2105	2106	2107	2108	2109	2110	2111
4100	2112	2113	2114	2115	2116	2117	2118	2119
4110	2120	2121	2122	2123	2124	2125	2126	2127
4120	2128	2129	2130	2131	2132	2133	2134	2135
4130	2136	2137	2138	2139	2140	2141	2142	2143
4140	2144	2145	2146	2147	2148	2149	2150	2151
4150	2152	2153	2154	2155	2156	2157	2158	2159
4160	2160	2161	2162	2163	2164	2165	2166	2167
4170	2168	2169	2170	2171	2172	2173	2174	2175
4200	2176	2177	2178	2179	2180	2181	2182	2183
4210	2184	2185	2186	2187	2188	2189	2190	2191
4220	2192	2193	2194	2195	2196	2197	2198	2199
4230	2200	2201	2202	2203	2204	2205	2206	2207
4240	2208	2209	2210	2211	2212	2213	2214	2215
4250	2216	2217	2218	2219	2220	2221	2222	2223
4260	2224	2225	2226	2227	2228	2229	2230	2231
4270	2232	2233	2234	2235	2236	2237	2238	2239
4300	2240	2241	2242	2243	2244	2245	2246	2247
4310	2248	2249	2250	2251	2252	2253	2254	2255
4320	2256	2257	2258	2259	2260	2261	2262	2263
4330	2264	2265	2266	2267	2268	2269	2270	2271
4340	2272	2273	2274	2275	2276	2277	2278	2279
4350	2280	2281	2282	2283	2284	2285	2286	2287
4360	2288	2289	2290	2291	2292	2293	2294	2295
4370	2296	2297	2298	2299	2300	2301	2302	2303

	0	1	2	3	4	5	6	7
4400	2304	2305	2306	2307	2308	2309	2310	2311
4410	2312	2313	2314	2315	2316	2317	2318	2319
4420	2320	2321	2322	2323	2324	2325	2326	2327
4430	2328	2329	2330	2331	2332	2333	2334	2335
4440	2336	2337	2338	2339	2340	2341	2342	2343
4450	2344	2345	2346	2347	2348	2349	2350	2351
4460	2352	2353	2354	2355	2356	2357	2358	2359
4470	2360	2361	2362	2363	2364	2365	2366	2367
4500	2368	2369	2370	2371	2372	2373	2374	2375
4510	2376	2377	2378	2379	2380	2381	2382	2383
4520	2384	2385	2386	2387	2388	2389	2390	2391
4530	2392	2393	2394	2395	2396	2397	2398	2399
4540	2400	2401	2402	2403	2404	2405	2406	2407
4550	2408	2409	2410	2411	2412	2413	2414	2415
4560	2416	2417	2418	2419	2420	2421	2422	2423
4570	2424	2425	2426	2427	2428	2429	2430	2431
4600	2432	2433	2434	2435	2436	2437	2438	2439
4610	2440	2441	2442	2443	2444	2445	2446	2447
4620	2448	2449	2450	2451	2452	2453	2454	2455
4630	2456	2457	2458	2459	2460	2461	2462	2463
4640	2464	2465	2466	2467	2468	2469	2470	2471
4650	2472	2473	2474	2475	2476	2477	2478	2479
4660	2480	2481	2482	2483	2484	2485	2486	2487
4670	2488	2489	2490	2491	2492	2493	2494	2495
4700	2496	2497	2498	2499	2500	2501	2502	2503
4710	2504	2505	2506	2507	2508	2509	2510	2511
4720	2512	2513	2514	2515	2516	2517	2518	2519
4730	2520	2521	2522	2523	2524	2525	2526	2527
4740	2528	2529	2530	2531	2532	2533	2534	2535
4750	2536	2537	2538	2539	2540	2541	2542	2543
4760	2544	2545	2546	2547	2548	2549	2550	2551
4770	2552	2553	2554	2555	2556	2557	2558	2559

5000 to 5777 (Octal)	2560 to 3071 (Decimal)

	0	1	2	3	4	5	6	7
5000	2560	2561	2562	2563	2564	2565	2566	2567
5010	2568	2569	2570	2571	2572	2573	2574	2575
5020	2576	2577	2578	2579	2580	2581	2582	2583
5030	2584	2585	2586	2587	2588	2589	2590	2591
5040	2592	2593	2594	2595	2596	2597	2598	2599
5050	2600	2601	2602	2603	2604	2605	2606	2607
5060	2608	2609	2610	2611	2612	2613	2614	2615
5070	2616	2617	2618	2619	2620	2621	2622	2623
5100	2624	2625	2626	2627	2628	2629	2630	2631
5110	2632	2633	2634	2635	2636	2637	2638	2639
5120	2640	2641	2642	2643	2644	2645	2646	2647
5130	2648	2649	2650	2651	2652	2653	2654	2655
5140	2656	2657	2658	2659	2660	2661	2662	2663
5150	2664	2665	2666	2667	2668	2669	2670	2671
5160	2672	2673	2674	2675	2676	2677	2678	2679
5170	2680	2681	2682	2683	2684	2685	2686	2687
5200	2688	2689	2690	2691	2692	2693	2694	2695
5210	2696	2697	2698	2699	2700	2701	2702	2703
5220	2704	2705	2706	2707	2708	2709	2710	2711
5230	2712	2713	2714	2715	2716	2717	2718	2719
5240	2720	2721	2722	2723	2724	2725	2726	2727
5250	2728	2729	2730	2731	2732	2733	2734	2735
5260	2736	2737	2738	2739	2740	2741	2742	2743
5270	2744	2745	2746	2747	2748	2749	2750	2751
5300	2752	2753	2754	2755	2756	2757	2758	2759
5310	2760	2761	2762	2763	2764	2765	2766	2767
5320	2768	2769	2770	2771	2772	2773	2774	2775
5330	2776	2777	2778	2779	2780	2781	2782	2783
5340	2784	2785	2786	2787	2788	2789	2790	2791
5350	2792	2793	2794	2795	2796	2797	2798	2799
5360	2800	2801	2802	2803	2804	2805	2806	2807
5370	2808	2809	2810	2811	2812	2813	2814	2815

	0	1	2	3	4	5	6	7
5400	2816	2817	2818	2819	2820	2821	2822	2823
5410	2824	2825	2826	2827	2828	2829	2830	2831
5420	2832	2833	2834	2835	2836	2837	2838	2839
5430	2840	2841	2842	2843	2844	2845	2846	2847
5440	2848	2849	2850	2851	2852	2853	2854	2855
5450	2856	2857	2858	2859	2860	2861	2862	2863
5460	2864	2865	2866	2867	2868	2869	2870	2871
5470	2872	2873	2874	2875	2876	2877	2878	2879
5500	2880	2881	2882	2883	2884	2885	2886	2887
5510	2888	2889	2890	2891	2892	2893	2894	2895
5520	2896	2897	2898	2899	2900	2901	2902	2903
5530	2904	2905	2906	2907	2908	2909	2910	2911
5540	2912	2913	2914	2915	2916	2917	2918	2919
5550	2920	2921	2922	2923	2924	2925	2926	2927
5560	2928	2929	2930	2931	2932	2933	2934	2935
5570	2936	2937	2938	2939	2940	2941	2942	2943
5600	2944	2945	2946	2947	2948	2949	2950	2951
5610	2952	2953	2954	2955	2956	2957	2958	2959
5620	2960	2961	2962	2963	2964	2965	2966	2967
5630	2968	2969	2970	2971	2972	2973	2974	2975
5640	2976	2977	2978	2979	2980	2981	2982	2983
5650	2984	2985	2986	2987	2988	2989	2990	2991
5660	2992	2993	2994	2995	2996	2997	2998	2999
5670	3000	3001	3002	3003	3004	3005	3006	3007
5700	3008	3009	3010	3011	3012	3013	3014	3015
5710	3016	3017	3018	3019	3020	3021	3022	3023
5720	3024	3025	3026	3027	3028	3029	3030	3031
5730	3032	3033	3034	3035	3036	3037	3038	3039
5740	3040	3041	3042	3043	3044	3045	3046	3047
5750	3048	3049	3050	3051	3052	3053	3054	3055
5760	3056	3057	3058	3059	3060	3061	3062	3063
5770	3064	3065	3066	3067	3068	3069	3070	3071

Octal-Decimal Integer Conversion Table (Cont)

	0	1	2	3	4	5	6	7
6000	3072	3073	3074	3075	3076	3077	3078	3079
6010	3080	3081	3082	3083	3084	3085	3086	3087
6020	3088	3089	3090	3091	3092	3093	3094	3095
6030	3096	3097	3098	3099	3100	3101	3102	3103
6040	3104	3105	3106	3107	3108	3109	3110	3111
6050	3112	3113	3114	3115	3116	3117	3118	3119
6060	3120	3121	3122	3123	3124	3125	3126	3127
6070	3128	3129	3130	3131	3132	3133	3134	3135
6100	3136	3137	3138	3139	3140	3141	3142	3143
6110	3144	3145	3146	3147	3148	3149	3150	3151
6120	3152	3153	3154	3155	3156	3157	3158	3159
6130	3160	3161	3162	3163	3164	3165	3166	3167
6140	3168	3169	3170	3171	3172	3173	3174	3175
6150	3176	3177	3178	3179	3180	3181	3182	3183
6160	3184	3185	3186	3187	3188	3189	3190	3191
6170	3192	3193	3194	3195	3196	3197	3198	3199
6200	3200	3201	3202	3203	3204	3205	3206	3207
6210	3208	3209	3210	3211	3212	3213	3214	3215
6220	3216	3217	3218	3219	3220	3221	3222	3223
6230	3224	3225	3226	3227	3228	3229	3230	3231
6240	3232	3233	3234	3235	3236	3237	3238	3239
6250	3240	3241	3242	3243	3244	3245	3246	3247
6260	3248	3249	3250	3251	3252	3253	3254	3255
6270	3256	3257	3258	3259	3260	3261	3262	3263
6300	3264	3265	3266	3267	3268	3269	3270	3271
6310	3272	3273	3274	3275	3276	3277	3278	3279
6320	3280	3281	3282	3283	3284	3285	3286	3287
6330	3288	3289	3290	3291	3292	3293	3294	3295
6340	3296	3297	3298	3299	3300	3301	3302	3303
6350	3304	3305	3306	3307	3308	3309	3310	3311
6360	3312	3313	3314	3315	3316	3317	3318	3319
6370	3320	3321	3322	3323	3324	3325	3326	3327

	0	1	2	3	4	5	6	7
6400	3328	3329	3330	3331	3332	3333	3334	3335
6410	3336	3337	3338	3339	3340	3341	3342	3343
6420	3344	3345	3346	3347	3348	3349	3350	3351
6430	3352	3353	3354	3355	3356	3357	3358	3359
6440	3360	3361	3362	3363	3364	3365	3366	3367
6450	3368	3369	3370	3371	3372	3373	3374	3375
6460	3376	3377	3378	3379	3380	3381	3382	3383
6470	3384	3385	3386	3387	3388	3389	3390	3391
6500	3392	3393	3394	3395	3396	3397	3398	3399
6510	3400	3401	3402	3403	3404	3405	3406	3407
6520	3408	3409	3410	3411	3412	3413	3414	3415
6530	3416	3417	3418	3419	3420	3421	3422	3423
6540	3424	3425	3426	3427	3428	3429	3430	3431
6550	3432	3433	3434	3435	3436	3437	3438	3439
6560	3440	3441	3442	3443	3444	3445	3446	3447
6570	3448	3449	3450	3451	3452	3453	3454	3455
6600	3456	3457	3458	3459	3460	3461	3462	3463
6610	3464	3465	3466	3467	3468	3469	3470	3471
6620	3472	3473	3474	3475	3476	3477	3478	3479
6630	3480	3481	3482	3483	3484	3485	3486	3487
6640	3488	3489	3490	3491	3492	3493	3494	3495
6650	3496	3497	3498	3499	3500	3501	3502	3503
6660	3504	3505	3506	3507	3508	3509	3510	3511
6670	3512	3513	3514	3515	3516	3517	3518	3519
6700	3520	3521	3522	3523	3524	3525	3526	3527
6710	3528	3529	3530	3531	3532	3533	3534	3535
6720	3536	3537	3538	3539	3540	3541	3542	3543
6730	3544	3545	3546	3547	3548	3549	3550	3551
6740	3552	3553	3554	3555	3556	3557	3558	3559
6750	3560	3561	3562	3563	3564	3565	3566	3567
6760	3568	3569	3570	3571	3572	3573	3574	3575
6770	3576	3577	3578	3579	3580	3581	3582	3583

6000 to 6777 (Octal)	3072 to 3583 (Decimal)

Octal	Decimal
10000	4096
20000	8192
30000	12288
40000	16384
50000	20480
60000	24576
70000	28672

	0	1	2	3	4	5	6	7
7000	3584	3585	3586	3587	3588	3589	3590	3591
7010	3592	3593	3594	3595	3596	3597	3598	3599
7020	3600	3601	3602	3603	3604	3605	3606	3607
7030	3608	3609	3610	3611	3612	3613	3614	3615
7040	3616	3617	3618	3619	3620	3621	3622	3623
7050	3624	3625	3626	3627	3628	3629	3630	3631
7060	3632	3633	3634	3635	3636	3637	3638	3639
7070	3640	3641	3642	3643	3644	3645	3646	3647
7100	3648	3649	3650	3651	3652	3653	3654	3655
7110	3656	3657	3658	3659	3660	3661	3662	3663
7120	3664	3665	3666	3667	3668	3669	3670	3671
7130	3672	3673	3674	3675	3676	3677	3678	3679
7140	3680	3681	3682	3683	3684	3685	3686	3687
7150	3688	3689	3690	3691	3692	3693	3694	3695
7160	3696	3697	3698	3699	3700	3701	3702	3703
7170	3704	3705	3706	3707	3708	3709	3710	3711
7200	3712	3713	3714	3715	3716	3717	3718	3719
7210	3720	3721	3722	3723	3724	3725	3726	3727
7220	3728	3729	3730	3731	3732	3733	3734	3735
7230	3736	3737	3738	3739	3740	3741	3742	3743
7240	3744	3745	3746	3747	3748	3749	3750	3751
7250	3752	3753	3754	3755	3756	3757	3758	3759
7260	3760	3761	3762	3763	3764	3765	3766	3767
7270	3768	3769	3770	3771	3772	3773	3774	3775
7300	3776	3777	3778	3779	3780	3781	3782	3783
7310	3784	3785	3786	3787	3788	3789	3790	3791
7320	3792	3793	3794	3795	3796	3797	3798	3799
7330	3800	3801	3802	3803	3804	3805	3806	3807
7340	3808	3809	3810	3811	3812	3813	3814	3815
7350	3816	3817	3818	3819	3820	3821	3822	3823
7360	3824	3825	3826	3827	3828	3829	3830	3831
7370	3832	3833	3834	3835	3836	3837	3838	3839

	0	1	2	3	4	5	6	7
7400	3840	3841	3842	3843	3844	3845	3846	3847
7410	3848	3849	3850	3851	3852	3853	3854	3855
7420	3856	3857	3858	3859	3860	3861	3862	3863
7430	3864	3865	3866	3867	3868	3869	3870	3871
7440	3872	3873	3874	3875	3876	3877	3878	3879
7450	3880	3881	3882	3883	3884	3885	3886	3887
7460	3888	3889	3890	3891	3892	3893	3894	3895
7470	3896	3897	3898	3899	3900	3901	3902	3903
7500	3904	3905	3906	3907	3908	3909	3910	3911
7510	3912	3913	3914	3915	3916	3917	3918	3919
7520	3920	3921	3922	3923	3924	3925	3926	3927
7530	3928	3929	3930	3931	3932	3933	3934	3935
7540	3936	3937	3938	3939	3940	3941	3942	3943
7550	3944	3945	3946	3947	3948	3949	3950	3951
7560	3952	3953	3954	3955	3956	3957	3958	3959
7570	3960	3961	3962	3963	3964	3965	3966	3967
7600	3968	3969	3970	3971	3972	3973	3974	3975
7610	3976	3977	3978	3979	3980	3981	3982	3983
7620	3984	3985	3986	3987	3988	3989	3990	3991
7630	3992	3993	3994	3995	3996	3997	3998	3999
7640	4000	4001	4002	4003	4004	4005	4006	4007
7650	4008	4009	4010	4011	4012	4013	4014	4015
7660	4016	4017	4018	4019	4020	4021	1022	4023
7670	4024	4025	4026	4027	4028	4029	4030	4031
7700	4032	4033	4034	4035	4036	4037	4038	4039
7710	4040	4041	4042	4043	4044	4045	4046	4047
7720	4048	4049	4050	4051	4052	4053	4054	4055
7730	4056	4057	4058	4059	4060	4061	4062	4063
7740	4064	4065	4066	4067	4068	4069	4070	4071
7750	4072	4073	4074	4075	4076	4077	4078	4079
7760	4080	4081	4082	4083	4084	4085	4086	4087
7770	4088	4089	4090	4091	4092	4093	4094	4095

7000 to 7777 (Octal)	3584 to 4095 (Decimal)

Octal-Decimal Fraction Conversion Table

Octal	Decimal	Octal	Decimal	Octal	Decimal	Octal	Decimal
.000	.000000	.100	.125000	.200	.250000	.300	.375000
.001	.001953	.101	.126953	.201	.251953	.301	.376953
.002	.003906	.102	.128906	.202	.253906	.302	.378906
.003	.005859	.103	.130859	.203	.255859	.303	.380859
.004	.007812	.104	.132812	.204	.257812	.304	.382812
.005	.009765	.105	.134765	.205	.259765	.305	.384765
.006	.011718	.106	.136718	.206	.261718	.306	.386718
.007	.013671	.107	.138671	.207	.263671	.307	.388671
.010	.015625	.110	.140625	.210	.265625	.310	.390625
.011	.017578	.111	.142578	.211	.267578	.311	.392578
.012	.019531	.112	.144531	.212	.269531	.312	.394531
.013	.021484	.113	.146484	.213	.271484	.313	.396484
.014	.023437	.114	.148437	.214	.273437	.314	.398437
.015	.025390	.115	.150390	.215	.275390	.315	.400390
.016	.027343	.116	.152343	.216	.277343	.316	.402343
.017	.029296	.117	.154296	.217	.279296	.317	.404296
.020	.031250	.120	.156250	.220	.281250	.320	.406250
.021	.033203	.121	.158203	.221	.283203	.321	.408203
.022	.035156	.122	.160156	.222	.285156	.322	.410156
.023	.037109	.123	.162109	.223	.287109	.323	.412109
.024	.039062	.124	.164062	.224	.289062	.324	.414062
.025	.041015	.125	.166015	.225	.291015	.325	.416015
.026	.042968	.126	.167968	.226	.292968	.326	.417968
.027	.044921	.127	.169921	.227	.294921	.327	.419921
.030	.046875	.130	.171875	.230	.296875	.330	.421875
.031	.048828	.131	.173828	.231	.298828	.331	.423828
.032	.050781	.132	.175781	.232	.300781	.332	.425781
.033	.052734	.133	.177734	.233	.302734	.333	.427734
.034	.054687	.134	.179687	.234	.304687	.334	.429687
.035	.056640	.135	.181640	.235	.306640	.335	.431640
.036	.058593	.136	.183593	.236	.308593	.336	.433593
.037	.060546	.137	.185546	.237	.310546	.337	.435546
.040	.062500	.140	.187500	.240	.312500	.340	.437500
.041	.064453	.141	.189453	.241	.314453	.341	.439453
.042	.066406	.142	.191406	.242	.316406	.342	.441406
.043	.068359	.143	.193359	.243	.318359	.343	.443359
.044	.070312	.144	.195312	.244	.320312	.344	.445312
.045	.072265	.145	.197265	.245	.322265	.345	.447265
.046	.074218	.146	.199218	.246	.324218	.346	.449218
.047	.076171	.147	.201171	.247	.326171	.347	.451171
.050	.078125	.150	.203125	.250	.328125	.350	.453125
.051	.080078	.151	.205078	.251	.330078	.351	.455078
.052	.082031	.152	.207031	.252	.332031	.352	.457031
.053	.083984	.153	.208984	.253	.333984	.353	.458984
.054	.085937	.154	.210937	.254	.335937	.354	.460937
.055	.087890	.155	.212890	.255	.337890	.355	.462890
.056	.089843	.156	.214843	.256	.339843	.356	.464843
.057	.091796	.157	.216796	.257	.341796	.357	.466796
.060	.093750	.160	.218750	.260	.343750	.360	.468750
.061	.095703	.161	.220703	.261	.345703	.361	.470703
.062	.097656	.162	.222656	.262	.347656	.362	.472656
.063	.099609	.163	.224609	.263	.349609	.363	.474609
.064	.101562	.164	.226562	.264	.351562	.364	.476562
.065	.103515	.165	.228515	.265	.353515	.365	.478515
.066	.105468	.166	.230468	.266	.355468	.366	.480468
.067	.107421	.167	.232421	.267	.357421	.367	.482421
.070	.109375	.170	.234375	.270	.359375	.370	.484375
.071	.111328	.171	.236328	.271	.361328	.371	.486328
.072	.113281	.172	.238281	.272	.363281	.372	.488281
.073	.115234	.173	.240234	.273	.365234	.373	.490234
.074	.117187	.174	.242187	.274	.367187	.374	.492187
.075	.119140	.175	.244140	.275	.369140	.375	.494140
.076	.121093	.176	.246093	.276	.371093	.376	.496093
.077	.123046	.177	.248046	.277	.373046	.377	.498046

Octal-Decimal Fraction Conversion Table (Cont)

Octal	Decimal	Octal	Decimal	Octal	Decimal	Octal	Decimal
.000000	.000000	.000100	.000244	.000200	.000488	.000300	.000732
.000001	.000003	.000101	.000247	.000201	.000492	.000301	.000736
.000002	.000007	.000102	.000251	.000202	.000495	.000302	.000740
.000003	.000011	.000103	.000255	.000203	.000499	.000303	.000743
.000004	.000015	.000104	.000259	.000204	.000503	.000304	.000747
.000005	.000019	.000105	.000263	.000205	.000507	.000305	.000751
.000006	.000022	.000106	.000267	.000206	.000511	.000306	.000755
.000007	.000026	.000107	.000270	.000207	.000514	.000307	.000759
.000010	.000030	.000110	.000274	.000210	.000518	.000310	.000762
.000011	.000034	.000111	.000278	.000211	.000522	.000311	.000766
.000012	.000038	.000112	.000282	.000212	.000526	.000312	.000770
.000013	.000041	.000113	.000286	.000213	.000530	.000313	.000774
.000014	.000045	.000114	.000289	.000214	.000534	.000314	.000778
.000015	.000049	.000115	.000293	.000215	.000537	.000315	.000782
.000016	.000053	.000116	.000297	.000216	.000541	.000316	.000785
.000017	.000057	.000117	.000301	.000217	.000545	.000317	.000789
.000020	.000061	.000120	.000305	.000220	.000549	.000320	.000793
.000021	.000064	.000121	.000308	.000221	.000553	.000321	.000797
.000022	.000068	.000122	.000312	.000222	.000556	.000322	.000801
.000023	.000072	.000123	.000316	.000223	.000560	.000323	.000805
.000024	.000076	.000124	.000320	.000224	.000564	.000324	.000808
.000025	.000080	.000125	.000324	.000225	.000568	.000325	.000812
.000026	.000083	.000126	.000328	.000226	.000572	.000326	.000816
.000027	.000087	.000127	.000331	.000227	.000576	.000327	.000820
.000030	.000091	.000130	.000335	.000230	.000579	.000330	.000823
.000031	.000095	.000131	.000339	.000231	.000583	.000331	.000827
.000032	.000099	.000132	.000343	.000232	.000587	.000332	.000831
.000033	.000102	.000133	.000347	.000233	.000591	.000333	.000835
.000034	.000106	.000134	.000350	.000234	.000595	.000334	.000839
.000035	.000110	.000135	.000354	.000235	.000598	.000335	.000843
.000036	.000114	.000136	.000358	.000236	.000602	.000336	.000846
.000037	.000118	.000137	.000362	.000237	.000606	.000337	.000850
.000040	.000122	.000140	.000366	.000240	.000610	.000340	.000854
.000041	.000125	.000141	.000370	.000241	.000614	.000341	.000858
.000042	.000129	.000142	.000373	.000242	.000617	.000342	.000862
.000043	.000133	.000143	.000377	.000243	.000621	.000343	.000865
.000044	.000137	.000144	.000381	.000244	.000625	.000344	.000869
.000045	.000141	.000145	.000385	.000245	.000629	.000345	.000873
.000046	.000144	.000146	.000389	.000246	.000633	.000346	.000877
.000047	.000148	.000147	.000392	.000247	.000637	.000347	.000881
.000050	.000152	.000150	.000396	.000250	.000640	.000350	.000885
.000051	.000156	.000151	.000400	.000251	.000644	.000351	.000888
.000052	.000160	.000152	.000404	.000252	.000648	.000352	.000892
.000053	.000164	.000153	.000408	.000253	.000652	.000353	.000896
.000054	.000167	.000154	.000411	.000254	.000656	.000354	.000900
.000055	.000171	.000155	.000415	.000255	.000659	.000355	.000904
.000056	.000175	.000156	.000419	.000256	.000663	.000356	.000907
.000057	.000179	.000157	.000423	.000257	.000667	.000357	.000911
.000060	.000183	.000160	.000427	.000260	.000671	.000360	.000915
.000061	.000186	.000161	.000431	.000261	.000675	.000361	.000919
.000062	.000190	.000162	.000434	.000262	.000679	.000362	.000923
.000063	.000194	.000163	.000438	.000263	.000682	.000363	.000926
.000064	.000198	.000164	.000442	.000264	.000686	.000364	.000930
.000065	.000202	.000165	.000446	.000265	.000690	.000365	.000934
.000066	.000205	.000166	.000450	.000266	.000694	.000366	.000938
.000067	.000209	.000167	.000453	.000267	.000698	.000367	.000942
.000070	.000213	.000170	.000457	.000270	.000701	.000370	.000946
.000071	.000217	.000171	.000461	.000271	.000705	.000371	.000949
.000072	.000221	.000172	.000465	.000272	.000709	.000372	.000953
.000073	.000225	.000173	.000469	.000273	.000713	.000373	.000957
.000074	.000228	.000174	.000473	.000274	.000717	.000374	.000961
.000075	.000232	.000175	.000476	.000275	.000720	.000375	.000965
.000076	.000236	.000176	.000480	.000276	.000724	.000376	.000968
.000077	.000240	.000177	.000484	.000277	.000728	.000377	.000972

Octal-Decimal Fraction Conversion Table (Cont)

Octal	Decimal	Octal	Decimal	Octal	Decimal	Octal	Decimal
.000400	.000976	.000500	.001220	.000600	.001464	.000700	.001708
.000401	.000980	.000501	.001224	.000601	.001468	.000701	.001712
.000402	.000984	.000502	.001228	.000602	.001472	.000702	.001716
.000403	.000988	.000503	.001232	.000603	.001476	.000703	.001720
.000404	.000991	.000504	.001235	.000604	.001480	.000704	.001724
.000405	.000995	.000505	.001239	.000605	.001483	.000705	.001728
.000406	.000999	.000506	.001243	.000606	.001487	.000706	.001731
.000407	.001003	.000507	.001247	.000607	.001491	.000707	.001735
.000410	.001007	.000510	.001251	.000610	.001495	.000710	.001739
.000411	.00101C	.000511	.001255	.000611	.001499	.000711	.001743
.000412	.001014	.000512	.001258	.000612	.001502	.000712	.001747
.000413	.001018	.000513	.001262	.000613	.001506	.000713	.001750
.000414	.001022	.000514	.001266	.000614	.001510	.000714	.001754
.000415	.001026	.000515	.001270	.000615	.001514	.000715	.001758
.000416	.001029	.000516	.001274	.000616	.001518	.000716	.001762
.000417	.001033	.000517	.001277	.000617	.001522	.000717	.001766
.000420	.001037	.000520	.001281	.000620	.001525	.000720	.001770
.000421	.001041	.000521	.001285	.000621	.001529	.000721	.001773
.000422	.001045	.000522	.001289	.000622	.001533	.000722	.001777
.000423	.001049	.000523	.001293	.000623	.001537	.000723	.001781
.000424	.001052	.000524	.001296	.000624	.001541	.000724	.001785
.000425	.001056	.000525	.001300	.000625	.001544	.000725	.001789
.000426	.001060	.000526	.001304	.000626	.001548	.000726	.001792
.000427	.001064	.000527	.001308	.000627	.001552	.000727	.001796
.000430	.001068	.000530	.001312	.000630	.001556	.000730	.001800
.000431	.001071	.000531	.001316	.000631	.001560	.000731	.001804
.000432	.001075	.000532	.001319	.000632	.001564	.000732	.001808
.000433	.001079	.000533	.001323	.000633	.001567	.000733	.001811
.000434	.001083	.000534	.001327	.000634	.001571	.000734	.001815
.000435	.001087	.000535	.001331	.000635	.001575	.000735	.001819
.000436	.001091	.000536	.001335	.000636	.001579	.000736	.001823
.000437	.001094	.000537	.001338	.000637	.001583	.000737	.001827
.000440	.001098	.000540	.001342	.000640	.001586	.000740	.001831
.000441	.001102	.000541	.001346	.000641	.001590	.000741	.001834
.000442	.001106	.000542	.001350	.000642	.001594	.000742	.001838
.000443	.001110	.000543	.001354	.000643	.001598	.000743	.001842
.000444	.001113	.000544	.001358	.000644	.001602	.000744	.001846
.000445	.001117	.000545	.001361	.000645	.001605	.000745	.001850
.000446	.001121	.000546	.001365	.000646	.001609	.000746	.001853
.000447	.001125	.000547	.001369	.000647	.001613	.000747	.001857
.000450	.001129	.000550	.001373	.000650	.001617	.000750	.001861
.000451	.001132	.000551	.001377	.000651	.001621	.000751	.001865
.000452	.001136	.000552	.001380	.000652	.001625	.000752	.001869
.000453	.001140	.000553	.001384	.000653	.001628	.000753	.001873
.000454	.001144	.000554	.001388	.000654	.001632	.000754	.001876
.000455	.001148	.000555	.001392	.000655	.001636	.000755	.001880
.000456	.001152	.000556	.001396	.000656	.001640	.000756	.001884
.000457	.001155	.000557	.001399	.000657	.001644	.000757	.001888
.000460	.001159	.000560	.001403	.000660	.001647	.000760	.001892
.000461	.001163	.000561	.001407	.000661	.001651	.000761	.001895
.000462	.001167	.000562	.001411	.000662	.001655	.000762	.001899
.000463	.001171	.000563	.001415	.000663	.001659	.000763	.001903
.000464	.001174	.000564	.001419	000664	.001663	.000764	.001907
.000465	.001178	.000565	.001422	.000665	.001667	.000765	.001911
.000466	.001182	.000566	.001426	000666	.001670	.000766	.001914
.000467	.001186	.000567	.001430	000667	.001674	.000767	.001918
.000470	.001190	.000570	.001434	.000670	.001678	.000770	.001922
.000471	.001194	.000571	.001438	.000671	.001682	.000771	.001926
.000472	.001197	.000572	.001441	.000672	.001686	.000772	.001930
.000473	.001201	.000573	.001445	.000673	.001689	.000773	.001934
.000474	.001205	.000574	.001449	.000674	.001693	.000774	.001937
.000475	.001209	.000575	.001453	.000675	.001697	.000775	.001941
.000476	.001213	.000576	.001457	.000676	.001701	.000776	.001945
.000477	.001216	.000577	.001461	.000677	.001705	.000777	.001949

APPENDIX D
INSTRUCTION SET

This appendix contains a complete list of the PDP16-M instructions. The instruction mnemonics and the octal machine codes are given. To facilitate quick reference, the instructions are ordered by class of instruction (basic classes and optional instructions), rather than by machine code.

ARITHMETIC GROUP

Mnemonic	Octal Machine Code
A = 0	000
A = B	001
A = A+1	002
A = A-1	003
A = A+B	004
A = A-B	005
A = A/2	012
A = AX2	013
A = B/2	272
A = A+1(S)	014
A = A-1(S)	015
A = A+B(S)	016
A = A-B(S)	017
A = A/2(S)	020
A = AX2(S)	021
A = B/2(S)	273
B = 0	022
B = A	023
B = A+1	257
B = A-1	261
B = A+B	024
B = A-B	025
B = A/2	265
B = AX2	263
B = B/2	032

Mnemonic	Octal Machine Code
B = A+1(S)	260
B = A-1(S)	262
B = A+B(S)	033
B = A-B(S)	034
B = A/2(S)	266
B = AX2(S)	264
B = B/2(S)	035
L = 0	074
L = 1	075
L = LNOT	270
L = OVF	267

TEST GROUP

Mnemonic	Octal Machine Code
EXA	036
EXB	037

LOGICAL GROUP

Mnemonic	Octal Machine Code
A = ANOT	011
A = AORB	007
A = AB	010
A = AXORB	006
B = BNOT	031
B = AORB	027
B = AB	030
B = AXORB	026

REGISTER GROUP

Mnemonic	Octal Machine Code
A = TR	133
A = TRU	131
A = TRL	132
TR = 0	250
TR = A	130
TRU = A	276
TRL = A	277

CONSTANT GROUP

Mnemonic	Octal Machine Code
B = C1	046
B = C2	047
B = C3	136
B = C4	137

BOOLEAN OUTPUT GROUP

Mnemonic	Octal Machine Code
FF1 = 0	056
FF1 = 1	057
FF2 = 0	060
FF2 = 1	061
FF3 = 0	062
FF3 = 1	063

PARALLEL I/O GROUP

Mnemonic	Octal Machine Code
A = GPI1	041
B = GPI1	256
GPI1 = A	040
GPI1 = B	255

COMMAND GROUP

Mnemonic	Octal Machine Code
PAGE0	072
PAGE1	073
MUX0	076
MUX1	077
HALT	271

CONDITIONAL JUMP GROUP (MUX0)

	Mnemonic	Octal Machine Code Into or Within MEM0	MEM1
	GOTO	300	301
	IF EXT1,	302	303
	IF EXT2,	304	305
	IF EXT3,	306	307
	IF EXT4,	310	311
	IF EXT5,	312	313
	IF EXT6,	314	315
	IF DZ,	316	317
	IF DP,	320	321
	IF DN,	322	323
	IF OVF,	324	325
	IF A<1>,	326	327
	IF A<3>,	330	331
	IF A<5>,	332	333
	IF A<7>,	334	335
	IF A<9>,	336	337
	IF A<11>,	340	341
	IF A<13>,	342	343
	IF A<15>,	344	345
	IF FF1,	346	347
	IF FF2,	350	351
	IF FF3,	352	353
a)	IF FF4,	354	355
a)	IF FF5,	356	357
a)	IF FF6,	360	361
b)	IF KF1,	362	363
b)	IF PF1,	364	365
c)	IF KF2,	366	367
c)	IF PF2,	370	371
	IF CLK,	372	373

a) These instructions are implemented only if Boolean output flag option KFL16 is installed.

b) These instructions are implemented only if serial interface option DC16-A is installed.

c) These instructions are implemented only if serial interface 2 option DC16-A is installed.

SUBROUTINE GROUP

Mnemonic	Octal Machine Code Into or Within	
	MEM0	MEM1
CALL	374	375
RETURN	376	376

SCRATCHPAD REGISTER OPTION MS16-C

Mnemonic	Octal Machine Code
A = SP1	140
A = SP2	141
A = SP3	142
A = SP4	143
A = SP5	144
A = SP6	145
A = SP7	146
A = SP8	147
A = SP9	150
A = SP10	151
A = SP11	152
A = SP12	153
A = SP13	154
A = SP14	155
A = SP15	156
A = SP16	157
SP1 = A	160
SP2 = A	161
SP3 = A	162
SP4 = A	163
SP5 = A	164
SP6 = A	165
SP7 = A	166
SP8 = A	167
SP9 = A	170
SP10 = A	171
SP11 = A	172
SP12 = A	173
SP13 = A	174
SP14 = A	175
SP15 = A	176
SP16 = A	177
A = SP17	200
A = SP18	201
A = SP19	202

Mnemonic	Octal Machine Code
A = SP20	203
A = SP21	204
A = SP22	205
A = SP23	206
A = SP24	207
A = SP25	210
A = SP26	211
A = SP27	212
A = SP28	213
A = SP29	214
A = SP30	215
A = SP31	216
A = SP32	217
SP17 = A	220
SP18 = A	221
SP19 = A	222
SP20 = A	223
SP21 = A	224
SP22 = A	225
SP23 = A	226
SP24 = A	227
SP25 = A	230
SP26 = A	231
SP27 = A	232
SP28 = A	233
SP29 = A	234
SP30 = A	235
SP31 = A	236
SP32 = A	237

CONSTANT GENERATOR OPTION MR16-D

Mnemonic	Octal Machine Code
B = K1	100
B = K2	101
B = K3	102
B = K4	103
B = K5	104
B = K6	105
B = K7	106
B = K8	107
B = K9	110
B = K10	111
B = K11	112
B = K12	113
B = K13	114

Mnemonic	Octal Machine Code
B = K14	115
B = K15	116
B = K16	117
B = K17	120
B = K18	121
B = K19	122
B = K20	123
B = K21	124
B = K22	125
B = K23	126
B = K24	127

DATA PROM OPTION MR16-E/F

Mnemonic	Octal Machine Code
RMAR = A	134
RMAR = B	246
A = ROM	135
B = ROM	247

DATA READ/WRITE MEMORY OPTION MS16-D/E

Mnemonic	Octal Machine Code
MAR1 = A	240
MAR2 = A	243
MEM1 = A	241
MEM2 = A	244
A = MEM1	242
A = MEM2	245

PARALLEL I/O OPTION DB16-A

Mnemonic	Octal Machine Code
A = GPI2	043
A = GPI3	045
GPI2 = A	042
GPI3 = A	044

PDP-11 PERIPHERAL INTERFACE OPTION DA16-F

Mnemonic	Octal Machine Code
FF1 = 0	056
FF1 = 1	057
GPI1 = A	040
GPI1 = B	255
DATO	275
DATI	274

SERIAL I/O OPTION DC16-A

Mnemonic	Octal Machine Code
TAPE1	052
IF KF1,	See Conditional Jump Group
A = SI1	051
SI1 = A	050
IF PF1,	See Conditional Jump Group
TAPE2	055
IF KF2,	See Conditional Jump Group
A = SI2	054
SI2 = A	053
IF PF2,	See Conditional Jump Group

BOOLEAN OUTPUT OPTION KFL16

Mnemonic	Octal Machine Code
FF4 = 0	064
FF4 = 1	065
FF5 = 0	066
FF5 = 1	067
FF6 = 0	070
FF6 = 1	071

BOOLEAN INPUT MULTIPLEXER (MUX1) OPTION PCS16-D

Mnemonic	Octal Machine Code Into or Within	
	MEM0	MEM1
GOTO	300	301
IF A<0>,	302	303
IF A<2>,	304	305
IF A<4>,	306	307
IF A<6>,	310	311
IF A<8>,	312	313
IF A<10>,	314	315
IF A<12>,	316	317
IF A<14>,	320	321
IF B<0>,	322	323
If B<15>,	324	325
IF EXT7,	326	327
IF EXT8,	330	331
IF EXT9,	332	333
IF EXT10,	334	335
IF EXT11,	336	337
IF EXT12,	340	341
IF EXT13,	342	343
IF EXT14,	344	345
IF EXT15,	346	347
IF EXT16,	350	351
IF EXT17,	352	353
IF EXT18,	354	355
IF EXT19,	356	357
IF EXT20,	360	361
IF EXT21,	362	363
IF EXT22,	364	365
IF L,	366	367
IF PWOK	370	371

APPENDIX E
PDP16-M MACHINE CODES

This is a list, in octal machine code order, of all PDP16-M machine instructions. The basic PDP16-M hardware consists of:

Quantity	Model No.	Name	Module No.	Slot
1	KBS16-A	Bus Control	M7332	AB2
1	KAC16	GPA Control Unit	M7300	AB11
1	KAR16	GPA Register Unit	M7301	AB10
1	MR16-A	Four Word Constants Generator	M7307	AB5
1	MS16-A	Transfer Register	M7305	AB9
1	DB16-A	General Purpose Interface 1	M7311	AB12
1	PCS16-A	PCS Control	M7336	AB20
1	PCS16-B	PCS Control PROM	M7327	C16
4	PCS16-C	Evoke Decoders (0, 1, 2, 5)	M7328	CD3, 4, 5, 8
1	PCS16-D	MUX0	M7329	AB18
2	KFL16	Flags (1, 2, 3)	M7306	C2

REGISTER TRANSFER INSTRUCTIONS (EVOKE)

Machine Code (Octal)	Time (μsec)	Instruction	Evoke Decoder No.	Evoke Channel Decoder	Option	Slot
000	2.4	A = 0	0	0		
001	2.4	A = B	0	1		
002	2.4	A = A+1	0	2		
003	2.4	A = A-1	0	3		
004	2.4	A = A+B	0	4		
005	2.4	A = A-B	0	5		
006	2.4	A = AXORB	0	6		
007	2.4	A = AORB	0	7		
010	2.4	A = AB	0	10		
011	2.4	A = ANOT	0	11		
012	2.4	A = A/2	0	12		
013	2.4	A = AX2	0	13		
014	2.4	A = A+1(S)	0	14		
015	2.4	A = A-1(S)	0	15		

REGISTER TRANSFER INSTRUCTIONS (EVOKE) (Cont)

Machine Code (Octal)	Time (μsec)	Instruction	Evoke Decoder No.	Evoke Channel Decoder	Option	Slot
016	2.4	A = A+B(S)	0	16		
017	2.4	A = A-B(S)	0	17		
020	2.4	A = A/2(S)	0	20		
021	2.4	A = AX2(S)	0	21		
022	2.4	B = 0	0	22		
023	2.4	B = A	0	23		
024	2.4	B = A+B	0	24		
025	2.4	B = A-B	0	25		
026	2.4	B = AXORB	0	26		
027	2.4	B = AORB	0	27		
030	2.4	B = AB	0	30		
031	2.4	B = BNOT	0	31		
032	2.4	B = B/2	0	32		
033	2.4	B = A+B(S)	0	33		
034	2.4	B = A-B(S)	0	34		
035	2.4	B = B/2(S)	0	35		
036	2.4	BS = A	0	36		
037	2.4	BS = B	0	37		
040	2.2	GPI1 = A	1	0		
041	2.0	A = GPI1	1	1		
042	2.2	GPI2 = A	1	2	DB16-A	AB13
043	2.0	A = GPI2	1	3	DB16-A	AB13
044	2.2	GPI3 = A	1	4	DB16-A	AB14
045	2.0	A = GPI3	1	5	DB16-A	AB14
046	2.4	B = C1	1	6		
047	2.4	B = C2	1	7		
050	3.5	SI1 = A	1	10	DC16-A	AB16
051	2.1	A = SI1	1	11	DC16-A	AB16
052	1.7	TAPE1	1	12	DC16-A	AB16
053	3.5	SI2 = A	1	13	DC16-A	AB17
054	2.1	A = SI2	1	14	DC16-A	AB17
055	1.7	TAPE2	1	15	DC16-A	AB17
056	1.8	FF1 = 0	1	16		
057	1.8	FF1 = 1	1	17		
060	1.8	FF2 = 0	1	20		
061	1.8	FF2 = 1	1	21		
062	1.8	FF3 = 0	1	22		
063	1.8	FF3 = 1	1	23		
064	1.8	FF4 = 0	1	24	KLF16	D2
065	1.8	FF4 = 1	1	25	KLF16	D2
066	1.8	FF5 = 0	1	26	KLF16	D2
067	1.8	FF5 = 1	1	27	KLF16	D2
070	1.8	FF6 = 0	1	30	KLF16	D2
071	1.8	FF6 = 1	1	31	KLF16	D2

REGISTER TRANSFER INSTRUCTIONS (EVOKE) (Cont)

Machine Code (Octal)	Time (μsec)	Instruction	Evoke Decoder No.	Evoke Channel Decoder	Option	Slot
072	1.8	PAGE0	1	32	More Than 2 PCS16-B's	CD16 CD17
073	1.8	PAGE1	1	33	More Than 2 PCS16-B's	CD16 CD17
074	1.8	L = 0	1	34		
075	1.8	L = 1	1	35		
076	1.8	MUX0	1	36		
077	1.8	MUX1	1	37	PCS16-D	AB19
100	2.4	B = K1	2	0	MR16-D	AB8
101	2.4	B = K2	2	1	MR16-D	AB8
102	2.4	B = K3	2	2	MR16-D	AB8
103	2.4	B = K4	2	3	MR16-D	AB8
104	2.4	B = K5	2	4	MR16-D	AB8
105	2.4	B = K6	2	5	MR16-D	AB8
106	2.4	B = K7	2	6	MR16-D	AB8
107	2.4	B = K8	2	7	MR16-D	AB8
110	2.4	B = K9	2	10	MR16-D	AB8
111	2.4	B = K10	2	11	MR16-D	AB8
112	2.4	B = K11	2	12	MR16-D	AB8
113	2.4	B = K12	2	13	MR16-D	AB8
114	2.4	B = K13	2	14	MR16-D	AB8
115	2.4	B = K14	2	15	MR16-D	AB8
116	2.4	B = K15	2	16	MR16-D	AB8
117	2.4	B = K16	2	17	MR16-D	AB8
120	2.4	B = K17	2	20	MR16-D	AB8
121	2.4	B = K18	2	21	MR16-D	AB8
122	2.4	B = K19	2	22	MR16-D	AB8
123	2.4	B = K20	2	23	MR16-D	AB8
124	2.4	B = K21	2	24	MR16-D	AB8
125	2.4	B = K22	2	25	MR16-D	AB8
126	2.4	B = K23	2	26	MR16-D	AB8
127	2.4	B = K24	2	27	MR16-D	AB8
130	2.3	TR = A	2	30		
131	2.1	A = TRU	2	31		
132	2.1	A = TRL	2	32		
133	2.1	A = TR	2	33		
134	2.2	RMAR = A	2	34	PCS16-B and DB16-A	ABCD15
135	2.3	A = ROM	2	35	PCS16-B and DB16-A	ABCD15
136	2.4	B = C3	2	36		
137	2.4	B = C4	2	37		
140	2.2	A = SP1	3	0	MS16-C	AB4
141	2.2	A = SP2	3	1	MS16-C	AB4
142	2.2	A = SP3	3	2	MS16-C	AB4
143	2.2	A = SP4	3	3	MS16-C	AB4

REGISTER TRANSFER INSTRUCTIONS (EVOKE) (Cont)

Machine Code (Octal)	Time (μsec)	Instruction	Evoke Decoder No.	Evoke Channel Decoder	Option	Slot
144	2.2	A = SP5	3	4	MS16-C	AB4
145	2.2	A = SP6	3	5	MS16-C	AB4
146	2.2	A = SP7	3	6	MS16-C	AB4
147	2.2	A = SP8	3	7	MS16-C	AB4
150	2.2	A = SP9	3	10	MS16-C	AB4
151	2.2	A = SP10	3	11	MS16-C	AB4
152	2.2	A = SP11	3	12	MS16-C	AB4
153	2.2	A = SP12	3	13	MS16-C	AB4
154	2.2	A = SP13	3	14	MS16-C	AB4
155	2.2	A = SP14	3	15	MS16-C	AB4
156	2.2	A = SP15	3	16	MS16-C	AB4
157	2.2	A = SP16	3	17	MS16-C	AB4
160	2.6	SP1 = A	3	20	MS16-C	AB4
161	2.6	SP2 = A	3	21	MS16-C	AB4
162	2.6	SP3 = A	3	22	MS16-C	AB4
163	2.6	SP4 = A	3	23	MS16-C	AB4
164	2.6	SP5 = A	3	24	MS16-C	AB4
165	2.6	SP6 = A	3	25	MS16-C	AB4
166	2.6	SP7 = A	3	26	MS16-C	AB4
167	2.6	SP8 = A	3	27	MS16-C	AB4
170	2.6	SP9 = A	3	30	MS16-C	AB4
171	2.6	SP10 = A	3	31	MS16-C	AB4
172	2.6	SP12 = A	3	32	MS16-C	AB4
173	2.6	SP13 = A	3	33	MS16-C	AB4
174	2.6	SP13 = A	3	34	MS16-C	AB4
175	2.6	SP14 = A	3	35	MS16-C	AB4
176	2.6	SP15 = A	3	36	MS16-C	AB4
177	2.6	SP16 = A	3	37	MS16-C	AB4
200	2.2	A = SP17	4	0	MS16-C	AB3
201	2.2	A = SP18	4	1	MS16-C	AB3
202	2.2	A = SP19	4	2	MS16-C	AB3
203	2.2	A = SP20	4	3	MS16-C	AB3
204	2.2	A = SP21	4	4	MS16-C	AB3
205	2.2	A = SP22	4	5	MS16-C	AB3
206	2.2	A = SP23	4	6	MS16-C	AB3
207	2.2	A = SP24	4	7	MS16-C	AB3
210	2.2	A = SP25	4	10	MS16-C	AB3
211	2.2	A = SP26	4	11	MS16-C	AB3
212	2.2	A = SP27	4	12	MS16-C	AB3
213	2.2	A = SP28	4	13	MS16-C	AB3
214	2.2	A = SP29	4	14	MS16-C	AB3
215	2.2	A = SP30	4	15	MS16-C	AB3
216	2.2	A = SP31	4	16	MS16-C	AB3
217	2.2	A = SP32	4	17	MS16-C	AB3
220	2.6	SP17 = A	4	20	MS16-C	AB3
221	2.6	SP18 = A	4	21	MS16-C	AB3

REGISTER TRANSFER INSTRUCTIONS (EVOKE) (Cont)

Machine Code (Octal)	Time (μsec)	Instruction	Evoke Decoder No.	Evoke Channel Decoder	Option	Slot
222	2.6	SP19 = A	4	22	MS16-C	AB3
223	2.6	SP20 = A	4	23	MS16-C	AB3
224	2.6	SP21 = A	4	24	MS16-C	AB3
225	2.6	SP22 = A	4	25	MS16-C	AB3
226	2.6	SP23 = A	4	26	MS16-C	AB3
227	2.6	SP24 = A	4	27	MS16-C	AB3
230	2.6	SP25 = A	4	30	MS16-C	AB3
231	2.6	SP26 = A	4	31	MS16-C	AB3
232	2.6	SP27 = A	4	32	MS16-C	AB3
233	2.6	SP28 = A	4	33	MS16-C	AB3
234	2.6	SP29 = A	4	34	MS16-C	AB3
235	2.6	SP30 = A	4	35	MS16-C	AB3
236	2.6	SP31 = A	4	36	MS16-C	AB3
237	2.6	SP32 = A	4	37	MS16-C	AB3
240	2.3	MAR1 = A	5	0	MS16-D or E	AB6
241	3.6	MEM1 = A	5	1	MS16-D or E	AB6
242	3.9	A = MEM1	5	2	MS16-D or E	AB6
243	2.3	MAR2 = A	5	3	MS16-D or E	AB7
244	3.6	MEM2 = A	5	4	MS16-D or E	AB7
245	3.9	A = MEM2	5	5	MS16-D or E	AB7
246	2.2	RMAR = B	5	6	PCS16-B and DB16-A	ABCD15
247	2.3	B = ROM	5	7	PCS16-B and DB16-A	ABCD15
250	2.0	TR = 0	5	10		
251	2.1	B = SI1	5	11	DC16-A	AB16
252	3.5	SI1 = B	5	12	DC16-A	AB16
253	2.1	B = SI2	5	13	DC16-A	AB17
254	3.5	SI2 = B	5	14	DC16-A	AB17
255	2.2	GPI1 = B	5	15		
256	2.0	B = GPI1	5	16		
257	2.4	B = A+1	5	17		
260	2.4	B = A+1(S)	5	20		
261	2.4	B = A-1	5	21		
262	2.4	B = A-1(S)	5	22		
263	2.4	B = AX2	5	23		
264	2.4	B = AX2(S)	5	24		
265	2.4	B = A/2	5	25		
266	2.4	B = A/2(S)	5	26		
267	1.8	L = OVF	5	27		
270	1.8	L = LNOT	5	30		
271	–	HALT	5	31		
272	2.4	A = B/2	5	32		
273	2.4	A = B/2(S)	5	33		
274	–	DATI	5	34		
275	–	DATO	5	35		
276	2.3	TRU = A	5	36		
277	2.3	TRL = A	5	37		

CONDITIONAL JUMP INSTRUCTIONS (INPUT MULTIPLEXER)

Machine Code (Octal)*		Time (μsec)		Instruction	Multiplexer No.**	Multiplexer Channel
MEM0	MEM1	False	True			
300	301	2.0	3.2	GOTO	0	00
302	303	2.0	3.2	IF EXT1,	0	01
304	305	2.0	3.2	IF EXT2,	0	02
306	307	2.0	3.2	IF EXT3,	0	03
310	311	2.0	3.2	IF EXT4,	0	04
312	313	2.0	3.2	IF EXT5,	0	05
314	315	2.0	3.2	IF EXT6,	0	06
316	317	2.0	3.2	IF DZ,	0	07
320	321	2.0	3.2	IF DP,	0	10
322	323	2.0	3.2	IF DN,	0	11
324	325	2.0	3.2	IF OVF,	0	12
326	327	2.0	3.2	IF A<1>,	0	13
330	331	2.0	3.2	IF A<3>,	0	14
332	333	2.0	3.2	IF A<5>,	0	15
334	335	2.0	3.2	IF A<7>,	0	16
336	337	2.0	3.2	IF A<9>,	0	17
340	341	2.0	3.2	IF A<11>,	0	20
342	343	2.0	3.2	IF A<13>,	0	21
344	345	2.0	3.2	IF A<15>,	0	22
346	347	2.0	3.2	IF FF1	0	23
350	351	2.0	3.2	IF FF2	0	24
352	353	2.0	3.2	IF FF3	0	25
354	355	2.0	3.2	IF FF4	0	26
356	357	2.0	3.2	IF FF5	0	27
360	361	2.0	3.2	IF FF6	0	30
362	363	2.0	3.2	IF KF1	0	31
364	365	2.0	3.2	IF PF1	0	32
366	367	2.0	3.2	IF KF2	0	33
370	371	2.0	3.2	IF PF2	0	34
372	373	2.0	3.2	IF CLK	0	35
300	301	2.0	3.2	GOTO	1	00
302	303	2.0	3.2	IF A<0>	1	01
304	305	2.0	3.2	IF A<2>	1	02
306	307	2.0	3.2	IF A<4>	1	03
310	311	2.0	3.2	IF A<6>	1	04
312	313	2.0	3.2	IF A<8>	1	05
314	315	2.0	3.2	IF A<10>	1	06
316	317	2.0	3.2	IF A<12>	1	07
320	321	2.0	3.2	IF A<14>	1	10
322	323	2.0	3.2	IF B<0>	1	11
324	325	2.0	3.2	IF B<15>	1	12
326	327	2.0	3.2	IF EXT7	1	13
330	331	2.0	3.2	IF EXT8	1	14
332	333	2.0	3.2	IF EXT9	1	15
334	335	2.0	3.2	IF EXT10	1	16
336	337	2.0	3.2	IF EXT11	1	17

CONDITIONAL JUMP INSTRUCTIONS (INPUT MULTIPLEXER) (Cont)

Machine Code (Octal)*		Time (μsec)		Instruction	Multiplexer No.**	Multiplexer Channel
MEM0	MEM1	False	True			
340	341	2.0	3.2	IF EXT12	1	20
342	343	2.0	3.2	IF EXT13	1	21
344	345	2.0	3.2	IF EXT14	1	22
346	347	2.0	3.2	IF EXT15	1	23
350	351	2.0	3.2	IF EXT16	1	24
352	353	2.0	3.2	IF EXT17	1	25
354	355	2.0	3.2	IF EXT18	1	26
356	357	2.0	3.2	IF EXT19	1	27
360	361	2.0	3.2	IF EXT20	1	30
362	363	2.0	3.2	IF EXT21	1	31
364	365	2.0	3.2	IF EXT22	1	32
366	367	2.0	3.2	IF L	1	33
370	371	2.0	3.2	IF PWOK	1	34
372	373	2.0	3.2	Not used		

* Codes are given for jumps into or within Memory 1 and into or within Memory 2. The least significant bit of the machine code is the M (memory) bit.

**Multiplexer 1 (PCS16-D) is optional and resides in slot AB19 of the logic assembly. The MUX0 and MUX1 commands are used to select one or the other multiplexer.

SUBROUTINE INSTRUCTIONS

Machine Code (Octal)	Time (μsec)	Instruction
374	3.2	CALL – Jump to a Subroutine in Memory 0
375	3.2	CALL – Jump to a Subroutine in Memory 1
376	3.2	EXIT – Return from a Subroutine

APPENDIX F
DEFINITION TAPE LISTING

	INIT	
A = 0	DI	000
A = B	DI	001
A = A+1	DI	002
A = A-1	DI	003
A = A+B	DI	004
A = A-B	DI	005
A = AXORB	DI	006
A = AORB	DI	007
A = AB	DI	010
A = NOTA	DI	011
A = A/2	DI	012
A = AX2	DI	013
A = A+1(S)	DI	014
A = A-1(S)	DI	015
A = A+B(S)	DI	016
A = A-B(S)	DI	017
A = A/2(S)	DI	020
A = AX2(S)	DI	021
B = 0	DI	022
B = A	DI	023
B = A+B	DI	024
B = A-B	DI	025
B = AXORB	DI	026
B = AORB	DI	027
B = AB	DI	030
B = NOTB	DI	031
B = B/2	DI	032
B = A+B(S)	DI	033
B = A-B(S)	DI	034
B = B/2(S)	DI	035
BS = A	DI	036
BS = B	DI	037
GPI1 = A	DI	040
A = GPI1	DI	041
GPI2 = A	DI	042
A = GPI2	DI	043
GPI3 = A	DI	044
A = GPI3	DI	045
B = C1	DI	046

B = C2	DI	047
SI1 = A	DI	050
A = SI1	DI	051
TAPE1	DI	052
SI2 = A	DI	053
A = SI2	DI	054
TAPE2	DI	055
FF1 = 0	DI	056
FF1 = 1	DI	057
FF2 = 0	DI	060
FF2 = 1	DI	061
FF3 = 0	DI	062
FF3 = 1	DI	063
FF4 = 0	DI	064
FF4 = 1	DI	065
FF5 = 0	DI	066
FF5 = 1	DI	067
FF6 = 0	DI	070
FF6 = 1	DI	071
PAGE0	DI	072
PAGE1	DI	073
L = 0	DI	074
L = 1	DI	075
MUX0	DI	076
MUX1	DI	077
B = K1	DI	100
B = K2	DI	101
B = K3	DI	102
B = K4	DI	103
B = K5	DI	104
B = K6	DI	105
B = K7	DI	106
B = K8	DI	107
B = K9	DI	110
B = K10	DI	111
B = K11	DI	112
B = K12	DI	113
B = K13	DI	114
B = K14	DI	115
B = K15	DI	116
B = K16	DI	117
B = K17	DI	120
B = K18	DI	121
B = K19	DI	122
B = K20	DI	123
B = K21	DI	124
B = K22	DI	125
B = K23	DI	126
B = K24	DI	127
TR = A	DI	130
A = TRU	DI	131
A = TRL	DI	132
A = TR	DI	133

RMAR = A	DI	134
A = ROM	DI	135
B = C3	DI	136
B = C4	DI	137
A = SP1	DI	140
A = SP2	DI	141
A = SP3	DI	142
A = SP4	DI	143
A = SP5	DI	144
A = SP6	DI	145
A = SP7	DI	146
A = SP8	DI	147
A = SP9	DI	150
A = SP10	DI	151
A = SP11	DI	152
A = SP12	DI	153
A = SP13	DI	154
A = SP14	DI	155
A = SP15	DI	156
A = SP16	DI	157
SP1 = A	DI	160
SP2 = A	DI	161
SP3 = A	DI	162
SP4 = A	DI	163
SP5 = A	DI	164
SP6 = A	DI	165
SP7 = A	DI	166
SP8 = A	DI	167
SP9 = A	DI	170
SP10 = A	DI	171
SP11 = A	DI	172
SP12 = A	DI	173
SP13 = A	DI	174
SP14 = A	DI	175
SP15 = A	DI	176
SP16 = A	DI	177
A = SP17	DI	200
A = SP18	DI	201
A = SP19	DI	202
A = SP20	DI	203
A = SP21	DI	204
A = SP22	DI	205
A = SP23	DI	206
A = SP24	DI	207
A = SP25	DI	210
A = SP26	DI	211
A = SP27	DI	212
A = SP28	DI	213
A = SP29	DI	214
A = SP30	DI	215
A = SP31	DI	216
A = SP32	DI	217
SP17 = A	DI	220

SP18 = A	DI	221
SP19 = A	DI	222
SP20 = A	DI	223
SP21 = A	DI	224
SP22 = A	DI	225
SP23 = A	DI	226
SP24 = A	DI	227
SP25 = A	DI	230
SP26 = A	DI	231
SP27 = A	DI	232
SP28 = A	DI	233
SP29 = A	DI	234
SP30 = A	DI	235
SP31 = A	DI	236
SP32 = A	DI	237
MAR1 = A	DI	240
MEM1 = A	DI	241
A = MEM1	DI	242
MAR2 = A	DI	243
MEM2 = A	DI	244
A = MEM2	DI	245
RMAR = B	DI	246
B = ROM	DI	247
TR = 0	DI	250
B = SI1	DI	251
SI1 = B	DI	252
B = SI2	DI	253
SI2 = B	DI	254
GPI1 = B	DI	255
B = GPI1	DI	256
B = A+1	DI	257
B = A+1(S)	DI	260
B = A-1	DI	261
B = A-1(S)	DI	262
B = AX2	DI	263
B = AX2(S)	DI	264
B = A/2	DI	265
B = A/2(S)	DI	266
L = OVF	DI	267
L = LNOT	DI	270
HALT	DI	271
A = B/2	DI	272
A = B/2(S)	DI	273
DATI	DI	274
DATO	DI	275
TRU = A	DI	276
TRL = A	DI	277
EXT1	DC	01
EXT2	DC	02
EXT3	DC	03
EXT4	DC	04
EXT5	DC	05
EXT6	DC	06
DZ	DC	07

```
DP          DC  10
DN          DC  11
OVF         DC  12
A<1>        DC  13
A<3>        DC  14
A<5>        DC  15
A<7>        DC  16
A<9>        DC  17
A<11>       DC  20
A<13>       DC  21
A<15>       DC  22
FF1         DC  23
FF2         DC  24
FF3         DC  25
FF4         DC  26
FF5         DC  27
FF6         DC  30
KF1         DC  31
PF1         DC  32
KF2         DC  33
PF2         DC  34
CLK         DC  35
A<0>        DC  01
A<2>        DC  02
A<4>        DC  03
A<6>        DC  04
A<8>        DC  05
A<10>       DC  06
A<12>       DC  07
A<14>       DC  10
B<0>        DC  11
B<15>       DC  12
EXT7        DC  13
EXT8        DC  14
EXT9        DC  15
EXT10       DC  16
EXT11       DC  17
EXT12       DC  20
EXT13       DC  21
EXT14       DC  22
EXT15       DC  23
EXT16       DC  24
EXT17       DC  25
EXT18       DC  26
EXT19       DC  27
EXT20       DC  30
EXT21       DC  31
EXT22       DC  32
L           DC  33
PWOK        DC  34
EXT23       DC  35
            FIX
/
PAUSE
```

APPENDIX G
SIGNAL LISTINGS

Alphabetic signal listings with associated pin assignments of PDP16-M I/O and bus signals and PDP-11 I/O signals are contained in this Appendix.

Table G-1
PDP16-M I/O Slot Pin Assignments
(By Signal Name)

Signal	Slot	Pin
AUTO RUN	D01	S1/U1
CONTINUE	C19	M2
EXT1	C19	A1
EXT2	C19	B1
EXT3	C19	C1
EXT4	C19	D1
EXT5	C19	E1
EXT6	C19	F1
EXT7	C19	H1
EXT8	C19	J1
EXT9	C19	K1
EXT10	C19	L1
EXT11	C19	M1
EXT12	C19	N1
EXT13	C19	P1
EXT14	C19	R1
EXT15	C19	S1
EXT16	C19	U1
EXT17	C19	V1
EXT18	C19	B2
EXT19	C19	D2
EXT20	C19	L2
EXT21	C19	A2
EXT22	C19	C2
FF1	C19	N2
FF1	D19	V1
FF2	C19	P2
FF2	C20	V1
FF3	C19	R2

Table G-1 (Cont)
PDP16-M I/O Slot Pin Assignments
(By Signal Name)

Signal	Slot	Pin
FF3	D20	V1
FF4	C19	S2
FF5	C19	T2
FF6	C19	U2
GPI1 D00	D19	B2
GPI1 D01	D19	D2
GPI1 D02	D19	E2
GPI1 D03	D19	F2
GPI1 D04	D19	H2
GPI1 D05	D19	J2
GPI1 D06	D19	K2
GPI1 D07	D19	L2
GPI1 D08	D19	M2
GPI1 D09	D19	N2
GPI1 D10	D19	P2
GPI1 D11	D19	R2
GPI1 D12	D19	S2
GPI1 D13	D19	T2
GPI1 D14	D19	U2
GPI1 D15	D19	V2
GPI1 FF1	D19	V1
GPI1 I00	D19	A1
GPI1 I01	D19	B1
GPI1 I02	D19	C1
GPI1 I03	D19	D1
GPI1 I04	D19	E1
GPI1 I05	D19	F1
GPI1 I06	D19	H1
GPI1 I07	D19	J1
GPI1 I08	D19	K1
GPI1 I09	D19	L1
GPI1 I10	D19	M1
GPI1 I11	D19	N1
GPI1 I12	D19	P1
GPI1 I13	D19	R1
GPI1 I14	D19	S1
GPI1 I15	D19	U1
GPI2 D00	C20	B2
GPI2 D01	C20	D2
GPI2 D02	C20	E2
GPI2 D03	C20	F2
GPI2 D04	C20	H2
GPI2 D05	C20	J2
GPI2 D06	C20	K2
GPI2 D07	C20	L2
GPI2 D08	C20	M2

Table G-1 (Cont)
PDP16-M I/O Slot Pin Assignments
(By Signal Name)

Signal	Slot	Pin
GPI2 D09	C20	N2
GPI2 D10	C20	P2
GPI2 D11	C20	R2
GPI2 D12	C20	S2
GPI2 D13	C20	T2
GPI2 D14	C20	U2
GPI2 D15	C20	V2
GPI2 FF2	C20	V1
GPI2 I00	C20	A1
GPI2 I01	C20	B1
GPI2 I02	C20	C1
GPI2 I03	C20	D1
GPI2 I04	C20	E1
GPI2 I05	C20	F1
GPI2 I06	C20	H1
GPI2 I07	C20	J1
GPI2 I08	C20	K1
GPI2 I09	C20	L1
GPI2 I10	C20	M1
GPI2 I11	C20	N1
GPI2 I12	C20	P1
GPI2 I13	C20	R1
GPI2 I14	C20	S1
GPI2 I15	C20	U1
GPI3 D00	D20	B2
GPI3 D01	D20	D2
GPI3 D02	D20	E2
GPI3 D03	D20	F2
GPI3 D04	D20	H2
GPI3 D05	D20	J2
GPI3 D06	D20	K2
GPI3 D07	D20	L2
GPI3 D08	D20	M2
GPI3 D09	D20	N2
GPI3 D10	D20	P2
GPI3 D11	D20	R2
GPI3 D12	D20	S2
GPI3 D13	D20	T2
GPI3 D14	D20	U2
GPI3 D15	D20	V2
GPI3 FF3	D20	V1
GPI3 I00	D20	A1
GPI3 I01	D20	B1
GPI3 I02	D20	C1
GPI3 I03	D20	D1
GPI3 I04	D20	E1

Table G-1 (Cont)
PDP16-M I/O Slot Pin Assignments
(By Signal Name)

Signal	Slot	Pin
GPI3 I05	D20	F1
GPI3 I06	D20	H1
GPI3 I07	D20	J1
GPI3 I08	D20	K1
GPI3 I09	D20	L1
GPI3 I10	D20	M1
GPI3 I11	D20	N1
GPI3 I12	D20	P1
GPI3 I13	D20	R1
GPI3 I14	D20	S1
GPI3 I15	D20	U1
GROUND	C19	T1
MSYN	C19	K2
SI1 SI	C19	F2
SI1 SO	C19	E2
SI2 SI	C19	J2
SI2 SO	C19	H2
SSYN	C19	V2

Table G-2
PDP16-M RTM Data Bus Pin Assignments
(By Signal Name)

Signal	Pin (Slots AB01 – AB17)
DATA BIT 00	AA1
DATA BIT 01	AB1
DATA BIT 02	AC1
DATA BIT 03	AD1
DATA BIT 04	AE1
DATA BIT 05	AF1
DATA BIT 06	AH1
DATA BIT 07	AJ1
DATA BIT 08	AK1
DATA BIT 09	AL1
DATA BIT 10	AM1
DATA BIT 11	AN1
DATA BIT 12	AP1
DATA BIT 13	AR1
DATA BIT 14	AS1
DATA BIT 15	AU1
DATA ACCEPT	BA1
DATA READY	BB1
DONE	BC1
OVERFLOW	BD1
POWER CLEAR	BE1

Table G-3
PDP-11 UNIBUS Pin Assignments
(By Signal Name)

Signal	Pin	Signal	Pin
A00 L	BH2	D06 L	AF1
A01 L	BH1	D07 L	AH2
A02 L	BJ2	D08 L	AH1
A03 L	BJ1	D09 L	AJ2
A04 L	BK2	D10 L	AJ1
A05 L	BK1	D11 L	AK2
A06 L	BL2	D12 L	AK1
A07 L	BL1	D13 L	AL2
A08 L	BM2	D14 L	AL1
A09 L	BM1	D15 L	AM2
A10 L	BN2	GROUND	AB2
A11 L	BN1	GROUND	AC2
A12 L	BP2	GROUND	AN1
A13 L	BP1	GROUND	AP1
A14 L	BR2	GROUND	AR1
A15 L	BR1	GROUND	AS1
A16 L	BS2	GROUND	AT1
A17 L	BS1	GROUND	AV2
ACLO L	BF1	GROUND	BB2
BBSY L	AP2	GROUND	BC2
BG4 H	BE2	GROUND	BD1
BG5 H	BB1	GROUND	BE1
BG6 H	BA1	GROUND	BT1
BG7 H	AV1	GROUND	BV2
BR4 L	BD2	INIT L	AA1
BR5 L	BC1	INTR L	AB1
BR6 L	AU2	MSYN L	BV1
BR7 L	AT2	NPG H	AU1
C0 L	BU2	NPR L	AS2
C1 L	BT2	PA L	AM1
D00 L	AC1	PB L	AN2
D01 L	AD2	+5V*	AA2
D02 L	AD1	+5V*	BA2
D03 L	AE2	SACK L	AR2
D04 L	AE1	DCLO L	BF2
D05 L	AF2	SSYN L	BU1

*+5V is wired to these pins to supply power to the bus terminator only.

+5V should never be connected via the UNIBUS between system units.

APPENDIX H
USER OPTIONS

What can a potential user do and expect in terms of PDP16-M support? This depends largely on how many machines a user wishes to implement. Since program assembly, program and hardware interface debugging, as well as PROM loading, must be done with the aid of a PDP-8/E, the additional cost for the utility PDP-8/E computer must be considered. The small user would not necessarily be able to justify this additional cost. Therefore, Digital Equipment Corporation offers a service to load the PROMs for the small user at a nominal cost. However, the larger user, whether an OEM or an end user, would benefit considerably by purchasing a PDP-8/E and the MR16-SL Utility Interface Option. Although many PDP-8/E configurations, ranging in price from $5,000 to $50,000 are available, the PDP16-M program development and utility requirements are satisfied completely with the least expensive configuration. Having a PDP-8/E with the MR16-SL option equips the user to do his own program development, debugging, and loading, thereby saving the user time and money, and at the same time exposes the user to state-of-the-art process control and computer techniques.

READER'S COMMENTS

PDP-16M USERS GUIDE
DEC-16-IMUGA-A-D

Your comments and suggestions will help us in our continuous effort to improve the quality and usefulness of our publications.

What is your general reaction to this manual? In your judgment is it complete, accurate, well organized, well written, etc.? Is it easy to use? _____

_____

_____

_____

What features are most useful? _____

_____

_____

_____

What faults do you find with the manual? _____

_____

_____

_____

Does this manual satisfy the need you think it was intended to satisfy? _____

Does it satisfy *your* needs? _____ Why? _____

_____

_____

_____

Would you please indicate any factual errors you have found. _____

_____

_____

_____

Please describe your position. _____

Name _____ Organization _____

Street _____ Department _____

City _____ State _____ Zip or Country _____

Fold Here

Do Not Tear - Fold Here and Staple

FIRST CLASS
PERMIT NO. 33
MAYNARD, MASS.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

Postage will be paid by:

Digital Equipment Corporation
Technical Documentation Department
146 Main Street
Maynard, Massachusetts 01754