Digital Equipment Corporation
Maynard, Massachusetts

digital

R. C. Gunson

PRELIMINARY!

PDP-12

USER HANDBOOK

R. L. Emerson

DEC-12-GRZA-D

PDP-12
USER HANDBOOK

(PRELIMINARY)

DIGITAL EQUIPMENT CORPORATION ▫ MAYNARD, MASSACHUSETTS

1st printing April 1969
2nd printing June 1969
3rd printing August 1969
4th printing November 1969

TABLE OF CONTENTS

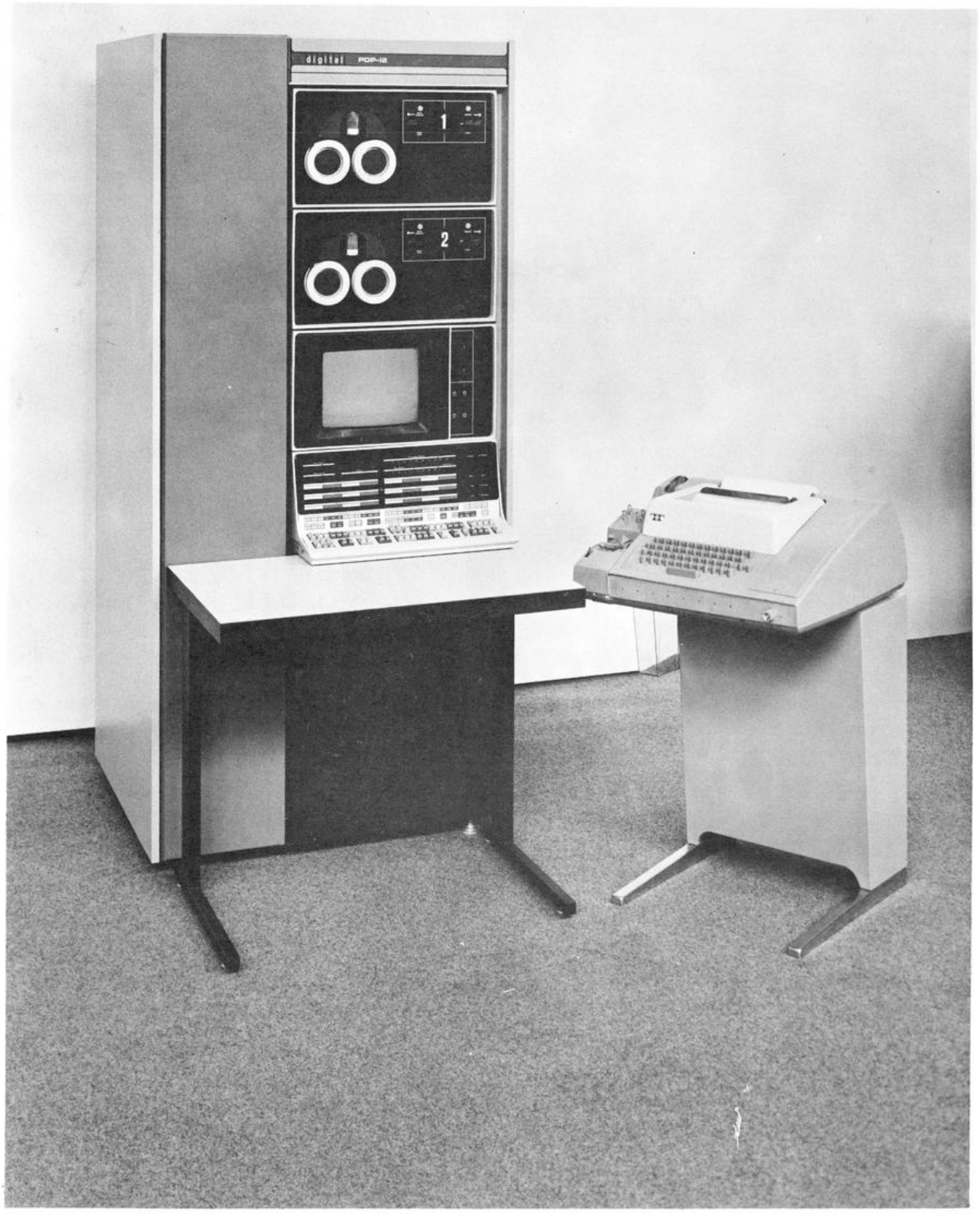
Chapter Page

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS (cont)

ABSTRACT

The coverage of the PDP-12 Programmed Data Processor contained in this User's Handbook is preliminary. A final version of the handbook will be available in the near future and can be obtained from your DEC sales office (see Rear Cover) or upon written request from the Digital Equipment Corporation, Maynard, Massachusetts 01754.



PDP-12 Programmed Data Processor

CHAPTER 1
GENERAL DESCRIPTION

1.1 DESCRIPTION

1.1.1 System

The PDP-12 Programmed Data Processor is a versatile digital computer which includes within its single central processor two distinct operating modes, each with its own complete instruction set. This versatility of the PDP-12 makes it, on the one hand, a laboratory-oriented machine with several built-in facilities for input/output, auxiliary storage, and control and sensing of external equipment, and, on the other hand, a general-purpose computer with a flexible input/output capability to which numerous peripheral devices may be easily attached. The central processor logic is fully parallel using a basic word length of 12 bits. The instruction cycle time is 1.6 microseconds; most instructions require from 1 to 3 cycles for execution.

Like its predecessor, the LINC-8, the PDP-12 operates in one mode as a LINC (*L*aboratory *IN*strument *C*omputer), and in the other mode as a PDP-8 computer – specifically, a PDP-8/I. Unlike the LINC-8, however, the PDP-12 has one central processor and both operating modes have equal status. (In the LINC-8, the LINC mode was subordinate to the PDP-8 mode). The computer may be stopped and started in either mode, and programs may switch from one to the other at will. Computations in one mode are immediately available to programs operating in the other mode because only one set of processing registers is involved.

The PDP-12 is offered in three configurations: A, B, and C in order of decreasing capacity. The two smaller systems, B and C, are expandable into the A configuration. The system discussed in this handbook is the PDP-12A.

The basic memory capacity of the PDP-12 is 4096 (4K) 12-bit words and can be expanded to 32,768 (32K) words of core storage.

As shown in Figure 1-1, the input/output facilities are available to the two operating modes of the PDP-12 in the following manner through LINC mode programming:

LINCtape – two TU55 tape transports controlled by a buffered subprocessor.

CRT Display – 6" x 9" screen, two intensification channels.

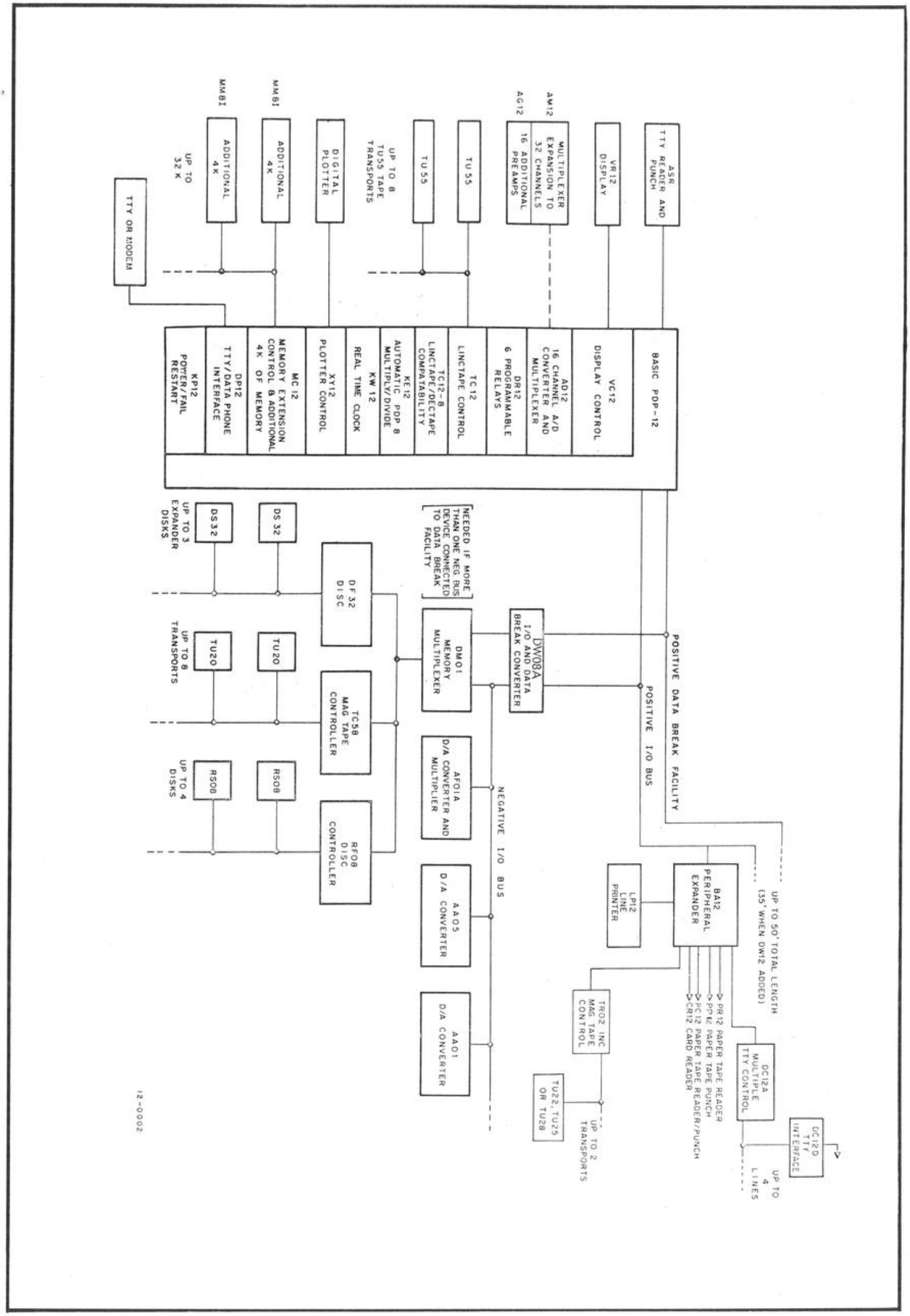


Figure 1-1. PDP-12 Programmed Data Processor System, Functional Block Diagram

Analog Inputs – eight external inputs, eight variable potentiometers

Relay Buffer – six relays for control of external equipment.

In addition to these, the PDP-12 is also equipped with a positive logic PDP-8/I type input/output (I/O) bus, to which can be attached, all 8 family peripherals and options such as a high-speed paper tape reader and punch, as well as the standard ASR-33 Teletype.

1.1.2 Central Processor

The central processor contains all the logic and registers required to carry out the functions of both operating modes of the PDP-12. The central processor can best be described in terms of its active registers:

Accumulator (AC) 12 Bits – This register contains data being operated upon. Its contents may be shifted or rotated right or left; incremented, cleared, or complemented; stored in memory or added to the contents of a memory register; and logically or arithmetically compared with the contents of any memory register. The AC holds the sum after an addition, and part of the product after a multiplication. The AC is also involved in the transfer of data to and from various other registers outside the central processor.

Link (L) 1 Bit – The Link is an extension of the AC. When a carry occurs out of bit 0 of the AC during a 2's complement addition, the Link is complemented. It may be set or cleared independently of the AC, and may be included (or not) in shifting and rotating operations performed on the contents of the AC.

Multiplier Quotient (MQ) 12 Bits - This register is used as a second arithmetic register for multiply and some rotate instructions. It is also used for the extended Arithmetic Option (KE12) functions.

Program Counter (PC) 12 Bits – This register contains the address of the next instruction to be executed within the memory field selected by the Instruction Field Register (see below). In PDP-8 mode, the PC acts as a 12-bit counter; in LINC mode, it acts as a 10-bit counter.

Memory Address Register (MA) 12 Bits – This register contains the address for memory references. Whenever a core memory location is being accessed, either for reading or for writing, the MA contains the address of that location.

Instruction Register (IR) 12 Bits – This register contains the complete binary code of the instruction being executed.

Memory Buffer (MB) 12 Bits – All information passing between memory and any other register in the PDP-12 must go through the Memory Buffer Register, whether the transfer involves the central processor, an external device, or another memory register.

Instruction Field Register (IF) 5 Bits – This register selects the memory field containing the executable program. In LINC mode, it is used to designate one of up to thirty-two 1024-word segments. In PDP-8 mode, the three high-order bits of the IF are used to designate one of up to eight 4096-word fields.

Data Field Register (DF) 5 Bits This register selects the memory field containing data to be indirectly accessed by the memory reference instructions of a program. The fields are specified in each mode in the same way that the IF specifies the Instruction Field.

1.1.3 Memory

The principal unit of core memory is a module of 4096 (4K) 12-bit words. Additional 4K banks may be added, to a total of eight, or 32,768 words. Within each bank, the logical organization of memory depends on the operating mode. In LINC mode, the bank is divided into four 1024-word segments. At any given time, only two of these segments are active: the Instruction Field, which contains the executable program and the directly accessed data; and the Data Field, which contains only indirectly accessed data. Absolute addresses may be assigned and changed at will using the IF and DF described above.

In the PDP-8 mode, the memory field, which is the size of a 4K module, is divided into 32 pages of 128 words each. Within a single page, data may be accessed directly; between pages, indirect addressing must be used. If more than 4K of memory is provided, the IF and DF registers specify the active fields.

1.1.4 Operating Modes

The two operating modes, LINC and PDP-8, are independent of each other, though they may be combined and intermixed within a program. The user can run programs from the already-existing libraries for the 8 family of computers including the LINC-8. Using the I-O Handler (PROGOFOP simulator) program provided with the PDP-12 basic software, most programs written for the LINC-8 can be run without modification. (Some LINC-8 programs may require slight changes). A complete software system designed for the PDP-12 allows the programmer to assemble coding for either or both modes in a single program.

LINC Mode

In this mode, the instruction set of the classic LINC computer is implemented. In addition, several new provisions are available:

Extended Tape Addressing – This allows the programmer to transfer information between LINCtape and any section of core, removing the restriction to specific quarters of a given memory field. Other features include:

1. Tape Interrupt – which connects the tape processor status to the Program Interrupt.

2. No-pause - which permits the central processor to resume operation after initiating a tape transfer without waiting for completion.

3. Hold-motion – which allows a unit to remain in motion after deselection.

I/O Bus Access - In LINC mode the user has immediate access to those devices activated by LINC instructions A-D, DISPLAY, RELAYS, SENSE LINES, and TAPE. Any other device connected to the I/O bus may be

directly accessed from LINC mode programming by means of a special two-word instruction, in which the second word enables the bus and initiates the PDP-8 IOT timing chain. This second word is interpreted as a standard PDP-8 IOT instruction. The program continues to operate in LINC mode.

Special Functions – The LINC programmer may, by setting certain flip-flops: 1) change the size of characters displayed on the CRT; 2) enable the program trap, which intercepts certain LINC instruction codes; 3) disable interrupts from the ASR-33; 4) speed the sampling of analog inputs; and 5) clear the PDP-12 status by generating an I/O preset pulse.

PDP-8 Mode

In this mode, the user has available the entire PDP-8/I instruction set.

Interaction Between Modes

The user may switch from one mode to the other at will. In LINC mode, execution of the instruction *PDP* causes the processor to change immediately to PDP-8 mode operation, and all subsequent instructions are interpreted as PDP-8/I instructions. To switch from PDP-8 mode to LINC mode the IOT instruction *LINC* is used.

1.1.5 **Input/Output Facilities and Display**

As can be seen from Figure 1-1, there are two main paths for the transmission of data from the central processor or memory to peripheral devices. One path, which is controlled by LINC mode programming, leads to the CRT display, LINCtape, A-D converter, and relays. The other path, which is the I/O bus, leads to the ASR-33 and to a large number of optional devices, such as: plotters, high-speed tape and card readers, disk storage, line printers, etc.

Display

The Cathode Ray Tube Display has a 58.5-square inch (6.5 x 9 inches) screen, on which individual points and whole characters may be displayed. The unit has two intensification channels, controlled by programming and by a switch on the display. Characters are plotted on a 4 point x 6 point matrix; a full character can be displayed with two instructions. Provision is made for displaying two sizes of characters.

Data Terminal

The data terminal provides a flexible means of receiving analog inputs and controlling the operation of external equipment not directly interfaced to the PDP-12.

Analog Inputs – Sixteen analog inputs feed a 10-bit A-D converter. A single LINC mode installation samples any of the 16 channels. A second set of sixteen inputs with preamplifiers, can be added to the basic facility.

Eight of the inputs, taken from phone jacks mounted on the Data Terminal Panel, are fed through preamplifiers to the converter. The remaining eight are taken from continuously-variable, ten-turn potentiometers, which are also mounted on the panel.

Relay Buffer – Six relays, mounted on the Data Terminal Panel, may be switched by means of a LINC mode instruction. The relays may be used to start and stop operations in external equipment. The status of the relays can be read back into the AC.

Auxiliary Scope Connector – A Blue Ribbon connector mounted on the Data Terminal Panel is wired to accept an auxiliary CRT for displaying information also sent to the screen of the built-in scope.

Sense Lines

These 12 digital sense lines may be individually tested with a LINC mode instruction.

LINCtape

Two TU55 transports are controlled by a fully-buffered tape processor; once initiated by the LINC program, tape operations are carried out independently of the central processor. Tapes normally are written and read in standard LINCtape format, though non-standard format can be used. A special hardware option, TC12-8, permits the use of PDP-8 DECtape format. In addition to the basic LINCtape commands, the PDP-12 also includes an Extended Operations facility, which allows, among other features, the transmission of data between tape and any program-defined area of memory, and the addition of TU55 transports to a total of eight.

Input/Output (I/O) Bus

This connecting facility provides the control and data transmission path between the central processor and any peripheral devices that are attached to the bus. For some devices, such as: paper tape readers and punches, line printers, and incremental plotters, data is transferred via the accumulator (AC). Others, including magnetic tape and disk, use the three-cycle or single-cycle Data Break for direct memory access. The I/O bus uses positive logic and accepts peripherals used with 8 family of computers. The processor is prewired to accept the following I/O bus options:

Extended Arithmetic Element (EAE), Type KE12

Programmable Real-time clock, Type KW12

Incremental Plotter and control, Type XY12

TTY or Dataphone Interface, Type DP12

Many other devices can be added to the PDP-12 I/O bus with the inclusion of the BA12 Peripheral Expander and the DW08 I/O and Bus Converter. The Peripheral Expander allows the addition of high-speed paper tape reader and punch, card reader, and optional communication interfaces. The Bus Converter provides for the addition of disk and IBM-compatible magnetic tape storage, and A/D and D/A converters and associated multiplexers designed for the negative-logic PDP-8 I/O Bus.

Keyboard/Printer (ASR-33)

An important means of direct communication between the user and the operating program is the ASR-33 Keyboard/Printer, standard on all configurations of the PDP-12. The ASR-33 is connected to the I/O bus, and can be accessed for input or output by programs in either operating mode. The ASR-33 is equipped with paper tape reader and punch; the reader and keyboard use the same input path and instructions, while the printer and punch

use the same output path and instructions. The maximum transfer rate in either direction is 10 characters per second.

The ASR-33 has both full- and half-duplex capability. In full-duplex operation, data may be transmitted in both directions simultaneously, without interference. In half-duplex operation, data may be transmitted in only one direction at a time.

1.2 SYMBOLS AND ABBREVIATIONS

The following symbols and abbreviations are used throughout this handbook:

AC, MB, PC, MQ, MA – L, IF, DF, IR	Central Processor registers: Accumulator, Memory Buffer, Program Counter, Multiplier-Quotient, Memory Address, Link, Instruction Field, Data Field, Instruction Register, respectively.
R –	General representation of any register.
C(R) –	The contents of register R.
$C(R_j)$ –	The contents of bit j of register R.
$C(R_{j-k})$ –	The contents of bits j through k, inclusive, of register R.
$C(R_l)$ –	The contents of the left half of register R.
$C(R_r)$ –	The contents of the right half of register R.
$\overline{C(R)}$ –	The complement of the contents of register R.
Y –	The effective address of an operand.
I –	The Indirect address bit of an instruction. In LINC mode, I represents bit 7; in PDP-8 mode, bit 3.
C(R)→C(S) –	The contents of register R replace those of register S.
N→C(R) –	The quantity N replaces the contents of register R.

CHAPTER 2
CONTROLS AND INDICATORS

This chapter describes the function of the controls and indicators of the PDP-12 computer console, data termination panel, Type VR12 Oscilloscope, and the Type TU55 Tape Transport, and ASR35 Teletype.

2.1 PDP-12 CONSOLE CONTROLS AND INDICATORS

Tables 2-1 through 2-8 describe the controls and indicators located on the console of the PDP-12. Figure 2-1 provides a front panel view of the console.

2.2 DATA TERMINAL PANEL

The Data Terminal Panel refers to the area behind the door on the left side of the front of the PDP-12. Normally up to four separate panels are placed here. In addition, a storage rack to hold LINC tapes can be placed in any of the unused spaces of the Data Terminal Panel area. The four standard panels are:

1. Power Switch Panel
2. Relay and Analog Input Panel
3. Analog Extension Panel
4. Clock Input Panels

The first two are described in Tables 2-9 and 2-10 and illustrated in Figure 2-2 and 2-3; the last two are described with the associated options (AG12 and KW12).

2.3 TYPE VR-12 OSCILLOSCOPE

Table 2-11 lists the controls and indicators of the Type VR12 Oscilloscope. Figure 2-4 shows a front panel view of the oscilloscope.

2.4 TYPE TU55 TAPE TRANSPORT

Table 2-12 lists the functions of controls and indicators of the Tape TU55 Tape Transport. Figure 2-5 provides a front panel view of the tape transport.

2.5 MODEL ASR33 TELETYPE CONTROLS

Table 2-13 lists the functions of controls of the Model ASR 33 Teletype. Figure 2-6 provides a front view of the teletype.

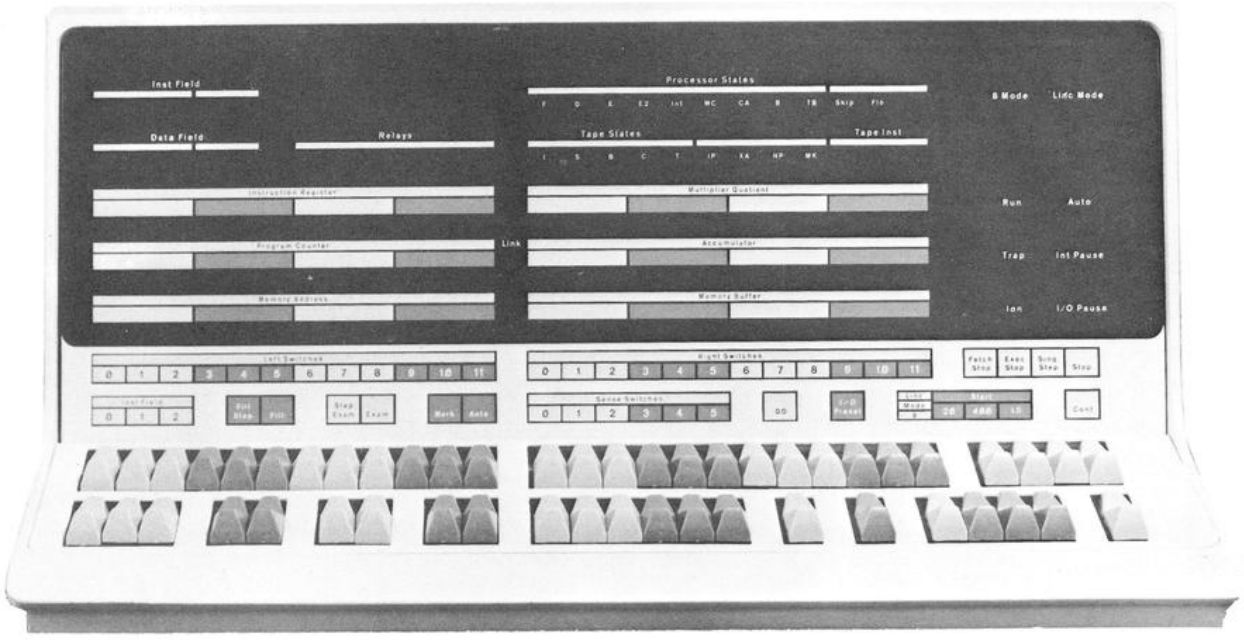


Figure 2-1. PDP-12 Operator Console

TABLE 2-1. CENTRAL PROCESSOR REGISTER INDICATORS

Indicator	Bits
Instruction Field	5
Data Field	5
Relays	6
Instruction Register	12
Program Counter	12
Memory Address	12
Multiplier Quotient	12
Accumulator	12
Link	1
Memory Buffer	12

TABLE 2-2. CENTRAL PROCESSOR MAJOR STATE INDICATORS

Indicator	State
F	Instruction Fetch
D	Deferred Address
E	Instruction Execution
E2	Instruction Execution 2
Int.	Program Interrupt
WC	Word Count
CA	Current Address
B	Break
TB	Tape Break

TABLE 2-3. CENTRAL PROCESSOR MISCELLANEOUS INDICATORS

Indicator	Interpretation When Lit
Skip	Skip Flip-Flop is set
Flo	Overflow Flip-Flop is set
8 Mode	Processor is in PDP-8 Mode
Linc Mode	Processor is in Linc Mode
Run	Processor is running
Auto	Auto Restart Flip-Flop is set
Trap	Instruction trap is enabled
Int Pause	An internal pause is occurring
Ion	Program Interrupt facility enabled
I/O Pause	An I/O Pause is occurring

TABLE 2-4. TAPE PROCESSOR MAJOR STATE INDICATORS

Indicator	State	When lit, indicates that:
I	Idle	The tape processor is in the Idle state.
S	Search	The tape processor is in the Search state.
B	Block	The tape processor is in the Block state.
C	Check Word	The tape processor is in the Check Word state.
T	Turn Around	The tape processor is in the Turn Around state.
IP	In Progress	A tape operation is In Progress.

TABLE 2-5. TAPE PROCESSOR MISCELLANEOUS INDICATORS

Indicator	Interpretation	Function
XA	Extended Address mode	Indicates that the processor is in the Extended Address mode.
NP	No Pause mode	Indicates that the processor is in the No Pause mode.
MK	Mark flip-flop	Indicates that the Mark flip-flop is set.
Tape Inst	3-bit Tape Instruction register	These three lights indicate the contents of the 3-bit Tape Instruction register.

TABLE 2-6. FUNCTION OF COMPUTER CONSOLE KEYS

Key	Function
I/O Preset	This switch causes the processor to halt when it is in Internal Pause state during a tape instruction, and sets processor mode to the state of the console MODE switch. The INST Field register is set to 2 and the Data Field register is set to 3. I/O PRESET also generates the Processor I/O PRESET pulse and the I/O BUS INITIALIZE pulse thereby clearing all I/O device flags and operations.
DO	This switch causes the processor to perform one instruction. If the processor is in the LINC mode, the processor performs the instruction defined by the Left Switches (and the Right Switches if it is a double word instruction); if the processor is in the PDP-8 mode, the processor performs the instruction defined by the Left Switches.
Start 20	This switch causes the processor to start at location 20 of the currently selected Memory Bank when the processor is in the LINC mode, and at absolute location 0020 when in PDP-8 mode.
Start 400	This switch causes the processor to start at location 400 of the currently selected Memory Bank when the processor is in the LINC mode, and at absolute address 0400 when in PDP-8 mode.
Start LS	This switch causes the processor to start at the 15-bit address specified by the Left Switches and the Instruction Field Switches.
Cont	This switch causes the processor to resume operation.
Exam	This switch does the following: 1) Transfers the contents of the Left Switchesinto the Memory Address register. 2) Display in the Memory Buffer register, the contents of the absolute core address designated by the Left Switches.
Step Exam	This switch increments the contents of the Memory Address register and displays the contents of this new address in the Memory Buffer register. This incrementing extends for 10 bits in LINC mode and 12 bits in PDP-8 mode.
Fill	This switch does the following: 1) Transfers the contents of the Left Switches into the Memory Address register. 2) Deposits the contents of the Right Switches into the memory location whose absolute address is designated by the Left Switches and the Instruction Field Switches.
Fill Step	This switch does the following: 1) Deposits the contents of the Right Switches into the memory location whose address is in the Memory Address register.

TABLE 2-6. FUNCTION OF COMPUTER CONSOLE KEYS (cont.)

Key	Function
	2) Increments the contents of the Memory Address register. (This incrementing extends for 10 bits in LINC mode and 12 bits in PDP-8 mode). 3) Displays in the Memory Buffer register the contents of the location specified by the new contents of the Memory Address register.
Mode	This switch determines the mode (LINC or PDP-8) to which the processor will be set when the I/O PRESET switch is activated.
Auto	This switch is used to conditionally set the Auto Restart flip-flop. The Auto Restart flip-flop causes the processor to automatically restart at a variable time (determined by the console controls) after a processor stops for any of the following reasons: 1) Single Step Switch activated 2) F Stop address match 3) E Stop address match 4) The end of a Step Exam operation 5) The end of a Fill Step operation 6) The end of a DO switch operation The Auto Restart Flip-flop is cleared by any of the following conditions: 1) Stop Switch pressed while processor is running 2) DO, FILL STEP, or STEP EXAM Switch activated and the Auto Switch NOT pressed. 3) A processor HLT instruction is executed (either Mode) 4) I/O Preset pulse is generated.

TABLE 2-7. TOGGLE SWITCH REGISTERS

Register	Bits	Function
Left Switches	12	These switches form a 12-bit word which can be read into the accumulator with the LINC mode instruction LSW (517). This word also specifies the address to be examined when the Exam Switch is used, the address into which data will be placed when the Fill Switch is used, the stopping address for

TABLE 2-7. TOGGLE SWITCH REGISTERS

Register	Bits	Function
		E Stop and F Stop functions, and the instruction to be performed when the DO switch is used.
Right Switches	12	These switches form a 12-bit word which can be read into the accumulator with the LINC mode instruction RSW (516), or the PDP-8 mode instruction OSR (7404). This word also provides data to be stored in memory when the Fill or Fill Step Switches are used. When the DO Switch is used, the Right Switches contain the second word of two-word instructions.
Inst Field	3	These switches are a high-order extension of the Left Switches. They provide addressing information for systems equipped with 8K or more of memory, for the EXAM, FILL Start LS, E Stop and F Stop functions.
Sense Switches	6	These switches are individually interrogated by LINC mode skip instructions, thereby enabling user control of program branching.

TABLE 2-8. INDIVIDUAL CONSOLE TOGGLE SWITCHES

Switch	Function
Stop	This switch causes the processor to stop at the end of every instruction. For the purposes of the Stop switch, Traps, Interrupt, Tape Break, and single-cycle Data Break are considered to be single-cycle instructions. During a three-cycle Data Break, the processor is stopped after the break cycle.
Single Step	This switch causes the Run flip-flop to be cleared thereby disabling the timing circuits at the end of one cycle of operation. Thereafter, repeated operation of the Cont switch steps the program one cycle at a time so that the operator can observe the contents of registers in each major state.
Fetch Stop	This switch causes the processor to stop when the address designated by the Left Switches matches the current address in the Memory Address register during the Fetch cycle. For systems with more than 4K of memory, the Instruction Field Switches designate the three most significant bits of the address.
Exec Stop	This switch causes the processor to stop when the address designated by the Left Switches matches the current address in the Memory Address register during any computer cycle *except* a Fetch cycle. For systems having more than 4K of memory, the Inst Field Switches designate the three most significant bits of the address.

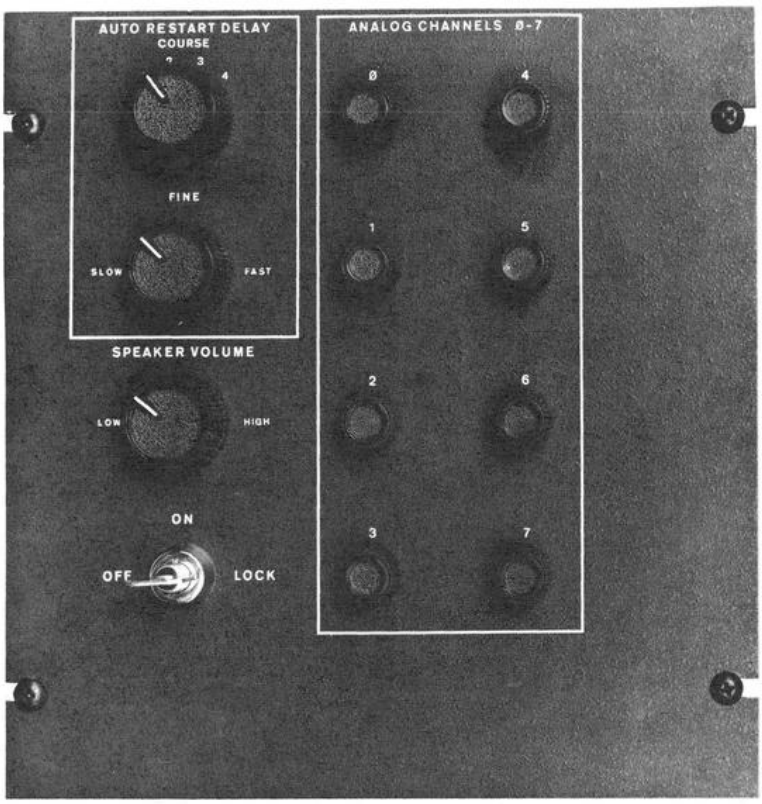


Figure 2-2. Power Switch Panel

TABLE 2-9. KNOB AND POWER SWITCH PANEL

Panel Controls	Function
PWR-Panel Lock	This 3-position, key-locking switch is used to turn the PDP-12 on as well as inhibit console intervention during an operating program. When fully counterclockwise, the PDP-12 power is off. When turned to the center position, the PDP-12 is turned on and the console activated. When the switch is fully clockwise, the PDP-12 is on but console control functions are totally inhibited while the PDP-12 RUN light is on. The LEFT SWITCHES, RIGHT SWITCHES, and SENSE SWITCHES remain operative.
Speaker Volume	The volume of the speaker, which is driven by bit AC 00, is controlled by this knob. (The speaker is added to the system when the KD12 is included in the system configuration).
Auto Restart	These two knobs control the delay period of Auto Restart after a processor stop due to E Stop, F Stop or Single Instruction operation. (See Auto switch description in Table 2-6.)

TABLE 2-9. KNOB AND POWER SWITCH PANEL (cont.)

Panel Controls	Function
Coarse and Fine	The Coarse delay selects overlapping ranges from 10 μsec to 10 sec. The Fine control gives variation within a range of 20.1 of the selected Coarse delay.
Analog Knobs	These 10-turn potentiometers are connected to analog input channels 0-7 of the AD12 Analog-to-Digital Converter. These knobs therefore provide eight continuously variable parameters within the range of ± 512_{10} for program usage.

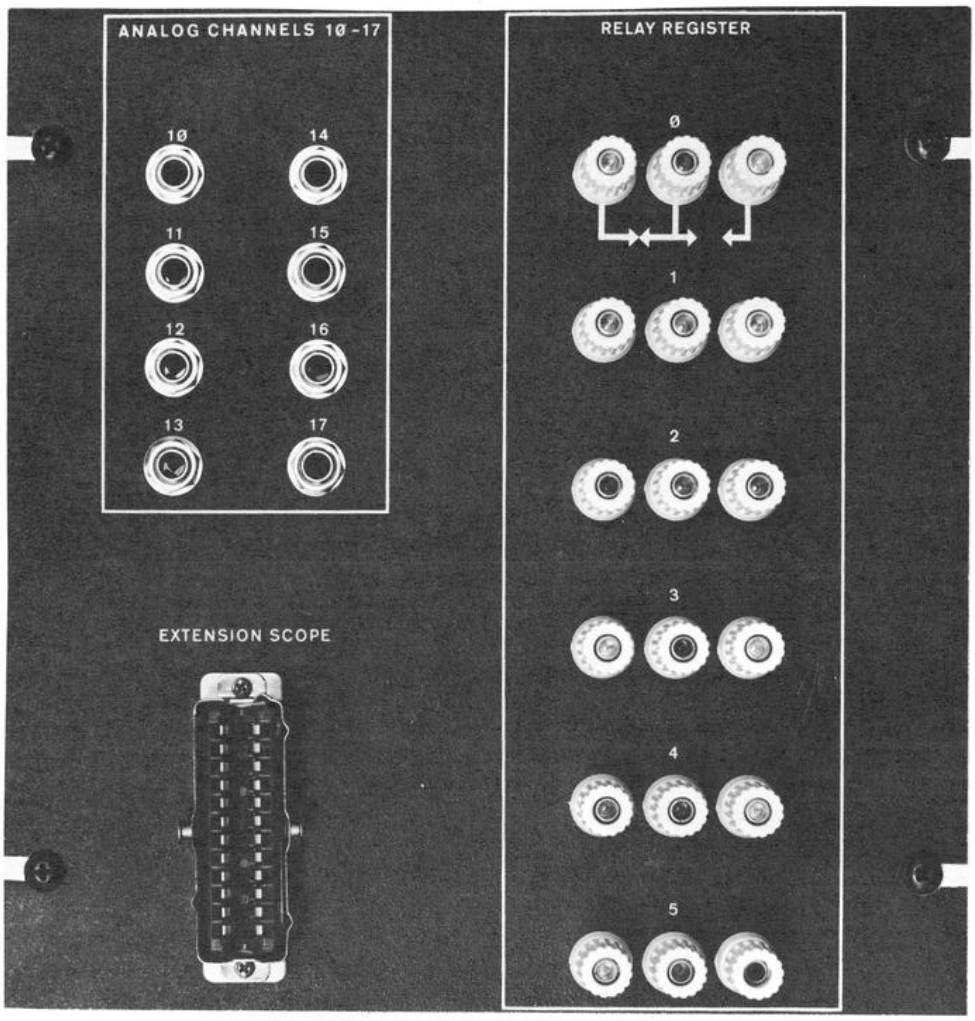


Figure 2-3. Relay and Analog Input Panel

TABLE 2-10. INPUTS AND OUTPUTS OF ANALOG INPUT-RELAY PANEL

Terminals	Function
Analog Inputs	± IV input connections for AD12 Analog-to Digital converter channels 10_8 - 17_8
Relay Contacts	One form C set of contacts for each of the six system relays is available at the binding posts.
Extension Scope	This 24-pin connector is used to connect an extension scope for remote operations, multiple displays, or photographing of display output. See the VC12 description for pin connections and drive characteristics.

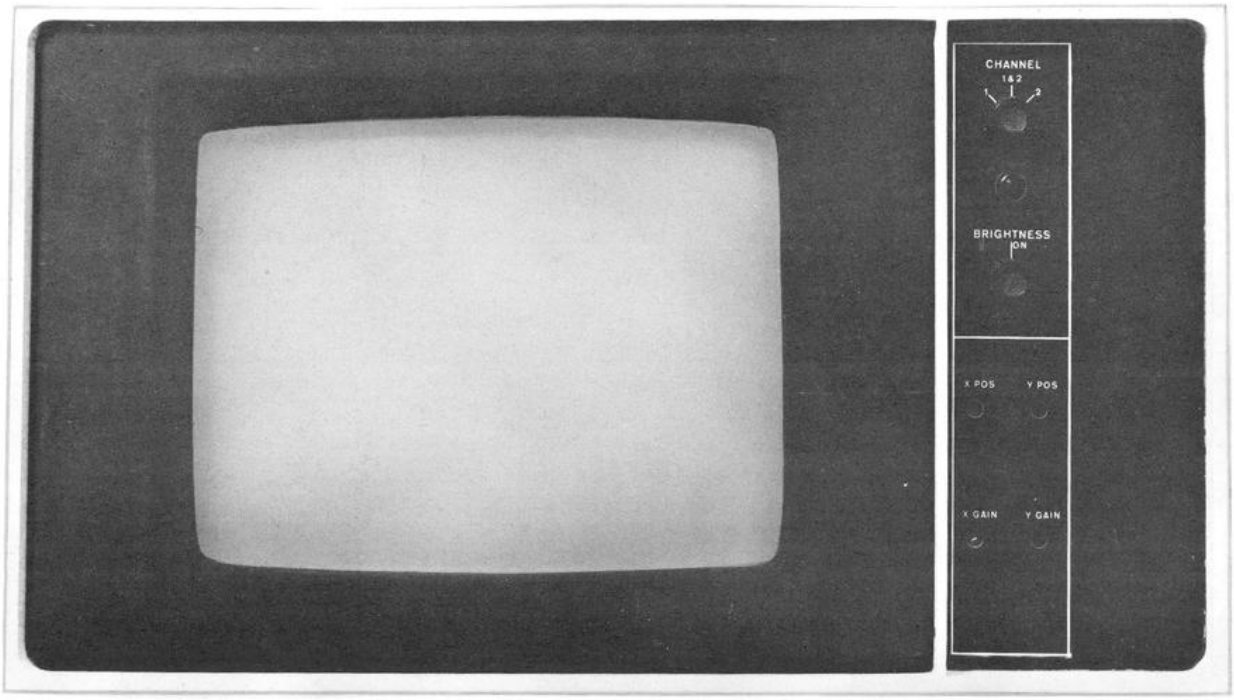


Figure 2-4. Type VR12 Oscilloscope

TABLE 2-11. VR12 DISPLAY SCOPE CONTROLS

Control	Function
CHANNEL SELECT	Select which of two intensity channels will cause scope display.
BRIGHTNESS	Control level of Brightness
X GAIN	Horizontal size of display
X POSITION	Horizontal position of display
Y GAIN	Vertical size of display
Y POSITION	Vertical position of display

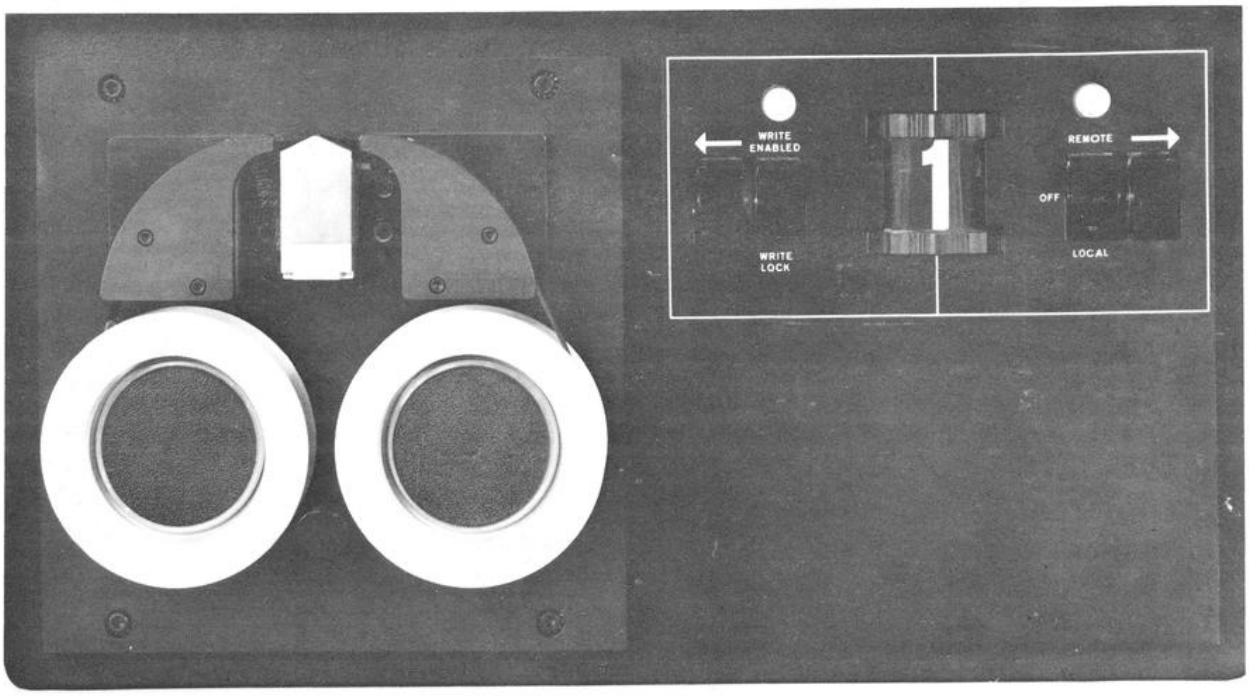


Figure 2-5. TU30 Tape Transport Control Panel

TABLE 2-12. TU55 TAPE TRANSPORT CONTROLS AND INDICATORS

Control or Indicator	Function
Forward tape-motion switch (designated in Figure 2-5 by arrow pointing to the left)	Provides forward tape motion (i.e., from right to left) only if REMOTE/OFF/LOCAL switch is set to LOCAL.
WRITE ENABLED/WRITE LOCK switch	
WRITE ENABLED	Permits TC12 control system to write information on the TU55.
WRITE LOCK	Prevents above writing. If TC12 control system is commanded to write on tape during the WRITE LOCK setting, the control signals a write lock error.
WRITE ENABLED indicator	Lights when WRITE ENABLE/WRITE lock switch is in the WRITE ENABLE position.
Address selector (or unit selector) 1 2 3 4 5 6 7 8	When set to one of the numerals (designating addresses) and REMOTE/OFF/LOCAL switch is on REMOTE or OFF, the transport is selected when the line indicated by the switch wiper corresponds to the computer selection through the TC12 control. Then the transport responds to command signals from external control and can assert a write enabling signal to the control. In addition, all head channels are connected through the head relay to data bus information lines.
OFF LINE	Prevents TC12 control system from selecting the TU55.
REMOTE/OFF/LOCAL switch	
REMOTE	Permits TU55 to accept command and control signals from the TC12 control system; also enables head relay logic to connect all head channels to data bus information lines as soon as appropriate transport is selected.
OFF	Inhibits operation of the G850 SCR Motor Control Modules and releases the brakes. Power for the logic components comes from power supplies associated with the external control, and therefore, the OFF position does not turn off the +10 and -15v power.

TABLE 2-12. TU55 TAPE TRANSPORT CONTROLS AND INDICATORS (cont.)

Control or Indicator	Function
	The OFF position is used when loading new tape reels since it releases the motor hubs. The transport should be set to OFF when not in use.
LOCAL	Permits forward and reverse tape-motion switch to provide tape motion in direction of arrow. Transport cannot be selected.
REMOTE indicator	Lights only when transport is selected by the control
Reverse tape-motion switch (designated in Figure 2-5 by arrow pointing to the right)	Provides for motion in the reverse direction (i.e., from left to right), but only when REMOTE/OFF/LOCAL switch is on LOCAL. If both reverse and forward tape-motion switches are pressed simultaneiously, the reverse motion takes place.

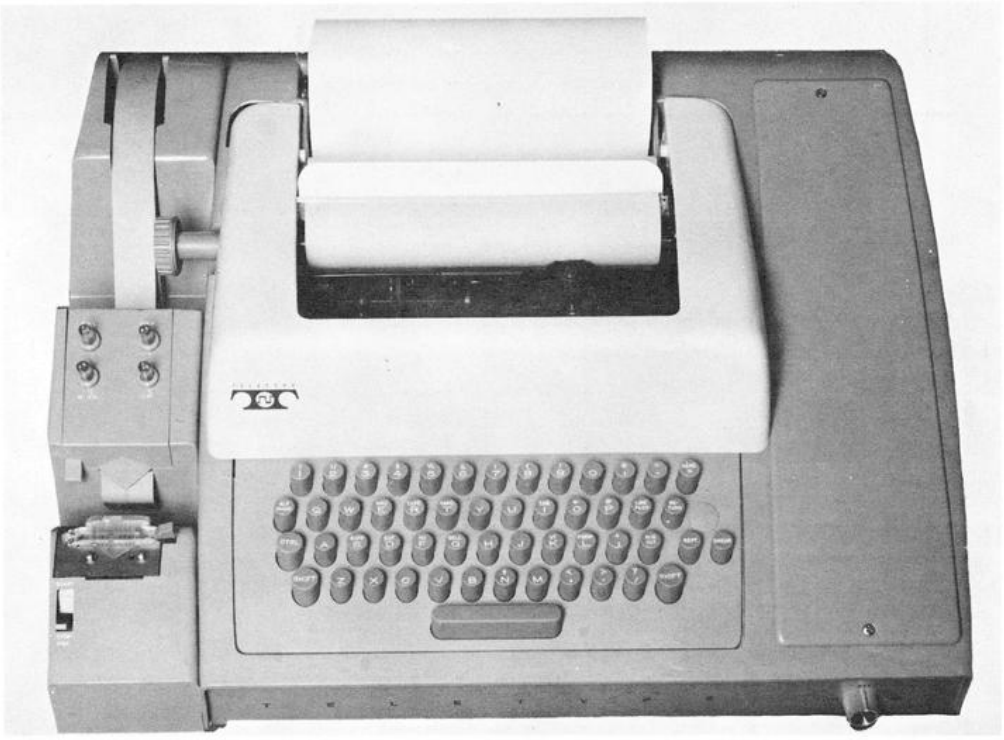


Figure 2-6. ASR-33 Teletype

TABLE 2-13. ASR-33 TELETYPE FUNCTION OF CONTROLS

Control	Function
REL. pushbutton	Disengages the tape in the punch to allow tape removal or tape loading.
B. SP. pushbutton	Backspaces the tape in the punch by one space, allowing manual correction or rub out of the character just punched.
OFF and ON pushbuttons	Control use of the tape punch with operation of the Teletype keyboard/printer.
START/STOP/FREE switch	Controls use of the tape reader with operation of the Teletype. In the FREE (lowest) position the reader is disengaged and can be loaded or unloaded. In the STOP (center) position the reader mechanism is engaged by de-energized. In the START (highest) position the reader is engaged and operated under program control.
Keyboard	Provides a means of printing on paper in use as a typewriter and punching tape when the punch ON pushbutton is pressed, and provides a means of supplying input data to the computer when the LINE/OFF/LOCAL switch is in the LINE position.
LINE/OFF/LOCAL switch	Controls applicaton of primary power in the Teletype and data connection to the processor. In the LINE position the Teletype is energized and connected as an I/O device of the computer. In the OFF position the Teletype is de-energized. In the LOCAL position the Teletype is energized for off-line operation, and signal connections to the processor are broken. Both LINE and LOCAL use of the Teletype require that the computer be energized through the POWER switch.

CHAPTER 3
LINC MODE PROGRAMMING

Section I. ORGANIZATION OF MEMORY

In LINC mode programming, each memory bank (Figure 3-1) is divided into four 1024-word segments. When a LINC mode program is in progress, two of these segments are active. The *Instruction Field* segments contains the executable program, and data which may be directly or indirectly addressed. The *Data Field* segment contains only indirectly addressable data. The fields are assigned to any two 1K segments by setting the 5-bit Instruction Field and Data Field Registers, each of which can select one of up to 32 segments. The two fields need not be adjacent, in any particular order (either field can be above or below the other), or even in the same memory bank.

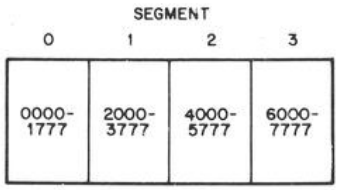


Figure 3-1. Memory Bank Segments and Addresses

With respect to the LINC program, addresses within a field remain constant, regardless of the field's actual location. Instruction Field Addresses run from 0000 through 1777; Data Field addresses run from 2000 through 3777. Thus, no matter where they are assigned, the two fields may be considered logically contiguous.

3.1 PROGRAM COUNTER

The Program Counter acts as a 10-bit counter in LINC mode, so that executable programs can be stored only in the Instruction Field. If the contents of PC_{2-11} are incremented beyond 1777, they return to 0000; the two high-order bits of the PC are unaffected. Thus, incrementing C(PC) = 3777 yields C(PC) = 2000. Likewise, 5777 increments to 4000, and 7777 to 6000. This 10-bit indexing is very common in LINC mode operations.

3.2 INSTRUCTION AND DATA FIELD REGISTERS

These two 5-bit registers select the 1K segments to be used by the LINC program. The three high-order bits of each register are identical with the three bits of the corresponding PDP-8 mode Memory Field register. The contents of the IF and DF may be set, changed, or examined at any time by the use of LINC instructions.

3.3 INSTRUCTION FIELD RESERVED LOCATIONS

This field contains the executable program. The following registers are set aside in this field for special uses:

Field Address	*Use*
0000	Holds return address after execution of JMP.
0001	Holds horizontal co-ordinate during execution of DSC.
0001-0017	As *β-registers*, used by all indirect-reference instructions to hold the effective address of an operand.
0000-0017	As *α-registers*, used by SET, XSK, and DIS.
0020	Program start location when START 20 key is pressed.
0400	Program start location when START 400 key is pressed.

When the instruction field occupies the lowest segment of memory (that is, when C(IF) = 00), the following addresses are also reserved:

0140	Holds return address after an instruction trap.
0141	Location to which control is transferred after an instruction trap.
0040	Holds return address after a program interrupt during LINC mode.
0041	Location to which control is transferred after a program interrupt during LINC mode.

3.4 DATA FIELD RESERVED LOCATIONS

There are no specially-reserved registers in this field. Its contents cannot be accessed directly; data can be stored or retrieved only by indirect addressing.

Section II. MEMORY ADDRESSING METHODS

Almost every program, at some time during its execution, will need an item of data stored in memory. Such an *operand* can be obtained only by specifying the address of the register in which it is stored (or to be stored, if the data is going the other way). An instruction which requires a reference to memory can designate the desired location in two ways. It may include the address of the operand as part of the instruction itself and *directly address* the location of the operand. Or, the instruction may specify the address, not of the operand, but of a register containing the address of the operand, thus *indirectly* addressing the data storage register.

The need for indirect addressing is readily apparent: with eleven bits required to specify a Data Field address, not much is left of a 12-bit word to use for instruction codes. It is necessary to reduce the number of address bits available within a memory reference instruction, and to use a limited set of directly addressable locations as *pointers* containing the *effective addresses* of the desired data. The LINC instruction set provides for both types of addressing.

3.5 DIRECT ADDRESSING

In LINC programming, direct access to memory registers is limited to the Instruction Field. A full address in this field requires ten bits (0000-1777), leaving only two bits for instruction codes. The three instructions, ADD, STC, and JMP, are described in detail in Section III. The format of a direct-address instruction is shown in Figure 3-2. Bits 0 and 1 are used for the operation code, bits 2-11 for the address.

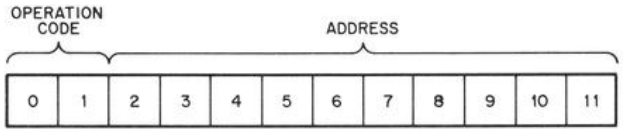


Figure 3-2. Direct Address Instruction Format

3.6 INDIRECT ADDRESSING: β-CLASS

For access to registers in the Data Field, an indirect address is required. The instruction specifies one of a small set of special registers which are used to hold the effective address of desired data. The format of these *β-class* instructions is shown in Figure 3-3. Bits 3-6 are available for operation codes; bits 8-11, together with bit 7, determine which of four addressing schemes is to be used.

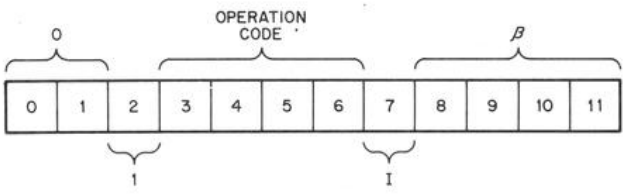


Figure 3-3 β-Class Instruction Format

3.6.1 β-Registers

In a β-class instruction, the contents of bits 8-11, when not zero, designate one of fifteen registers at locations 0001-0017 of the Instruction Field. The contents of the specified *β-register* are used to determine the effective address of the operand. When the contents of bits 8-11 are zero, the effective address is found in the register which immediately follows the referencing instruction.

Bit 7, the *I-Bit*, determines the manner in which the register designated by bits 8-11 is to be used in locating the operand. There are four addressing schemes, described in the following table:

Bit 7 *(I)*	*Bits 8-11* *(β)*	*Effective Address*
0	00	The contents of bits 1-11 of the register immediately following the instruction.
1	00	The address of the register immediately following the instruction. The operand itself is in this register.
0	01-17	The contents of bits 1-11 of the designated β-register.
1	01-17	The contents, *incremented by 1*, of bits 1-11 of the designated β-register. Ten-bit indexing is used (see text).

In the first scheme, the register which follows the referencing instruction contains the effective address. In the second scheme, the operand itself is in that register.

When either of these two schemes is used, that is, when the contents of bits 8-11 are zero, the program counter automatically skips over the register immediately following the instruction, and the next instruction is fetched from the second register following.

The following examples illustrate the use of all four addressing schemes.

The instruction STA (Store Accumulator) causes the contents of the AC to be stored in memory. The operation code for STA is 1040. The register R is the one which immediately follows that containing the STA instruction.

(1) STA 0	Octal Code: 1040 I=0, β=00 Destination of C(AC):	C(R)=2345 Location 2345
(2) STA I 0	Octal code: 1060 I=1, β=00 Destination of C(AC):	 Location R
(3) STA 12	Octal code: 1052 I=0, β=12 Destination of C(AC):	C(0012)=3456 Location 3456

(4) STA I 12	Octal code: 1072 I=1, β=12	C(0012)=3456
	Destination of C(AC):	Location 3457 (The contents of β-register 0012 are incremented by 1, and the result, 3457 is used as the effective address.)

In the next example, the use of these addressing schemes in a program sequence is demonstrated. The instruction ADA (Add to Accumulator) adds the operand to the contents of the AC, leaving the result in the AC. The program sequence starting at location 1000 adds the numbers N_1, N_2, N_3, and N_4, leaving the sum in the AC.

Instruction Field Address	*Contents*	*Octal Code*	*Remarks*
0000			
0001			
....			
0007	1500		Replaced by 1501 after indexing
....			
0017			
....			
....			
1000	ADA	1100	Indirect through 1001. Adds N_1 to C(AC)
1001	1477		Address of N_1
1002	ADA I	1120	Direct to 1003; adds N_2 to C(AC)
1003	N_2		
1004	ADA 7	1107	Indirect through 7 to 1500. Adds N_3
1005	ADA I 7	1127	Indirect through 7, indexed. Adds N_4
....			
....			
1477	N_1		These registers may be anywhere
1500	N_3		in the Instruction or Data
1501	N_4		fields.

3.6.2 β-Register Indexing

When the β-indexing scheme is used (I = 1, β ≠ 00), effective addresses may specify registers in either memory field, but the β-register cannot be incremented from one field to the other. Indexing is only over ten bits, as it is in the PC; the two high-order bits are unaffected. Thus, the contents of the β-register will be incremented from 1777 to 0000, from 3777 to 2000, from 5777 to 4000, and from 7777 to 6000. To change access from one field to the other, it is necessary to change the contents of bit 1 of the β-register.

Bit 0 of the β-register has no effect in most indirect references, but it does have a special use in half-word operations, in multiplication, and in character display.

3.7 ADDRESSING: α-CLASS

Three LINC mode instructions – SET, XSK, and DIS – have specialized memory reference schemes. Although each of them access memory in a unique way, all make use of one of the registers in locations 0000 through 0017. These are called *α-registers,* to differentiate between these instructions and those of the β-class.

SET and XSK are described in Section III. DIS is described in Section IV.

Section III. LINC MODE INSTRUCTIONS

3.8 INSTRUCTION FORMATS

There are three basic LINC mode instruction formats, to wit:

(1) *Direct Address* (Figure 3-2)

This class consists of the three instructions ADD, STC, and JMP.

(2) *Indirect Address, β-class* (Figure 3-4)

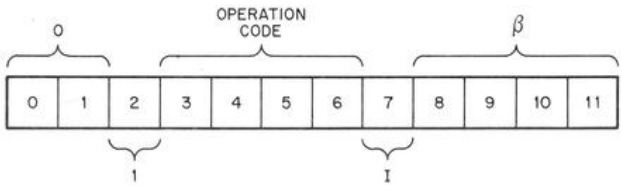


Figure 3-4. β-Class Format

This class consists of 16 β-class instructions, with operation codes between 1000 and 1740.

(3) *α-class and others* (Figure 3-5)

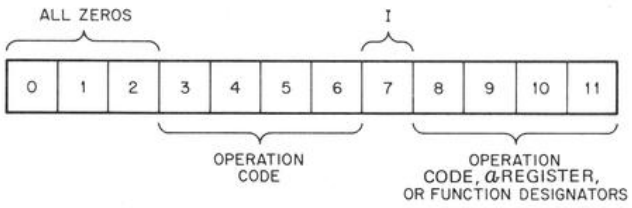


Figure 3-5. *a*-Class and Non Memory Reference Format

There are 16 basic instructions and instruction subclasses with this format, giving a total of 43 instructions with operation codes between 0000 and 0740. The I-bit has various functions depending on the subclass.

3.9 INSTRUCTION DESCRIPTIONS

The descriptions are organized according to function and class, as follows:

Full-word Data Transfers	STC, LDA, STA
Full-word Arithmetic	ADD, ADA, ADM, LAM, MUL
Full-word logic	BCL, BSE, BCO
Full-word comparison	SAE, SRO
Half-word operations	LDH, STH, SHD
a-class operations	SET, XSK
Program control	JMP
Shift and rotate	ROL, ROR, SCR
Skips	APO, AZE, LZE, QLZ, FLO, SNS, SXL, KST
Miscellaneous	HLT, CLR, COM, NOP, QAC
Console Switches	LSW, RSW
Mode control switch	PDP
I/O Bus Enable	IOB
Memory address control	LIF, LDF, IOB/IOTs
Program Interrupt	IOB/IOT, DJR
Special Functions	ESF, SFA

Instructions related to the Display, Data Terminal, and LINCtape are described in Sections IV, V and VI.

In general, the description of each instruction is presented in the following manner:

Mnemonic Operation Performed

Form

Octal code

Execution time

Operation

The second line shows the general *form* of the instruction when used in a program. The *octal code* is that of the instruction itself, plus the octal value of any other elements which may be present, such as the I-bit or β-register bits. (The I-bit, for example, being represented by bit 7, has an octal value of 20 when it is present).

3.10 FULL-WORD DATA TRANSFERS

These three instructions move complete 12-bit words between the Accumulator and Memory.

STC Store and Clear

Form: STC Y
Octal code: 4000+Y
Execution time: 3.2 microseconds
Operation: Store the contents of the AC in register Y, then clear the AC. This is a *direct address* instruction; Y must be in the Instruction Field.

LDA Load Accumulator

Form: LDA I β
Octal code: 1000 + 20I + β
Exectution time: 4.8 microseconds; 3.2 μsec. when I=1 and β=00
Operation: Place the contents of register Y, where Y is the address specified by I and C(β), in the AC. The previous C(AC) are lost; the contents of Y are unchanged.

STA Store Accumulator

Form: STA I β
Octal code: 1040 + 20I + β
Execution time: 4.8 microseconds; 3.2 μsec. when I = 1 and β=00
Operation: Store the contents of the AC in memory register Y, where Y is the address specified by I and C(β). The previous C(Y) are lost; the contents of the AC are not changed.

3.11 FULL-WORD ARITHMETIC

The instructions ADD, ADA, and ADM use *1's-complement arithmetic.* If, as the result of an addition, a 1 is carried out of bit 0 of the sum, 1 is added to the sum. This *end-around* carry is the defining property of a 1's-complement addition. If there is no carry, the sum is left as is.

Example 1:

```
  2435
 +1704
 ----
  4341
```

no carry; sum is left as is.

Example 2:

```
  2435
 +5704
 ----
 1 0341
   ->  1
 ----
  0342
```

end-around carry; 1 added to sum.

In either case, the Link is not affected.

The instruction LAM uses *2's-complement* arithmetic. If a carry from bit 0 of the sum occurs, the Link is set to 1; the sum is left unaffected.

3.11.1 Overflow

In any LINC mode addition, a number is considered to be positive if its high-order bit (bit 0) is 0, and negative if this *sign* bit is 1. Whenever two addends of like sign produce a sum of opposite sign, *overflow* is said to occur. When this happens, the Overflow flip-flop, FLO FF, is set to 1. If no overflow occurs, FLO FF is set to 0. Overflow cannot, by definition, occur when the addends have unlike signs. Note that overflow and carry are *not* the same thing.

3.11.2 INSTRUCTIONS

ADD Add to Accumulator (Direct Address)

Form: ADD Y
Octal code: 2000 + Y
Execution time: 3.2 μsec
Operation: The contents of register Y are added to the contents of the AC using 1's-complement addition, leaving the sum in the AC. The previous C(AC) are lost; the Link and C(Y) are not changed.

ADA Add to Accumulator (β-Class)

Form: ADA I β
Octal code: 1100 + 20I + β
Execution time: 4.8 μsec; 3.2 μsec when I=1 and β=00
Operation: The contents of register Y, as specified by I and C(β), are added to the contents of the AC using 1's-complement addition, leaving the sum in the AC. The previous C(AC) are lost; the Link and C(Y) are not changed.

ADM Add to Memory

Form: ADM I β
Execution time: 4.8 μsec; 3.2 μsec when I=1 and β=00
Operation: The contents of register Y, as specified by I and C(β), are added to the contents of the AC using 1's-complement addition, leaving the sum in both the AC and Y. The previous contents of both registers are lost; the Link is not changed.

LAM Link Add to Memory

Form: LAM I β
Octal code: 1200 + 20I + β
Execution time: 4.8 μsec; 3.2 μsec when I=1 and β=00

NOTE

The description of the operation, below, presents the logical sequence of events; in practice, the operations are carried out simultaneously.

Operation: The contents of the Link are added to the contents of the AC using 2's complement addition, leaving the sum in the AC. If there is a carry out of bit 0, the Link is set to 1; if not, the Link is cleared. Next the contents of register Y, as specified by I and C(β), are added to the new contents of the AC, again using 2's-complement addition; the sum is left in both the AC and Y. If there is a carry from bit 0 this time, the Link is set to 1; if not, the Link is left unchanged.

Example: C(AC) = 3743. C(Y) = 6517 C(L) = 1.

(1)	C(AC)	3743	
	+ C(L)	+ 1	
		3744	no carry; Link is cleared.
(2)	C(AC)	3744	
	+ C(Y)	+6517	
		1 2463	carry; Link is set to 1.

Results: C(AC = 2463
C(Y) = 2463
C(L) = 1

MUL Multiply

Form: MUL I β
Octal code: 1240 + 20I + β
Execution time: 9.6 microseconds; 8 μsec. when I=1 and β=00

Operation: The contents of the AC (multiplicand) are multiplied by the contents of register Y (multiplier). The product is left in the AC and the MQ. The sign of the product appears in the Link, and AC_{00}.

The multiplier and multiplicand are treated as 12-bit 1's-complement numbers. If bit 0 of an operand is set to 1, the operand is negative. The sign of the product is always correct, that is, operands of like sign give a positive product, and operands of unlike sign give a negative product. Overflow cannot occur; the Overflow flip-flop is not affected by multiplication.

Either integer or fractional operands may be specified, as follows: if bit 0 of the designated β-register contains a 0, the operands are treated as integers; the binary points of both multiplier and multiplicand are considered to be to the right of bit 11. If C(β_0) = 1, the operands are taken as fractions; the binary points are considered to be between bit 0 (sign) and bit 1. Note that when I = 1 and β = 00, there is no effective address. In this case, integer multiplication is performed.

When integer multiplication is performed, the low-order 11 bits of the product appear in bits 1-11 of the AC and the absolute value of the low order 11 bits of the product appear in bits 0-10 of the MQ. The sign appears in AC_0 and the Link. The high-order bits of the product are lost.

When fractional multiplication is performed, the high-order 11 bits of the product appear in AC_{1-11}, and the absolute value of the low-order bits appear in bits MQ_{0-10}. The sign appears in AC_0, and the Link. The contents of the MQ can be accessed by using the QAC instruction (see page 3-21).

Examples: (all octal form)

(a) Integers

1.

```
    0432      C(AC)
   x0006      C(Y)
00003234      product      C(AC)=3234  C(MQ)=6470
```

2.

```
    2764
   x0153
00476374                   C(AC)=2374  C(MQ)=4770
```

3.

```
    2764
   x7624      (=-153)
77301403      (=-00476374) C(AC)=5403  (=-3154)   C(MQ)=4770
```

(b) Fractions

1.

```
    0432      C(AC)
   x0006      C(Y)
00003234      product      C(AC)=0000  C(MQ=6470
```

2.

```
    2764      C(AC)
   x0153      C(Y)
00476374      product      C(AC)=0117  C(MQ)=4770
```

3.

```
    2764      C(AC)
   x7624      C(Y)
77301403      (-00476374)  C(AC)=7660  C(MQ)=4770
```

3.12 FULL-WORD LOGIC

In each of these Boolean functions, the operation is performed between corresponding bits of the AC and the operand, independent of the other bits in either word.

BCL Bit Clear

Form: BCL I β
Octal code: $1540 + 20I + \beta$
Execution time: 4.8 μsec; 3.2 μsec when I=1 and β=00
Operation: For each bit of the operand that is a 1, the corresponding bit of the AC is cleared to 0. For each operand bit that is a 0, the corresponding AC bit is unchanged. The operand is not changed. The following truth table gives the relationship between the corresponding bits, with the results of the comparison.

$C(Y_j)$ \ $C(AC_j)$	0	1
0	0	1
1	0	0

The Boolean statement of this relation is $AC \vee \overline{Y}$

Example		
	C(AC)=2307	010011000111
	C(Y) =1616	001110001110
	Result: =2101	010001000001

BSE Bit Set

Form: BSE I β
Octal code: $1600 + 20I + \beta$
Execution time: 4.8 μsec; 3.2 μsec when I=1 and β=00
Operation: For each bit of the operand that is a 1, the corresponding bit of the AC is set to 1. For each operand bit that is 0, the corresponding AC bit is not changed. The operand is not affected. The truth table for this relation, which is the familiar inclusive OR, is given below:

$C(Y_j)$ \ $C(AC_j)$	0	1
0	0	1
1	1	1

The Boolean statement of this relation is AC V Y.

Example:

C(AC) = 2307	010011000111
C(Y) = 1616	001110001110
Result: = 3717	011111001111

BCO Bit Complement

Form: BCO I β
Octal code: $1640 + 20I + \beta$
Execution time: 4.8 μsec; 3.2 μsec when I=1 and β=00
Operation: For each bit of the operand that is a 1, the corresponding bit of the AC is complemented. For each operand bit that is 0, the corresponding AC bit is unchanged. The operand is not changed. The truth table for this relation, which is the exclusive OR, is given below:

$C(Y_j)$ \ $C(AC_j)$	0	1
0	0	1
1	1	0

The Boolean statement of this relation is AC $\forall$ Y

Example:

C(AC) = 2307	010011000111
C(Y) = 1616	001110001110
Result: = 3511	011101001001

3.13 FULL-WORD COMPARISON

In both of these operations, the next succeeding instruction in the program sequence is skipped if the stated condition is met. When $\beta \neq 00$, this presents no unusual circumstance. When β=00, however, the register immediately following the skip instruction contains either the operand itself or its address. When such is the case, this register is automatically passed over, and the one beyond that is considered to be the next register in the program sequence. If a skip occurs under these conditions, the program will proceed from the *third* register following the skip instruction.

SAE Skip If Accumulator Equal (To Operand)

Form: SAE I β
Octal code: $1440 + 20I + \beta$
Execution time: 4.8 μsec ; 3.2 μsec when I=1 and β=00
Operation: If the contents of the Accumulator are equal to the contents of Y (where Y is specified by I and β), the next instruction in the program sequence is skipped. Otherwise, the program continues without skipping. The contents of the AC and of Y are not changed.

SRO Skip and Rotate

Form: SRO I β
Execution time: 4.8 μsec.; 3.2 μsec. when I=1 and β=00
Operation: If bit 11 of the operand is 0, the next instruction in the program sequence is skipped; Otherwise, the program proceeds without skipping. After the bit is tested, the contents of Y (that is, the operand) are rotated right one place.

Example:

Address	*Contents*	*Action*
p	SRO I 0	/The operand is in register p + 1.
p+1	3725	/Operand. $C(R_{11}) = 1$, so no skip.
p+2	....	/Program continues from here.
p+3		

After the test is performed, the contents of p+1 are rotated right one place; the result, which is retained in p+1, is 5752. If the instruction in register p were now to be executed again, the skip would occur, because the new contents of bit 11 of p+1 equal 0. The program would then proceed from register p+3, skipping the instruction in p+2.

3.14 HALF-WORD OPERATIONS

3.14.1 Half-Word Addressing

The three instructions, LDH, STH, and SHD, operate on either half of a memory register, independent of the other half. The addressing scheme is basically that of other β-class instructions, with the following difference: Whenever bit 0 of the register containing the effective address holds a 0, the *left* half of the addressed operand is used; when bit 0 contains a 1, the *right* half is used. In either case, the data are transferred or compared between the designated half of the operand and the right half of the Accumulator.

The following examples demonstrate the effects of half-word addressing. The instruction LDH transfers the designated half of the operand into the right half of the AC; the left half of the AC is cleared.

1. LDH 0 I = 0, β = 00. C(R) = 0370 (R is the register following LDH). The effective address is 0370. Because $C(R_0) = 0$, the contents of the left half of register 0370 are placed in the right half of the AC.

2. LDH 12 I = 0, β = 12. C(0012) = 4370. Bit 0 of β-register 12 contains a 1; therefore, the contents of the right half of register 0370 are placed in the right half of the AC.

3. LDH I 0 I = 1, β = 00. C(R) = 6527. This is a direct reference, so that there is no explicit effective address. In this case, the *left* half of the operand in register R is taken. In the example, the quantity 65 is placed in the right half of the AC.

4. LDH I 12 I = 1, β = 12. C(0012) = 0370.

The effective address, that is, C(0012), must be incremented before it is used. Instead of 1, however, 4000 is added to the contents of the β-register (remember that the half-word indicator is in bit 0). Given the conditions specified above, the contents of β-register 12 are first augmented from 0370 to 4370, and the right half of the operand in register 0370 is taken. If a second LDH I 12 is now executed, the β-register is again incremented by 4000. The sum

which leaves $C(\beta_0)=0$, results in a carry out of the high-order bit. The carry causes 1 to be added to the sum, resulting in a final effective address of 0371. The new operand, then, is taken from the left half of the new register. The indexing sequence is thus: left half, right half, left half of the next succeeding register, etc.

Because the basic indexing scheme is operative only over bits 2-11 of the β-register, half-word addressing proceeds from the right half of register 1777 to the left half of register 0000, and from the right half of register 3777 to the left half of register 2000. $C(\beta)$ are thus incremented from 1777 to 5777 to 0000, and from 3777 to 7777 to 2000.

NOTE

Another way of looking at the half-word indicator may help clarify this method of addressing. If the indicator is considered to be just to the right of bit 11, rather than in bit 0, it becomes apparent that half-word indexing is just like full-word indexing, with 1 added to the low-order bit, that is, the half-word indicator, each time. You can, if you like, imagine a binary point between the half-word indicator and bit 11 of the β-register, so that successive addresses might be read as 0370, 0370½, 0371, 0371½, 0372, etc. (Or, in octal, 0370.0, 0370.4, 0371.0, 0371.4, 0372, etc.).

LDH Load Half

Form: LDH I β
Octal code: $1300 + 20I + \beta$
Execution time: 4.8 μsec.; 3.2 μsec. when I=1 and β=00
Operation: The contents of the designated half of register Y (where Y is specified by I and $C(\beta)$ are placed in the right half of the Accumulator. The left half of the AC is cleared. The previous $C(AC_r)$ are lost. The contents of Y are not changed.

STH Store Half

Form: STH I β
Octal code: $1340 + 20I + \beta$
Execution time: 4.8 μsec.; 3.2 μsec. when I=1 and β=00
Operation: The contents of the right half of the Accumulator are stored in the designated half of register Y. The contents of the AC and of the other half of Y are not disturbed.

SHD Skip If Half Differs

Form: SHD I β
Octal code: $1400 + 20I + \beta$
Execution time: 4.8 μsec.; 3.2 μsec. when I=1 and β=00
Operation: If the contents of the designated half of register Y are not equal to the contents of the right half of the AC, the next instruction in the program sequence is skipped; otherwise, the program proceeds without skipping. The contents of Y and of the AC are not changed. As in the other β-class skips (SAE, SRO), the register immediately following the SHD is automatically passed over when β=00.

3.15 α-CLASS OPERATION

Each of these instructions uses the registers 0000-0017 in a unique way. A third α-class instruction, DIS, is described in Section IV, *CRT Display*.

SET Set α-Register

Form: SET I α
Octal code: 0040 + 20I + α
Execution time: 6.4 μsec.; 4.8 μsec. when I=1
Operation: The contents of the α-register specified by bits 8-11 of the SET instruction are replaced by the operand, whose location is determined by the state of the I-bit, as follows:

If I = 1, the operand is in the register immediately following that containing the SET instruction.

If I = 0, the effective address of the operand is in the register immediately following that containing the SET instruction.

SET always requires two successive locations; the program always continues from the second register following, as in this example:

Address	*Contents*	*Action*
p	SET I 15	/THE OPERAND IS IN REGISTER p+1
p+1	2537	/OPERAND. 2537 IS STORED IN REGISTER 15
p+2	. . . .	/PROGRAM CONTINUES FROM THIS REGISTER

The previous contents of the α-register are lost. The AC is not disturbed, and the contents of the register containing the operand are not changed.

XSK Index and Skip

Form: XSK I α
Octal code: 0200 + 20I + α
Execution time: 3.2 μsec
Operation: If I=1, the contents of the designated α-register are incremented by 1, using 10-bit 2's complement addition as in β-class indexing. If I=0, α are left undisturbed. Then, if the contents of bits 2-11 of the α-register are equal to 1777, the next instruction in the program sequence is skipped. Otherwise, the skip does not occur.

When C(α) are incremented, the two high-order bits are not affected. Thus, 1777 is incremented to 0000, 3777 to 2000, etc.

3.16 PROGRAM CONTROL

JMP Jump

Form: JMP Y
Octal code: 6000 + Y
Execution time: 3.2 μsec when Y ≠ 0000; 1.6 μsec when Y = 0000
Operation: The quantity Y is placed in bits 2-11 of the Program Counter, the next instruction is taken from register Y. The program proceeds from that point.

If Y≠000, the 10-bit address of the register immediately following the JMP instruction, i. e., the contents of the Program Counter, is stored in location 0000 of the Instruction Field, as a JMP instruction. This permits the JMP to be used not only as an unconditional transfer of program control, but also as a subroutine calling instruction. (See the example below.)

If Y=0000, the jump is executed, but nothing is stored in register 0000. JMP 0 is used to return from a subroutine, as shown in the example.

Example:

Address	*Contents*	*Action*
0000	0000	/WILL CONTAIN 6573 AFTER JMP 175
. . . .		/IS EXECUTED
. . . .		
0175	. . . .	/START OF SUBROUTINE
. . . .		
. . . .		
0242	JMP 0	/RETURN FROM SUBROUTINE
. . . .		
. . . .		
0572	JMP 175	/JUMP TO SUBROUTINE AT REGISTER 0175
0573	. . . .	/SUBROUTINE RETURNS TO THIS LOCATION

When JMP 175 is executed, $C(PC_{2-11}) = 0573$. This, combined with 6000, the octal code for JMP, is placed in register 0000. When the subroutine has finished, the JMP 0 transfers program control to register 0000, where JMP 573 is executed, returning control to the calling program. (At the same time, JMP 1 is stored in register 0000, but that is incidental to the actions of interest here.)

When a new Instruction Field has been selected (see paragraph 3.22, MEMORY ADDRESS CONTROL), the first JMP Y (Y≠0000) following the field selection performs the actual switching of the field; the target register of the JMP is in the new field, and the return jump is stored in register 0000 of the new Instruction Field JMP 0 has no effect on the field registers.

3.17 SHIFT AND ROTATE OPERATIONS

These instructions rotate the contents of the Accumulator left or right, or shift them right (scaling), propagating the sign bit. A single instruction can cause a shift of up to 17_8 bit positions, or 1½ times the length of the AC. On shifts or rotations right, the MQ is treated as a 12-bit extension of the AC, so that bits shifted out of AC_{11} enter MQ_0, as shown in Figures 3-6, 3-7, and 3-8.

In all these operations, the Link is included when I = 1, and excluded when I = 0. Execution times depend on the length of the shift.

ROL Rotate Left

Form: ROL I N
Octal code: $0240 + 20I + N, 0 \leq N \leq 17_8$
Execution time: 1.6–7.0 μsec
Operation: The contents of the AC are rotated left N places. If I=1, the Link is included. The rotation scheme is shown in Figure 3-6. The contents of the MQ are not affected.

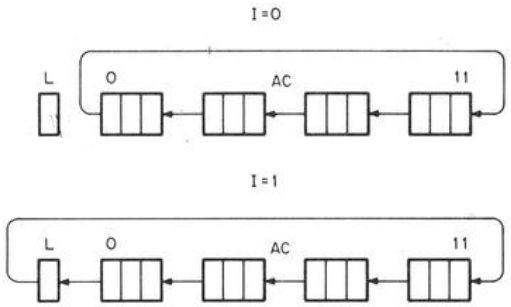


Figure 3-6. Rotate Left

ROR Rotate Right

Form: ROR I N
Octal code: $0300 + 20I + N, 0 \leqslant N \leqslant 17_8$
Execution time: 1.6–5.0 μsec
Operation: The contents of the AC are rotated right N places. Bits shifted out of AC_{11} enter MQ_0, and are shifted down the MQ. Bits shifted out of MQ_{11} are lost. If I=1, the Link is included in the rotation. The scheme is shown in Figure 3-7.

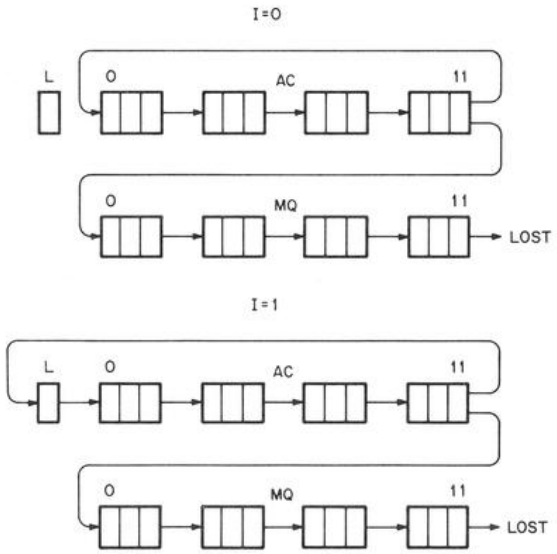


Figure 3-7. Rotate Right

SCR Scale Right

Form: SCR I N
Octal code: $0340 + 20I + N, 0 \leqslant N \leqslant 17_8$
Execution time: 1.6-5.0 microseconds
Operation: The contents of the AC are shifted right N places. The sign bit (contents of AC_0) is not changed, and is replicated in the N bits to the right of AC_0. Bits shifted out of AC_{11} enter MQ_0, and are shifted down the MQ. Bits shifted out of MQ_{11} are lost. If I=1, bits shifted out of AC_{11} also enter the Link, so that at the completion of the operation, $C(L)=C(MQ_0)$. If I=0, the Link is unaffected. The shifting scheme is shown in Figure 3-8.

Example:

C(AC) = 4371 C(MQ) = 0000

Instruction: SCR I 6

Because I = 1, the Link will receive the contents of AC_{11} at each shift of position. The result of the operation:

C(AC) = 7743	The sign bit, which was 1, was replicated in the vacated bits (AC_{1-6})
C(MQ) = 7100	Bits shifted out of the AC entered the MQ at the high-order end.
C(L) = 1	The last bit shifted out of AC_{11} was a 1. Check: $C(L) = C(MQ_0)$.

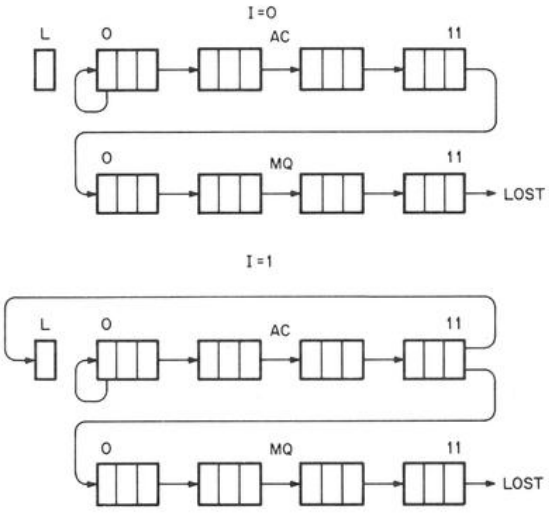


Figure 3-8. Scale Right

3.18 SKIPS

These instructions test the states of various registers, flip-flops, and external inputs. In every case, the next succeeding instruction in the program sequence is skipped if

(1) I = 0 and the condition is met
or
(2) I=1 and the condition is not met

Otherwise, the program proceeds without skipping.

APO Accumulator Positive

Form: APO I
Octal code: 0451 + 20I
Execution time: 1.6 μsec
Condition: The sign bit (contents of AC_0) is 0, that is, C(AC) is a positive number.

AZE Accumulator Zero

Form: AZE I
Octal code: 0450 + 20I
Execution time: 1.6 μsec
Condition: The contents of the AC equal 0000 (*+0*) or 7777 (-0)

LZE Link Zero

Form: LZE I
Octal code: 0452 + 20I
Execution time: 1.6 μsec
Condition: The contents of the Link equal 0.

QLZ MQ Low-Order Bit Zero

Form: QLZ I
Octal code: 0455 + 20I
Execution time: 1.6 μsec
Condition: The contents of MQ_{11} equal 0. (This is identical to the LINC-8 instruction, ZZZ.)

FLO Overflow

Form: FLO I
Octal code: 0454 + 20I
Execution time: 1.6 μsec
Condition: The overflow flip-flop (FLO FF) is set to 1. When overflow occurs as the result of an addition (ADD, ADA, ADM, or LAM), FLO FF is set to 1. When no overflow occurs, FLO FF is cleared.

The following skips test for various external input conditions.

SNS Sense Switch

Form: SNS I N
Octal code: 0440 + 20I + N, $0 \leqslant N \leqslant 5$
Execution time: 1.6 μsec
Condition: Sense Switch N on the Operator's Console is set to 1. If I=1, the skip will occur when the selected switch is set to 0.

SXL Skip On External Level

Form: SXL I N
Octal code: 0400 + 20I + N, $0 \leqslant N \leqslant 17_8$
Execution time: 1.6 μsec
Condition: An external input level is +3v. If I=1, the skip will occur when the external level is at ground (0V). In the basic PDP-12, only three of these levels have been defined; the others are available for the user's options. These external levels are digital inputs to the I/O bus, and should not be confused with the analog inputs to the A-D Converter.

NOTE

The connection for the External Level lines is made in the I/O Bus cables (See Chapter 5).

The three defined levels and their mnemonics are:

SXL I 15 (KST)	Key Struck (See below)
SXL I 16 (STD)	Tape Instruction Done (See Section VI)
SXL I 17 (TWC)	Tape Word Complete (See Appendix A)

KST Key Struck

Form: KST I
Octal code: 0415 + 20I
Execution time: 1.6 μsec
Condition: A key has been struck on the ASR-33 keyboard, the character code has been assembled in the Teletype buffer, and the Keyboard flag is raised. (The flag is cleared when the character is read into the AC).

3.19 MISCELLANEOUS

These instructions perform various tasks. Each is self-contained; they require no memory references, and the I-bit has no effect.

HLT Halt

Octal code: 0000
Execution time: 1.6 μsec to fetch and decode
Operation: The computer stops. The contents of the AC, MQ, Link, and other active registers and flip-flops are not affected. The Program Counter contains the address of the register immediately following the HLT. If the operator presses CONTINUE, the program resumes from the point indicated by the C(PC).

CLR Clear

Octal code: 0011
Execution time: 1.6 μsec
Operation: The AC, MQ, and Link are cleared to zero. No other registers or flip-flops are affected.

COM Complement AC

Octal code: 0017
Execution time: 1.6 μsec
Operation: The contents of the AC are complemented. Bits containing zeros are changed to contain 1's, and vice versa. No other registers are affected.

NOP No Operation

Octal code: 0016
Execution time: 1.6 μsec
Operation: None. Nothing happens. NOP provides a 1.6 microsecond delay and is often used to hold a place in the program for instructions which might be changed during the course of execution.

QAC Place MQ in AC

Octal code: 0005
Execution time: 1.6 μsec
Operation: The contents of bits 0-10 of the MQ are placed in bits 1-11 of the AC. AC_0 is cleared. This instruction provides access to the low-order bits of a fractional product. Figure 3-9 shows the transfer path.

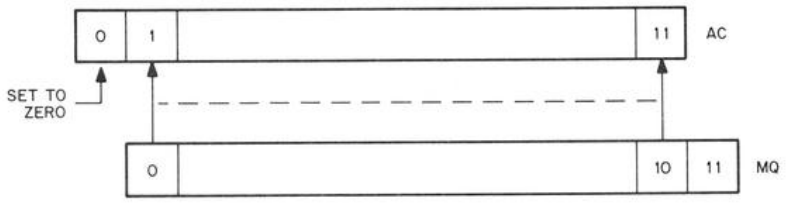


Figure 3-9. QAC Transfer Path

To obtain all 12 bits of the MQ, the following program sequence may be used:

Instruction	*Action*
QAC	/C(MQ_{0-10}) PLACED IN AC_{1-11}
ROL 1	/ROTATE C(AC) LEFT 1 PLACE, WITHOUT LINK.
QLZ I	/SKIP IF C(MQ_{11}) = 1
JMP. +3	/C(MQ_{11})=0. JUMP TO THIRD REGISTER BEYOND.
BSE I	/C(MQ_{11})=1. SET AC_{11} EQUAL TO 1.
0001	/OPERAND TO SET AC_{11}

(QAC is identical to the LINC-8 instruction, ZTA).

3.20 CONSOLE SWITCHES

These instructions provide access to the states of the switches in the Left and Right Switch Registers on the Operator's Console. The I-bit has no effect in these instructions.

LSW Left Switches

Octal code: 0517
Execution time: 1.6 μsec
Operation: The contents of the Left Switches Register on the Console are placed in the AC. The previous C(AC) are lost.

RSW Right Switches

Octal code: 0516
Execution time: 1.6 μsec
Operation: The contents of the Right Switches Register are placed in the AC. The previous C(AC) are lost.

3.21 MODE CONTROL

PDP Switch To PDP-8 Mode

Octal code: 0002
Execution time: 1.6 microseconds
Operation: Beginning with the next succeeding instruction, the central processor will operate in PDP-8 mode; all subsequent instructions are interpreted as PDP-8 operations. A similar instruction, LINC, in the PDP-8 mode instruction set, causes a switch to LINC mode.

Input/Output Bus

In addition to the input and output devices controlled directly by LINC instructions (see Sections IV, V, and VI), the LINC mode program also has direct access to any device connected to the PDP-12 I/O Bus. By using the special enabling instruction, IOB, he may include any PDP-8 mode IOT command within his LINC program sequence.

This access is provided by the special two-word instruction, IOB.

IOB I/O Bus Enable

Form: I/O (first word) IOT (second word)
Octal code: 0500 (first word)
Execution time: 5.9 μsec
Operation: The IOT timing chain is activated by the second word of this instruction. Bits 3-11 of this second word are interpreted as a PDP-8 IOT command; bits 0-2 have no effect.

Example 1:

The following sequence may be used to read and store a character from the high-speed tape reader:

```
....
IOB          /ENABLE I/O BUS
RRB          /READ TAPE READER BUFFER
STA 14       /STORE IN REGISTER SPECIFIED BY C(0014)
....
```

Example 2 The following sequence tests the high-speed reader flag:

```
. . . .
IOB          /ENABLE I/O BUS
RSF          /SKIP IF HIGH-SPEED READER FLAG IS SET
JMP .-2      /NOT SET. GO BACK TWO SPACES.
IOB          /FLAG IS SET. ENABLE THE BUS, AND . . .
RRB          /. . . READ THE CHARACTER.
```

Several IOB/IOT pairs are used in LINC Memory Address Control and Program Interrupt operations.

NOTE

In the skip loop, the program must jump back two locations, because the IOB must be executed each time.

3.22 MEMORY ADDRESS CONTROL

The two memory field segments used by a LINC program are program-selectable. The assignments are made by setting the two 5-bit Memory Field Registers, which can address any of 32 1024-word memory segments. Considered with respect to the physical configuration of memory, the three high-order bits of each Field Register determine which 4096-word memory bank is to be used, while the two low-order bits specify one of the four segments within that bank. The two LINC memory fields need not be adjacent, in any particular order, or even in the same memory bank. Normally, however, they would not be assigned to the same segment.

In addition to the Instruction Field (IF) and Data Field (DF) Registers described in Chapter 1, the PDP-12 Memory Control contains two other registers of interest to the LINC-mode programmer.

3.22.1 Instruction Field Buffer (IB) 5 Bits

This register holds the number specifying a new Instruction Field. Once loaded, its contents are transferred to the IF at the occurrence of the next JMP Y instruction.

3.22.2 Save Field Register (SF) 10 Bits

NOTE

For historical reasons the SF is also called *Interrupt Buffer* in some places.

Whenever the Instruction Field is changed, either by programmed action or by a program interrupt or trap, the contents of the IF and DF are placed in the Save Field Register. From the SF, the contents of the IF and DF can be restored, so that execution of an interrupted program, for example, can be resumed. The contents of the SF can be read into the AC.

3.22.3 Memory Control Programming

The Instruction Field and Data Field registers can be loaded directly, using LINC mode instructions.

LIF Load LINC Instruction Field Buffer

Form: LIF N
Octal code: $0600 + N, 0 \leqslant N \leqslant 37_8$
Execution time: 1.6 μsec
Operation: The five-bit quantity N is placed in the Instruction Field Buffer (IB). The present contents of the IF and DF are transferred to the Save Field Register (SF). When the next JMP Y instruction ($Y \neq 0000$) is executed, N is transferred from the IB to the Instruction Field Register. The return JMP is stored in location 0000 of the new Instruction Field, and program control is transferred to register Y of the new Instruction Field.

The automatic saving of the IF and DF in the Save Field Register is especially useful when subroutines are called *across memory fields* – that is, when a subroutine located in a memory field other than the current one is called. The calling program can ignore the problem of transmitting to the subroutine the memory field information which the subroutine requires to obtain arguments of the call and to return control to the calling program after the subroutine has run to completion. This information is available to the subroutine by virtue of the *field-saving* property of the instruction LIF.

The execution of the LIF instruction will internally inhibit the execution of a Program Interrupt even if ION has been given. This Interrupt Inhibit lasts from the LIF instruction until the first LINC mode JMP instruction executed in the newly selected Instruction Field. The above property allows the Save Field Register to be used for *cross field* subroutine linkage in programming which uses the Program Interrupt.

Example: The program is operating in Field 2. Control is to be transferred to location 1000 of Field 5.

Address	*Action*	*Contents*
. . . .		
p	LIF 5	/5 IS PLACED IN THE IB /C(IF) AND C(DF) ARE PLACED IN THE SF.
. . . .		
p+k	JMP 1000	/C(IB) ARE TRANSFERRED TO THE /IF. JMP P+K+1 IS STORED IN REGISTER /0000 OF FIELD 5, AND THE PROGRAM /PROCEEDS FROM REGISTER 1000 OF FIELD 5.

JMP 0 has no effect on the Memory Field registers. If it is used to return to a calling program in a different field, the change of field is effected by the JMP instruction stored in register 0000 of the subroutine's field.

(LIF replaces the LINC-8 instruction, LMB)

LDF Load LINC Data Field Register

Form: LDF N
Octal code: $0640 + N, 0 \leqslant N \leqslant 37_8$
Execution time: 1.6 μsec
Operation: The 5-bit quantity N is placed in the Data Field Register. All subsequent indirect references to the Data Field are made to the newly selected field. The previous C(DF) are lost. The contents of the other Memory Control registers are not affected.

(LDF is identical to the LINC-8 instruction, UMB)

The contents of the Memory Field Registers can be examined by using the following IOB/IOT pairs.

IOB
RIF Read Instruction Field

Octal code: 0500 6224
Execution time: 5.9 μsec, including IOB
Operation: The contents of the Instruction Field Register are placed in bits 6-10 of the AC. The remaining AC bits are unaffected, and the contents of the IF are unchanged.

IOB
RDF Read Data Field

Octal code: 0500 6214
Execution time: 5.9 μsec, including IOB
Operation: The contents of the Data Field Register are placed in bits 6-10 of the AC. The remaining AC bits are unaffected, and the contents of the DF are not changed.

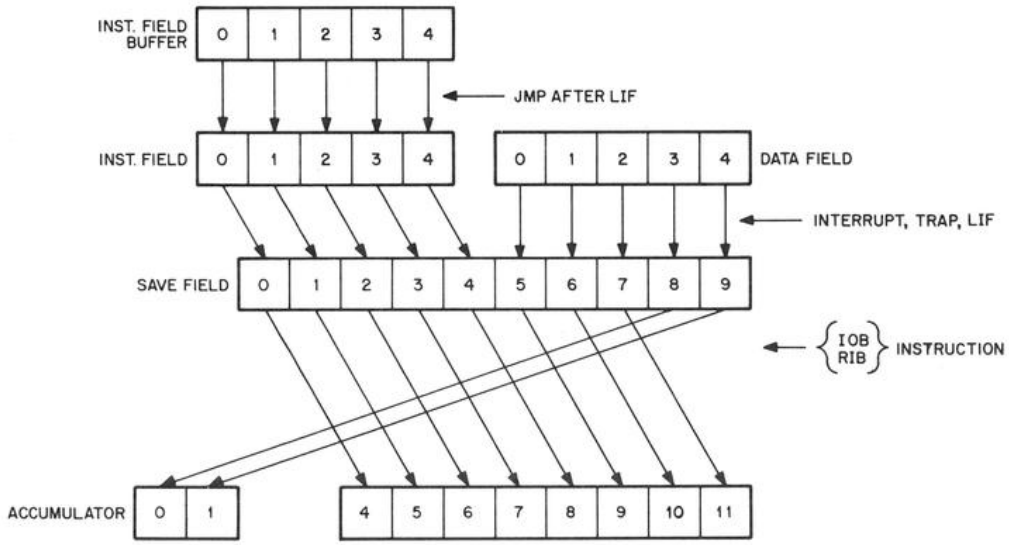


Figure 3-10. Data Path: IB, IF, DF, and AC

3.23 PROGRAM INTERRUPT IN LINC MODE

To facilitate the handling of data being transmitted to and from several peripheral devices, the PDP-12 includes a Program Interrupt Facility. When an external device is ready for servicing, a signal (flag) associated with that device is set. If the Interrupt is enabled at the time a flag is raised, the following sequence of events will occur:

1. The instruction being executed at the time of the interrupt request is completed.

2. The contents of the Program Counter are stored in register 0040 of memory field 0 (regardless of the current Instruction Field assignment).

3. The contents of the Memory Field registers are placed in the Interrupt Buffer, as shown in Figure 3-10.

4. Program execution proceeds from register 0041 of Memory Field 0.

The normal procedure from this point calls for the interrupt service routine beginning in location 0041 to determine which device flag caused the interrupt request, perform the appropriate tasks, then restore the Memory Field registers, re-enable the interrupt, and jump back to the interrupted program at the point of the Program Interrupt.

The interrupt control instructions and related memory field instructions are all IOB/IOT pairs.

Whenever a change of LINC Instruction Field occurs, the Program Interrupt is inhibited until the first JMP is executed *in the new field.* This allows the programmer to obtain and save the contents of the SF after the Field change, before a waiting interrupt request destroys the contents of the SF.

IOB
ION Interrupt On

Octal code: 0500 6001
Execution time: 5.9 μsec including IOB
Operation: The Interrupt Facility is enabled immediately after the next succeeding instruction (following the ION) is executed. From that point on, any interrupt request will cause the sequence of events described above. If a device flag is already raised when the Interrupt is enabled, the waiting request is serviced immediately. A one-instruction delay before enabling the interrupt ensures that the interrupt service routine can return to an interrupted program before a new request is honored, so as to avoid losing one's place.

IOB
IOF Interrupt Off

Octal code: 0500 6002
Execution time: 5.9 μsec, including IOB
Operation: The Interrupt is disabled. The facility is disabled immediately; subsequent request will not cause an interrupt until the facility is enabled again.

The next two instructions are related to the Save Field Register, wherein the original contents of the IF and DF are stored whenever the contents of the IF are being changed, by an LIF instruction, or as the result of an Interrupt request, or as the result of a Program Trap.

IOB
RIB Read Interrupt Buffer

Octal code: 0500 6234
Execution time: 5.9μsec, including IOB
Operation: The contents of the Interrupt Buffer (Save Field Register) are OR'ed into bits 0-1 and 4-11 of the AC, as shown in Figure 3-10. Bits 2 and 3 of the AC, and the contents of the SF are unchanged.

RIB is most commonly used immediately after a change of instruction field or a program trap, to save the record of the origin fields while the Program Interrupt is inhibited. (The first JMP instruction executed after a trap or change of Instruction field restores the Interrupt; a waiting request would destroy the contents of the Save Field Register.)

IOB

RMF Restore Memory Fields

Octal code: 0500 6244

Execution time: 5.9 μsec, including IOB

Operation: The contents of bits 5-9 of the SF are placed in the Data Field Register, and the contents of bits 0-4 of the SF are placed in the Instruction Field Buffer. At the next occurrence of a JMP Y instruction (Y ≠ 0000), the contents of the IB are transferred to the IF, effecting a return to the proper field after servicing an interrupt request. The data transfer path is shown in Figure 3-11.

DJR Disable JMP Return Save

Octal code: 0006

Execution time: 1.6 μsec

Operation: DJR sets a flip-flop which prevents the modification of location 0000 when the next and only the next LINC mode JMP is given. The DJR instruction is used when returning from Program Interrupt or Trap service routines. The DJR should be given prior to the ION of a return from Program Interrupt Servicing.

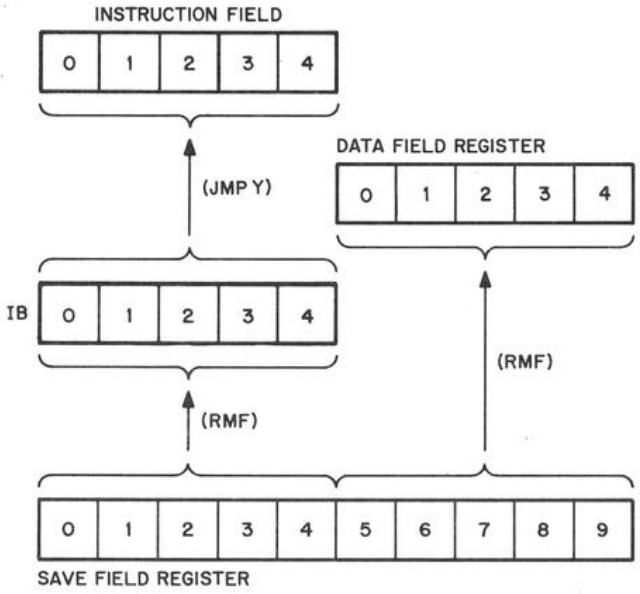


Figure 3-11. Data Path, RMF Instruction

Example: A program operating in Field 7 is interrupted while the instruction in register 0531 is being executed.

(1) C(PC) are stored in location 0040 of Field 0.

(2) C(DF) and C(IF) are placed in the SF, as shown in Figure 3-10.

(3) Program execution resumes in location 0041 of Field 0; in other words, 00 is placed in the IF, and 0041 in PC_{2-11}.

(4) The interrupt is disabled.

The interrupt service routine must do three things. First, it must set up a return jump to enable the program to get back to the point of the break. Next, it must identify the cause of the request and service the condition. Finally, it must restore the conditions prevailing at the time the interrupt occurred, and return to the main program. The following sequence shows how these tasks may be accomplished.

Address	*Contents*	*Action*
0040	0532	/CONTENTS OF PC AT TIME OF INTERRUPT
0041	STC ACSAV	/SAVE C(AC), THEN CLEAR AC
0042	ADD 0040	/SAVED ADDRESS (0532) TO AC
0043	BSE I	/MAKE JMP INSTRUCTION: C(AC) V C(0044)
0044	6000	/OCTAL CODE OF JMP
0045	STC RTN	/STORE JMP 532 AT END OF SERVICE ROUTINE
....	....	/MAIN PART OF SERVICE ROUTINE. IF NECESSARY,
....	....	/OTHER ACTIVE REGISTERS (MQ, L, ETC.) SHOULD
....	....	/BE SAVED ALSO.
	IOB	/EXIT SEQUENCE. ENABLE IOT TIMING CHAIN . . .
	RMF	/. . . AND RESTORE MEMORY FIELDS. (07 TO IB)
....	STC TEMP	/CLEAR AC WITHOUT DISTURBING MQ AND L.
	ADD ACSAV	/RESTORE ORIGINAL C(AC)
	DJR	/SET PROCESSOR SO THAT NEXT JMP INST WILL /NOT STORE IN LOCATION ZERO OF MEMORY BANK /TO WHICH JMP AT "RTN" WILL GO.
	IOB	
	ION	/RE-ENABLE INTERRUPT
RTN	(JMP 0532)	/JUMP TO ORIGINAL FIELD.

3.24 SPECIAL FUNCTIONS

A set of six Special Functions allows the LINC programmer to establish any of five operating states, or generate an I/O Preset pulse. The special functions are determined by bits 2-7 of the AC, as shown in Figure 3-12.

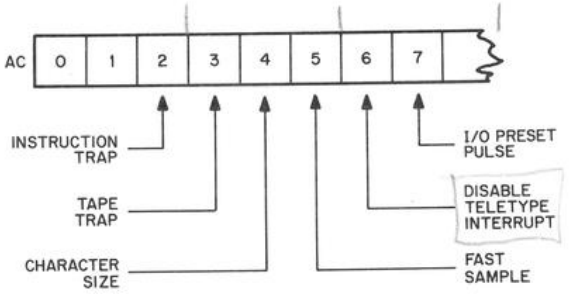


Figure 3-12. Special Functions

The functions have the following characteristics:

1. *Instruction Trap Enable.* Causes a program trap to register 0141 whenever one of these categories of LINC instruction codes is encountered:

OPR (I) 01-15	501-515, 521-535
Execute class	740-757
Undefined codes	540-577
	1700-1737

The instruction trap is described in detail in paragraph 3.25.

2. *Tape Trap.* When this function and Instruction Trap Enable are both set, a program trap to register 0141 will occur whenever a LINCtape instruction is encountered, in addition to the other trapped codes. The LINCtape instruction is not executed. Tape Trap is described in detail in Section VI.

3. *Character Size.* This function determines the size of a character displayed on the CRT by the DSC instruction. It is described in detail in Section IV.

4. *Fast Sample.* This function reverses the order of events of the SAM instruction, (i.e., read the converter buffer and then initiate a new conversion).

5. *Disable Teletype Interrupt.* Interrupt requests from the ASR-33 Keyboard or Printer are inhibited. No program interrupt will occur, even if the Interrupt Facility has been enabled, and either TTY flag is set.

6. *Generate I/O Preset.* If this bit is set when the enabling instruction (ESF) is executed, an I/O PRESET pulse is generated which clears all device flags, disables the Interrupt, clears the Tape Extended Operations Buffer, and generates the TAPE PRESET pulse. The effect is the same as if the I/O PRESET key on the Operator's Console were pressed. Other Special Functions are cleared, or in the case of CHARACTER SIZE, reset to 1. The active registers of the Central Processor are not affected, and the system continues to operate with RUN on.

Any or all of these functions may be enabled at the same time, except that they are effectively nullified if I/O PRESET is also enabled. All the Special Functions except I/O PRESET are controlled by flip-flops from the designated AC bits; the states of these flip-flops may be examined at any time.

ESF Enable Special Functions

Octal code: 0004
Execution time: 1.6 μsec
Operation: The contents of AC bits 2-6 are placed in their respectively designated function flip-flops, as shown in Figure 3-12. For each AC bit set to 1, the corresponding function is enabled. For each AC bit set to 0, the corresponding function is disabled. If AC_7 is set to 1, the I/O PRESET pulse is generated.

SFA Place Special Function Flip-Flops in AC

Octal code: 0024
Execution time: 1.6 μsec
Operation: The contents of the Special Function flip-flops are placed in their respectively designated AC bits, as shown in Figure 3-12. Bits 0, 1, and 7-11 of the AC are cleared.

3.25 INSTRUCTION TRAP

Several sets of operation codes in the LINC repertoire are undefined. The LINC programmer can make use of these codes, without having to hard-wire them, by means of subroutines and the Instruction Trap. When the Trap is

enabled (ESF with $C(AC_2) = 1$) all undefined LINC codes will cause a program trap. When the undefined code is encountered, program control is transferred to register 0141 of Memory Field 0, regardless of the current setting of the IF. The contents of the Program Counter are placed in register 140, and the contents of the IF and DF are are placed in the Save Field Register. The subroutine beginning on 0141 can examine the trapped code (using the information stored in 0140 and the SF) to determine what program-defined operations are to be performed.

These are the undefined LINC codes which cause a program trap:

Operate class	501-515, 521-535
Execute class	740-747
Undefined	540-577
Undefined	1700-1737

The most common use of the Instruction Trap is probably in the execution of programs written for the LINC-8. To facilitate the use of such programs, a LINC-8 Simulator is provided in the basic PDP-12 software package. Close study of this program, will be most helpful for the programmer using the Trap facility.

3.25.1 Tape Trap

When both the Tape Trap and Instruction Trap functions are enabled, the LINCtape instructions (codes 700-737) are also trapped. This is useful when the programmer wishes to substitute another external storage device, such as a Disk, for the LINCtape.

3.25.2 Program Interrupt and Instruction Trap

If the interrupt is enabled when an Instruction Trap occurs, the interrupt is inhibited until the execution of the first JMP after the trap. This permits the trap program to store the contents of the Save Field Register immediately after the trap, so that the record of where the trap took place is not destroyed by and interrupt request, which also causes the contents of the IF and DF to be placed in the SF.

Section IV. CRT DISPLAY

The 6.5 inch x 9 inch rectangular screen of the PDP-12 Display (Type VR12) has a total display area of 58.5 square inches. Grid dimensions are 512 x 512 points. The horizontal distance between points is 0.0176 inch; the vertical distance is 0.0127 inch. The (0,0) grid point is at the midpoint of the left side of the screen, as shown in the schematic representation in Figure 3-13. Grid co-ordinates are given in octal.

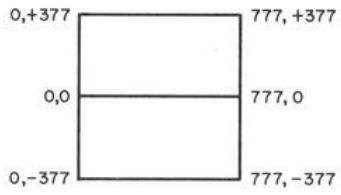


Figure 3-13. CRT Grid

The display system is fully buffered. Co-ordinates are held in two 9-bit buffers; during the execution of DSC, the pattern word is retained in a 12-bit Pattern Intensification Register. Either of two intensification channels can be specified. A switch on the CRT housing allows either or both channels to be displayed.

Below the channel selector is a continuously-variable knob which allows the user to change the intensity of the displayed points. A level control within the chassis sets the maximum brightness.

A 24-contact Blue Ribbon connector on the Data Terminal Panel allows the user to connect an auxiliary scope (VR-12A or Tektronix 503 or similar unit) for remote display of the same information sent to the main screen. The channel selectors can be set so that each scope displays one of the channels.

The pin assignments are:

1	Channel Select		
2	Not Used	9	Shield Chassis Ground
3	Shield Chassis Ground	10	Y HQ Ground
4	Intensified Pulse	11	Y Deflection
5	Not Used	12	Shield Chassis Ground
6	Shield Chassis Ground	13-18	Not Used
7	X HQ Ground	19	503 Intensify
8	X Deflection	20-24	Not Used

The output drive capability of the D-A converters is -0.3v to -6.0v capable of driving a load resistance of 1 kΩ connected to ground. This allows up to 200 feet of cable for a remote VR-12. The absolute values of the D-A outputs are not held closer than ±0.3v but are stable to within 0.3%. The D-A converters are loaded by jam transfer. The D-A used to drive the scope is also available as a single-ended output to drive external devices. The -0.3v D-A point is equivalent to the upper left hand corner of the display screen.

If a new DSC instruction is given while the previous one is still being performed by the display control, the central processor waits until the display is free before proceeding.

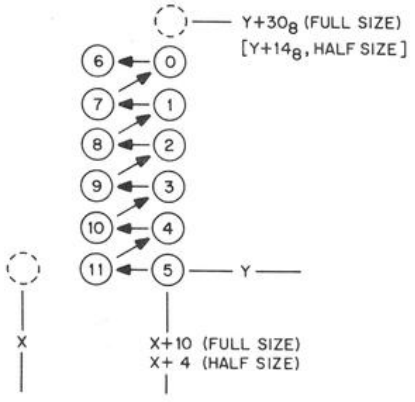


Figure 3-14. Display Pattern for DSC

3.26 HALF-SIZE CHARACTERS

If the programmer enables the HALF-SIZE CHARACTER Special Function [ESF with $C(AC_4) = 0$], all increments are by two grid positions, rather than four. Bits 3-7 of the AC provide the initial Y co-ordinate; after the two co-ordinates have been transferred to the display control's buffers, the contents of AC_{8-11} will be 14_8, and the X co-ordinate in register 0001 will be incremented by 4 instead of by 10_8. Vertical spacing is likewise halved; arrays may start at intervals of 20_8 points, with 4 points between lines.

3.27 CHARACTER SET

Each character on the ASR-33 keyboard can be represented on a 4 x 6 grid (24 points). A complete character can be displayed by using two DSC instructions, with two consecutive storage words providing the complete 24-bit character pattern. Table 3-1 lists the display patterns for the ASR-33 character set. Non-displayable characters have patterns of all zeros. The table entries, each consisting of two words, are arranged in order of ASCII codes.

NOTE

The instruction setup requires either 4.8 or 6.4 microseconds. The display control then allows 15 microseconds for point settling., 1 microsecond for each unintensified point and 3 microseconds for each intensified point. When no points remain to intensify, the display process is finished leaving the vertical and horizontal co-ordinates of the D-A at the last intensified point of the DSC instruction.

The LINC display instructions allow the programmer to display single grid points or a small array of points. In either case, the full buffering allows the program to proceed after the display operation has been initiated. If a subsequent display instruction is encountered before the previous display operation has been completed, the program will pause until the display control is free, then execute the new instruction.

3.28 POINT DISPLAYS

DIS Display

Form: DIS I a
Octal code: $0140 + 20I + a$, $0 \geqslant a \geqslant 17_8$
Execution time: 3.2 μsec; 16 μsec for completion
Operation: A single point on the screen is intensified. The vertical co-ordinate is specified by bits 3-11 of the AC; the horizontal co-ordinate by bits 3-11 of the designated a-register. If bit 0 of register a is set to 0, the point will be displayed on Channel 0; if $C(a_0) = 1$, the point will be displayed on Channel 1.

If I = 0, the contents of a are taken as is. If I = 1, $C(a)$ are first incremented by 1, using 10-bit, 2's-complement addition. Bits 0 and 1 are not affected.

3.29 CHARACTER DISPLAYS

DSC Display Character

Form: DSC I β
Octal code: $1740 + 20I + \beta$
Execution time: 4.8 μsec when I=1, β=00; 6.4 μsec when I=0 or $\beta \neq 00$ Control completion time 15-51 μsec
Operation: A vertical, 2 x 6 array of points is displayed, according to a pattern word stored in register Y. For each bit of the pattern that is a 1, the corresponding point is intensified; for each bit that is a 0, the corresponding point is left dark. In figure 3-14, the circles represent the points of the array; the small numbers refer to the corresponding bit positions of the pattern word. The small arrows show the order in which the pattern bits are examined and displayed.

As with DIS, the vertical co-ordinate is held in the Accumulator. The horizontal co-ordinate is held in register 0001; for this reason, register 0001 cannot be used as a β-register with DSC. The character may be displayed in either of two sizes: full size, in which the spacing between points in both directions is four grid positions, and half-size, in which the spacing is two positions. The following description assumes full-size characters.

When a DSC instruction is executed, the following events occur:

(1) The intensification pattern is transferred from Y to the display control Intensification Buffer.

(2) The contents of bits 3-6 of the AC are placed in the display control Y-buffer; bits 7-11 of the AC are set to 30_8.

The contents of register 0001, the X-co-ordinate, are incremented by 10_8, and transferred to the display control X-buffer.

The foregoing operations take 4.8 or 6.4 μsec. to do their work, after which the central processor is free to resume program execution. The remaining operations are performed by the display control.

(3) The pattern word is examined according to the order shown in Figure 3-15. The time required to scan and display the points varies according to the number of points to be intensified. Each bit to be left dark requires 1 μsec.; each bit to be displayed requires 3 μsec.

Because of the manner in which the Y-co-ordinate is used, full-size character arrays may start only at intervals of 40_8 points up and down the face of the scope, e.g., 000, 040, 100, -100, etc. The array itself being only 30_8 points high, this gives the programmer an automatic vertical spacing of 10_8 points between the bottom of one line and the top of the one immediately below it.

5

No flicking in histograms ē
DSC (-0) instructions

Section V. DATA TERMINAL PANELS

The Data Terminal provides analog inputs and relay-controlled outputs for use by LINC mode programs. The facility includes the following:

Analog Inputs	Sixteen channels
Relays	Six relays for external equipment control
Auxiliary Scope	Blue-Ribbon connector for an auxiliary CRT

3.30 ANALOG INPUTS

The AD12 Analog-Digital Converter and Multiplexer consists of sixteen input channels, a multiplexer, and a 10-bit A-D converter. Eight of the channels are external inputs via phone jacks. These feed through preamplifiers to the converter. The acceptable voltage range of these inputs is ±1v.

The other eight channels are controlled by continuously-variable knobs mounted on the Data Terminal Panel. The knobs give ten turns lock-to-lock, providing the full 10-bit range of the converter (-777_8 to $+777_8$). The knobs can be used to control speeds (as in the continuous display of data from tape), set threshold levels or other test parameters, etc.

A single LINC mode instruction selects the input channel, initiates the conversion, and transfers the contents of the buffer into the AC.

SAM Sample

Form: SAM N
Octal code: $0100 + N$, $0 \leqslant N \leqslant 17_8$ (Basic system); $0 \leqslant N \leqslant 37_8$ (Extended system)
Execution time: 18.2 μsec (Normal mode); 1.6 μsec (Fast sample mode)
Operation: Input channel N is selected. In normal mode, the voltage level present at the input is sampled and converted to a 10-bit number (including sign), which is assembled in the converter buffer. When the conversion is complete, the contents of the buffer are transferred to AC bits 3-11. The sign is placed in bits 0, 1, and 2.

When the FAST SAMPLE Special Function is selected [ESF with $C(AC_5) = 1$], the order of events is reversed. The current contents of the converter buffer are transferred to the AC. Then, the specified channel is selected and a new conversion is initiated. The results of this new conversion can be read by a subsequent SAM instruction.

If a conversion is still in progress when the next SAM is encountered, the processor waits until the conversion is complete before executing the new SAM.

3.31 RELAYS

Six DPDT relays mounted on the Data Terminal Panel can be switched by LINC mode instructions, allowing the user to control the operation of various pieces of external equipment that are not interfaced directly with the PDP-12. The states of the relays can be examined at any time.

ATR AC to Relays

Octal code: 0014
Execution time: 1.6 μsec

Operation: The contents of AC bits 6-11 are transferred to the Relay Buffer; the state of each relay is determined as follows:

If the corresponding AC bit is 0, the relay is switched off.

If the corresponding AC bit is 1, the relay is switched on.

RTA Relays to AC

Octal code: 0015
Execution time: 1.6 μsec
Operation: The contents of the Relay Buffer are transferred to bits 6-11 of the AC. If the relay is off, the corresponding bit will be 0; if on, the bit will be 1. Bits 0-5 of the AC are not changed.

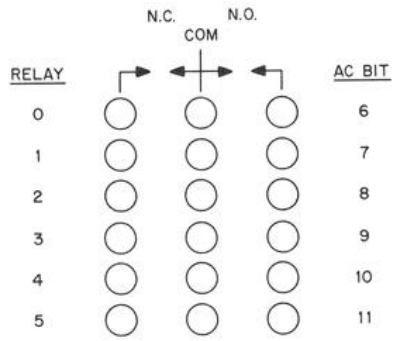


Figure 3-15. Relay Terminals and Corresponding AC Bits

Section VI. LINCTAPE

The basic LINCtape system consists of two TU55 transports controlled by a full-buffered subprocessor. A single ten-channel tape head serves for both reading and writing. Information is redundantly recorded; one line of tape contains five bits, each recorded in two non-adjacent channels, as shown in Figure 3-16. Three bits are actual data; the other two provide control information for the tape processor. The Timing track determines the position of each recorded line. Four lines are required to accommodate a full 12-bit word; the corresponding Mark Track code identifies the nature of the data word. The recording technique and tape layout are described in detail in the PDP-12 Maintenance Manual.

3.32 ORGANIZATION OF DATA

On a standard-format LINCtape, information is recorded in *blocks* of 256 12-bit words each, with identifying data at each end of the block. One LINCtape reel has a capacity of 512 standard blocks, a total of 131,072 words of data.

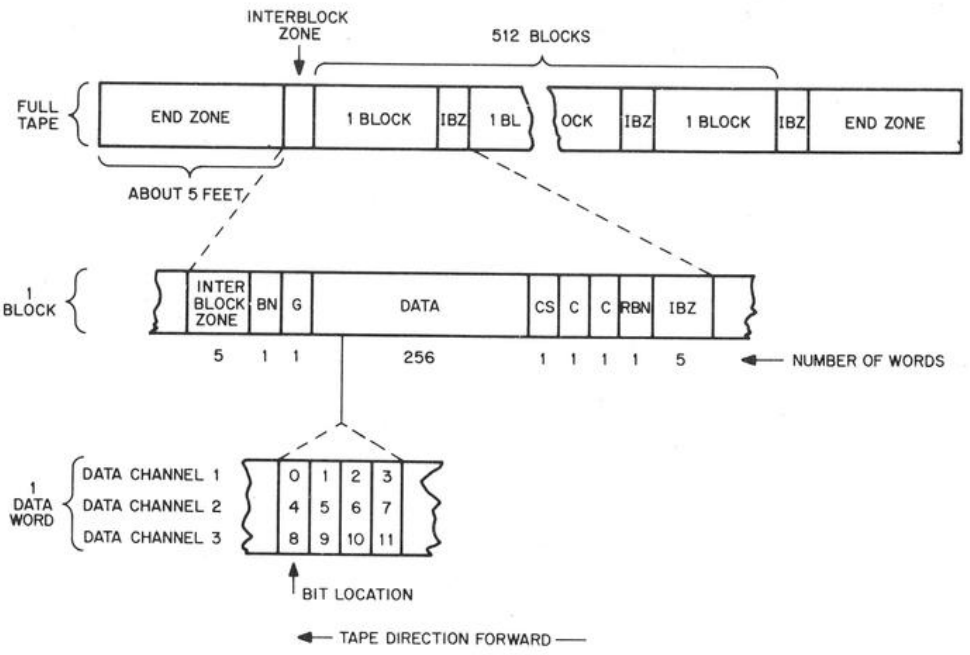


Figure 3-16. LINCtape Format

The organization of a tape is schematically presented in Figure 3-16. At each end of the tape is a long *End Zone* which allows the transport to reverse direction or come to rest without pulling the tape off the reel. Between the end zones and the terminal blocks, and between blocks, are *Interblock Zones* which can be sensed by the LINC instruction IBZ. An interblock zone is 5 words long.

A block consists of 256 data words preceded and followed by control and identifying information, as shown in the second drawing in Figure 3-16.

Block Number (BN)

This identifies the block. On a standard LINCtape, block numbers are sequential, from 0000 through 0777_8

Guard Word (G)

This protects the Block Number from transients when the read/write current is turned on and off, and allows time for the tape processor to switch from Search to Read or Write modes.

Data Words

This is the information recorded on tape from core memory. The *Final Data Word* is specially identified, to signal the end of the block. When writing a tape, this signal conditions the processor to write the checksum in the next word position.

Checksum (CS)

This is the 2's-complement (of the 12-bit sum of all the data words in the block plus the constant 7777_8). The result of adding the data sum to the checksum should be 0000; this provides a check (hence the name) on the accuracy of the transmission.

Check Words (C)	These are dummy words whose Mark Track code is the same as that for the Checksum. They are provided to insure that the Write current will be turned off before the Reverse Block Number is encountered. The Guard and Check words are not of general interest to the programmer except as they affect timing.
Reverse Block Number (RBN)	This is the complement of the Block Number, and identifies the block when tape is being searched in the reverse direction.

Subprocessor

The LINCtape subprocessor controls all information transfer between memory and tape. It is fully buffered; once an operation has been initiated by a LINC mode instruction, it is carried to completion by the tape processor. The central processor may either wait until the tape transfer is complete, or proceed immediately after the tape instruction has begun, testing at some later time for completion of the operation.

Transfers are effected in either Standard or Extended modes. In Standard mode, transfers are made to and from fixed memory locations. Extended Operations provide for a flexible addressing facility, program interrupt, and additional tape units.

As can be seen in Figure 3-17, the tape subprocessor contains seven registers which provide the transmission path for data and for control information.

Data Path

Read/Write Buffer (RWB) 12 Bits - When reading, the four lines of a data word are assembled in this register, in the bit positions shown in the third drawing of Figure 3-16. When writing, the contents of the RWB are disassembled and written on four consecutive lines of tape. Essentially, the RWB is a three-section shift register, with the three bits of a tape line entering (or leaving) the register at four-bit intervals, as indicated by the arrows in Figure 3-17.

Tape Buffer (TB) 12 Bits - When reading, the assembled word is transferred from the RWB to the TB, and from there sent to the MB in the central processor. On writing, the direction is reversed. Information from the MB enters the TB, and from there is placed in the RWB for disassembly onto the tape.

Tape Accumulator (TAC) 12 Bits - As each data word is read or written, the data sum is accumulated in the TAC. On reading, this sum is added to the Checksum read from tape, to determine whether the transfer was completed accurately. On writing, the data sum is complemented when the final data word signal is received, and the resulting Checksum is written in the word position following the final data word. The contents of the TAC can be brought into the central processor AC, using the LINC mode TAC instruction. In searching operations, the TAC also holds the sum of the desired block number and the last block number read from tape.

Control Registers

In Standard mode operations, these registers are automatically set up; in Extended Operations, the program must set the XOB and TMA Setup Register.

Tape Block Number (TBN) 12 Bits - This contains the number of the block to be accessed in a data transfer. As the tape is searched, the Block Number read from tape is compared with that in the TBN; when the numbers match, the tape is positioned so that the transfer can begin. During group operations, the TBN contains the first block number of the tape to be accessed.

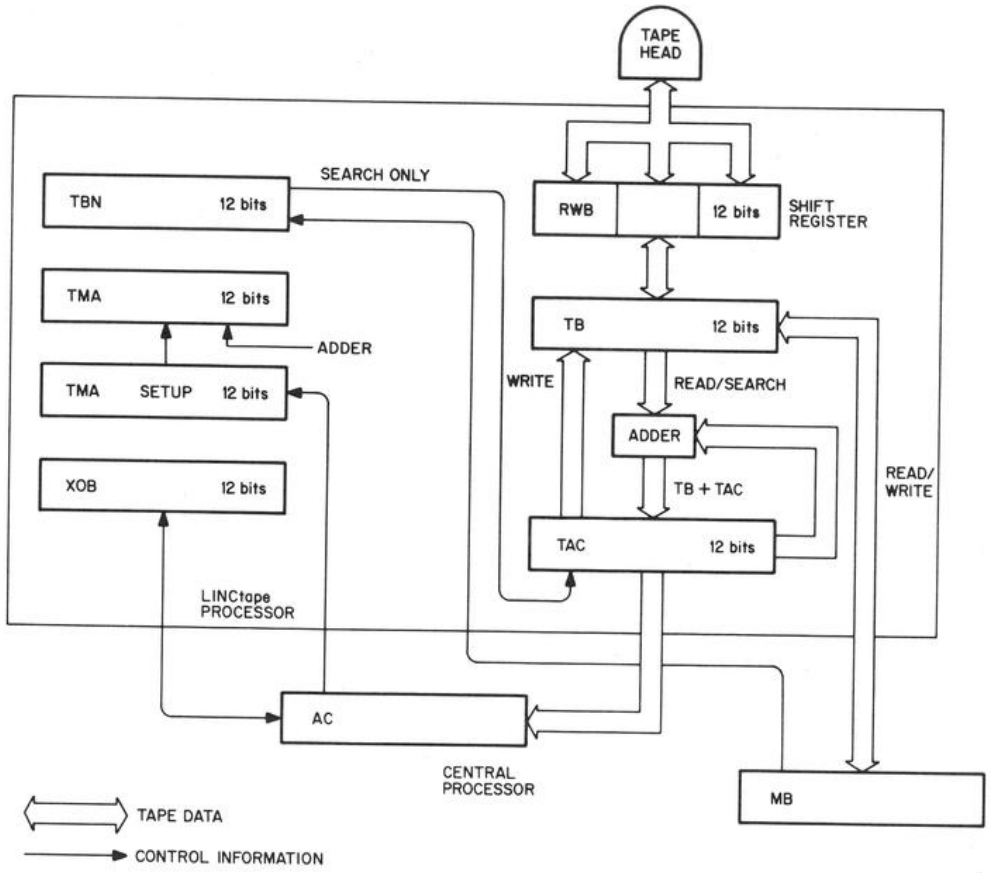


Figure 3-17. LINCtape Processor Information Paths

Tape Memory Address (TMA) 12 Bits - This contains the address of the memory location to or from which the data is being transferred. In extended address mode, TMA is loaded from the TMA Setup Register at the start of transmission; in Standard mode, the MBLK and TBLK information in the second tape instruction word are used to determine the initial contents of TMA. The TMA is incremented by 1 for each data word transferred.

TMA Setup Register 12 Bits - In Extended Address mode, this register retains the first memory address of the data to be transferred. If the transfer is not successful, the contents of TMA Setup are placed in the TMA, and the operation is repeated. The TMA Setup Register is loaded from the AC, using the TMA instruction.

Extended Operations Buffer (XOB) 12 Bits - The contents of this register determine which of the various extended tape operations are in effect. These include extended memory addressing, tape interrupt, and no-pause condition.

3.33 PROGRAMMING

The tape transfer operations are the same for both Standard and Extended Operation modes. Data can be read or written in single blocks or groups of contiguous blocks, with or without error-checking. Step-by-step searches can be performed, and block numbers can be identified without reading or writing data.

All LINCtape instructions require two words. The first word specifies the operation to be performed, one of two units, and the motion of tape at the end of the operation. The second word gives the tape block number and in Standard mode also gives the memory block number. The structure of the two words is shown in Figure 3-18.

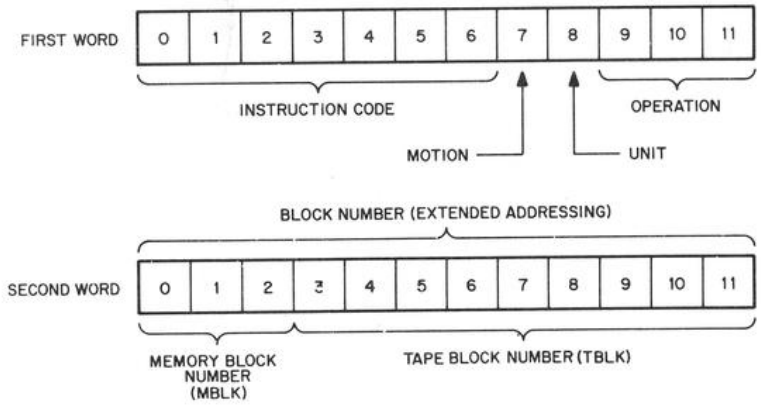


Figure 3-18. LINCtape Instruction Format

First Word

The general instruction code for the LINCtape class is 0700. The particular operation to be performed is specified by bits 9-11 of the first word. In the basic system, one of the two TU55 units is selected by bit 8. If this bit is 0, the unit dialed to 8 is selected; if bit 8 is 1, unit 1 is chosen. Bit 7 (the I-bit) is used to determine the state of the unit's motion when the operation is completed. See Section 3.34 below.

Second Word

In Standard Addressing, bits 3-11 of the second word specify the number of the tape block to be accessed. Bits 0-2 specify one of four regions of a LINC Memory Field to or from which data is to be transferred. These *memory blocks* are assigned to specific addresses in each field. There are four blocks in each field, each being 256 words (1 tape block) long. The blocks are assigned as follows:

Memory Block Number (MBLK)	*LINC Memory Field*	*LINC Field Addresses* (if IF=0 & DF=1)
0	Instruction Field	0000-0377
1	Instruction Field	0400-0777
2	Instruction Field	1000-1377
3	Instruction Field	1400-1777
4	Data Field	2000-2377
5	Data Field	2400-2777
6	Data Field	3000-3377
7	Data Field	3400-3777

In Standard Mode addressing, the contents of a tape block and a memory block correspond exactly. All single-block transfers are effected between the tape block and the memory block specified by the second word of the tape instruction. In group transfers, where several contiguous blocks are transferred, the second word is interpreted in a slightly different way. (See description of RCG and WCG instructions.) The two non-transfer instructions, MTB and CHK, use the second word in still different ways and are described below:

3.34 TAPE MOTION

Although tape can be read or written only in the forward direction, it may be searched either forward or backward. Bit 7 of the first word of a LINCtape instruction determines the motion of the tape when a LINCtape operation has been completed.

If bit 7 (the I-bit) is set to 1, the tape is left moving in the direction it was going at the completion of the operation. Except for searches backward, this will usually be the forward direction.

If bit 7 is set to 0, the tape processor enters the Turnaround state at the completion of the operation. Regardless of its former motion, the tape is set moving backwards. When a block mark is encountered, the tape stops, leaving the Turnaround state and entering the Idle state. If a new tape instruction occurs while the tape processor is in the Turnaround state, searching begins with the tape moving backwards. (A tape instruction is considered to be complete when the tape processor enters either the Turnaround or the Idle state.)

If the tape has stopped, a new tape instruction will always cause it to begin moving in the forward direction. It will not search backwards until one block number has been encountered and compared with the desired block number.

NOTE

The foregoing discussion applies only to tape motion at the completion of an instruction. It should not be confused with the NO PAUSE Extended Operation, which affects central processor action after a tape operation has begun.

3.35 LINCTAPE INSTRUCTIONS

In the subsequent instruction descriptions, the following terms are used:

Data sum	2's complement sum of all 256 data words in a block
Checksum	2's complement of (Data sum + 7777_8)
Transfer check	Data sum + Checksum + 7777_8 If the transfer is successful, the transfer check = 7777_8 If not, the transfer check $\neq 7777_8$

RDE Read Tape

Form: RDE I U
Octal code: 0702 + 20I + 10U
Execution time: 3.2 μsec
Operation: Block TBLK is read from tape into memory block MBLK. The transfer check is left in the TAC and AC. The contents of tape are unchanged.

RDC Read and Check

Form: RDC I U
Octal code: 0700 + 20I + 10U
Execution time: 3.2 μsec
Operation: Block TBLK is read from tape into memory block MBLK. If the block is transferred correctly, the transfer check is left in the TAC and AC; otherwise, the operation is repeated. The information on tape is unchanged.

RCG Read and Check Group

Form: RCG I U
Octal code: 0701 + 20I + 10U
Execution time: 3.2 μsec
Operation: MBLK is the number of additional consecutive blocks to be transferred. Block TBLK is read from tape into the memory block designated by the three low-order bits of MBLK, e.g., tape block 773 is read into memory block 3, tape block 027 into memory block 7, etc. The next consecutive TBLK blocks are read into successive memory blocks. Tape block 000 follows tape block 777, and memory block 0 follows memory block 7.
Example: Transfer blocks 202-205 from unit 1 to memory, leaving the unit in motion at the end.

```
. . . .
RCG I 1          0731
3202                          /MBLK=3, THE NUMBER OF ADDITIONAL BLOCKS
                              /TBLK=202.
```

Data is transferred from tape block 202 into memory block 2, then from 203 to memory block 3, 204 to memory block 4, and 205 to memory block 5.

Each block transfer is checked; if the transfer is successful, the transfer check (7777) is left in the TAC otherwise, that block is repeated. If the entire group is transferred successfully, 7777 is left in the TAC and AC at the end of the operation.

WRC Write and Check

Form: WRC I U
Octal Code: 704 + 20I + 10U
Execution Time: 3.2 μsec

Operation: The contents of memory block MBLK are copied into block TBLK. If the transfer is successful, the transfer check (7777) is left in the TAC; otherwise, the operation is repeated.

WRI Write Tape

Form: WRI I U
Octal code: 706 + 20I + 10U
Execution time: 3.2 μsec
Operation: The contents of memory block MBLK are copied into block TBLK; the transfer check is left in the TAC. The contents of memory are unchanged.

WCG Write and Check Group

Form: WCG I U
Octal code: 705 + 20I + 10U
Execution time: 3.2 μsec
Operation: MBLK is the number of *additional* consecutive memory blocks whose contents are written on tape. The low-order digit of TBLK specifies the first memory block to be accessed. The contents of this block are copied into block TBLK of tape. The contents of the remaining MBLK memory blocks are copied into the next successive tape blocks. Memory block 0 follows memory block 7, and tape block 000 follows tape block 777. The scheme is identical with that for RCG.

The following two instructions do not transfer any data.

MTB Move Toward Block

Form: MTB I U
Octal code: 703 + 20I + 10U
Execution time: 3.2 μsec
Operation: Subtract the next tape block number (or reverse block number, if the tape is moving backwards) encountered from TBLK, leaving the difference in the TAC and AC. If I = 0, the tape stops. If I = 1, the tape is left moving forward if the difference is positive or 0, and backward if negative. If the tape is at rest when this instruction is given, it starts moving forward; otherwise it continues in the direction it had been going. The MBLK bits of the second word are ignored.

CHK Check Tape Block

Form: CHK I U
Octal code: 707 + 20I + 10U
Execution time: 3.2 μsec
Operation: Tape block TBLK is found, and its contents read into the tape control registers only, to form the data sum (no transfer takes place). The checksum is read, and the transfer check is left in the TAC and AC. The information on tape is unchanged, and the MBLK bits of the second word are ignored.

The contents of the Tape Accumulator can be examined by using the following instruction.

TAC Tape Accumulator to AC

Octal code: 0003
Execution time: 1.6 μsec
Operation: The contents of the Tape Accumulator are placed in the central processor AC. The previous C(AC) are lost; C(TAC) are unchanged.

3.36 EXTENDED OPERATIONS

LINCtape Extended Operations give the programmer a more flexible addressing scheme for information transfers, additional control functions, and a tape processor maintenance facility. These operations are controlled by the contents of the Extended Operations Buffer, defined as shown in Figure 3-19. The XOB can be loaded from the AC and vice versa.

AXO AC to XOB

Octal code: 0001
Execution time: 1.6 μsec
Operation: The contents of the AC are placed in the Extended Operations Buffer. The previous C(XOB) are lost; C(AC) are unchanged.

XOA XOB to AC

Octal code: 0021
Execution time: 1.6 μsec
Operation: The contents of the Extended Operations Buffer are placed in the AC. The previous C(AC) are lost; C(XOB) are unchanged.

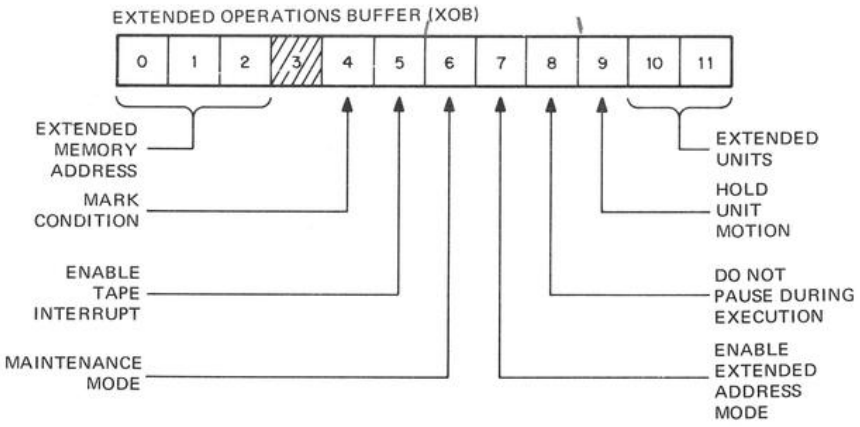


Figure 3-19. Extended Operations Buffer Bit Assignments

3.36.1 Extended Address Format

This facility releases the programmer from the limitation of block to block transfers, as described previously. Instead, a block transfer may begin in any register of any memory bank, regardless of the settings of the Memory Field Registers.

When the Extended Address Mode is enabled (by setting bit 7 of the XOB to 1), all subsequent tape transfers are executed as follows:

The starting address of the area in memory is placed in the TMA Setup Register by the program, using the instruction TMA.

TMA Load TMA Setup Register

Octal code: 0023
Execution time: 1.6 μsec
Operation: The contents of the AC are placed in the Tape Memory Address Setup Register. The previous contents of TMA Setup are lost; C(AC) are unchanged.

At the occurrence of a tape transfer instruction, the contents of the TMA Setup Register are placed in the TMA. The second word of the instruction is taken as a full 12-bit block number, and placed in the TBN. The transfer is effected between tape and the designated area of the 4096-word memory bank specified by bits 0-2 of the XOB. The transfer is thus independent of the LINC Memory Field assignments.

As in all extended memory operations, whether with tape or not, the transfer will not cross memory bank boundaries; register 7777 of a given bank is followed by register 0000.

NOTE

The group transfer instructions RCG and WCG cannot be used in extended address mode.

The non-transfer instructions MTB and CHK are not affected.

Example: Read the contents of block 365 of unit 0 into memory bank 1 (second 4K memory bank) starting in register (10)540.

Instruction	*Octal Code*	*Action*
....		
LDA I 0	1020	/LOAD AC WITH STARTING ADDRESS
0540	0540	/STARTING ADDRESS OF TARGET AREA
TMA	0023	/PLACE STARTING ADDRESS IN TMA SETUP
LDA I 0	1020	/LOAD AC WITH EXTENDED OPERATIONS BITS
1020		/BANK 1; ENTER EXTENDED ADDRESS MODE.
AXO	0001	/LOAD XOB FROM AC.
....	....	
....		
RDC 0	0700	/READ AND CHECK FROM UNIT 0
0365	0365	/BLOCK 365
....	....	

The data will be read into registers 540-1137 of memory bank 1.

3.36.2 Extended Units

The two Extended Units bits (XOB_{10-11}) may be thought of as an extension of the unit bit of a LINCtape instruction (bit 8). Taken together, the three bits can select one of up to eight TU55 transports which may be attached to the TC12 tape control. The logical unit numbers are assigned by rotating the dials on the transports; they correspond to the unit select bits as follows:

Extended Unit Bits (XOB) 10	11	*Instruction Unit Bit* 8	*Transport Selected*
0	0	0	8
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

3.36.3 Tape Interrupt Enable

When this bit (XOB_5) is set, a program interrupt will occur whenever a tape operation is not in progress. As with other LINC interrupts, control is transferred to register 0041 of memory field 0; the contents of the PC are stored in register 0040. (If the central processor is in PDP-8 mode, the interrupt uses registers 0001 and 0000.)

3.36.4 No Pause Condition

Normally, the central processor waits until a tape operation is finished before proceeding. Such delays are eliminated by setting the No Pause condition bit (XOB_8). When this condition is enabled, the processor continues with the program as soon as the LINCtape instruction has been interpreted and the operation initiated.

Subsequently, the program can monitor the Tape Done (STD) flag to determine when the operation has finished, before starting a new one. When NO PAUSE is set, the transfer of the Transfer Check to the accumulator at the end of tape instruction inhibited.

STD Skip if Tape Done

Form: STD I
Octal code: 0416 + 20 I
Execution time: 1.6 μsec
Operation: If I = 0, skip the next instruction if the tape operation is completed; otherwise execute the next instruction. If I = 1, skip if the operation is still in progress. This instruction is identical to SXL I 16.

Used in conjunction with the tape flag and tape interrupt, the No Pause condition can save considerable amounts of time, and gives the programmer added flexibility in the processing of data before a transfer is completed.

3.36.5 Hold Unit Motion

Normally, a tape transport stops as soon as another unit has been selected. When XOB_9 is set, however, the transport will continue in the direction it has been moving when the unit is deselected. This is a useful feature for certain operations involving several units, and must be used with care. Note that it is not the same as the motion bit of a LINCtape instruction, which determines the motion state of a unit at the completion of an instruction only.

3.36.6 MARK Condition

This bit (XOB_4) is used in conjunction with the MARK switch on the operator's console, to allow the MARK12 program (see Chapter 6, Section III on Program Library) to record Timing and Mark tracks on a new tape. The interaction between the switch and the XOB is designed to minimize the possibility of accidentally destroying a tape by enabling the MARK flip-flop. The flip-flop can be set only when the MARK switch is held down *while* an AXO instruction is being executed with AC_4 set to 1.

3.36.7 Maintenance Mode

When bit 6 of the XOB is set to 1, all timing signals and data are prevented from entering the tape control registers from the reader-writers. Instead, signals generated by PDP-8 IOT instructions are used as input to the tape control, in order to simulate the functions of the tape head and the tape processor. The Maintenance Mode is designed for diagnostic purposes only and is not intended for general use.

3.36.8 Tape Trap

Whenever the TAPE TRAP and INSTRUCTION TRAP Special Functions are enabled (ESF with AC_{2-3} set to 1s), LINCtape instructions are not executed. When one is encountered, a program trap to register 0140 of memory field 0 occurs. The Tape Trap is intended primarily for use with LINC-8 programs and the I/O Handler (PROGOFOP simulator) to ensure compatibility.

CHAPTER 4
PDP-8 MODE PROGRAMMING

This chapter covers the PDP-12 programming in the PDP-8 mode. The contents of this chapter are divided into six sections: Organization of Memory, Memory Addressing Methods, PDP-8 Instructions, Program Interrupt, Extended Arithmetic Element, and Extended Memory.

Section I. ORGANIZATION OF MEMORY

4.1 ORGANIZATION

In PDP-8 mode, the basic 4096-word memory is divided into 32 segments of 128 words each for addressing purposes. Within one of these *pages*, operands may be addressed directly by memory reference instructions. Access to operands across page boundaries (except for Page 0) requires indirect addressing.

Executable programs may be stored in any page of memory, and program sequences may extend across several pages. The program counter is indexed over all 12 bits in PDP-8 mode, so that a straight-line program sequence will pass from the last word of a page to the first word of the next. A programmed jump across page boundaries, however, requires an indirect reference.

The organization of one memory bank in PDP-8 mode is shown in Figure 4-1.

4.2 PAGE 0

The first page of memory (addresses 000-177) contains several registers reserved for special use, which the programmer must take into account. These are:

Address	*Use*
0000	During a program interrupt, holds C(PC).
0001	Contains the first instruction to be executed after a program interrupt.
0010-0017	Automatic index registers (see paragraph 4.6).

PAGE (OCTAL)	ADDRESSES
0	0000—0177
1	0200—0377
2	0400—0577
3	0600—0777
4	1000—1177
5	1200—1377
6	1400—1577
7	1600—1777
10	2000—2177
11	2200—2377
12	2400—2577
13	2600—2777
14	3000—3177
15	3200—3377
16	3400—3577
17	3600—3777
20	4000—4177
21	4200—4377
22	4400—4577
23	4600—4777
24	5000—5177
25	5200—5377
26	5400—5577
27	5600—5777
30	6000—6177
31	6200—6377
32	6400—6577
33	6600—6777
34	7000—7177
35	7200—7377
36	7400—7577
37	7600—7777

Figure 4-1. Organization of Memory, PDP-8 Mode

4.3 EXTENDED MEMORY

Additional 4K memory banks are organized in the same manner as is the basic bank. The Memory Field registers determine the assignment of banks (see Section VI).

Section II. MEMORY ADDRESSING METHODS

4.4 DIRECT ADDRESSING

In PDP-8 mode, all memory reference instructions have the same structure, shown in Figure 4-2. Note that only seven bits (5-11) are available for use as an address. This is just sufficient to give access to 128 registers, or exactly one page. The state of bit 4 of the instruction determines which of two possible pages the 7-bit *page address*

references. If this bit is 1, the page address is on the *current page,* that is, the one in which the instruction itself is stored. If bit 4 is 0, the page address is on Page 0. Thus, a memory reference instruction has direct access to a total of 256 registers of memory; the 128 locations of Page 0, and those of the current page.

Examples:

To store the contents of the AC in register 150 of the current page:

DCA 350	Octal code: 3150. The page address is 150; bit 4 set to 1 gives a total octal value of 350 for the address

To store the contents of the AC in register 150 of Page 0:

DCA 150	Octal code: 3150. With the page bits set to 0, the total octal address is 150.

As one can see from these examples, it is useful to think of page addresses running from 000-177 on Page 0, and from 200-377 on the current page.

4.5 INDIRECT ADDRESSING

To gain access to registers outside of Page 0 or the current page, indirect addressing must be used. If bit 3 of a memory reference instruction is set to 1 (see Figure 4-2), the contents of the register designated by bits 4-11 are taken as the effective address of the operand. This is a full 12-bit number which gives the absolute address of any register in the 4K memory bank.

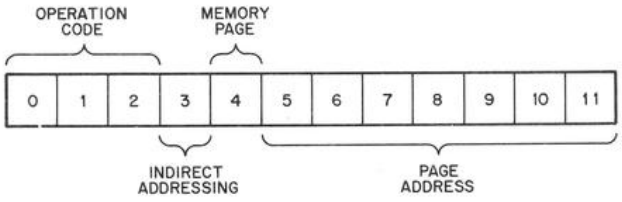


Figure 4-2. Memory Reference Instruction Format

In the following examples, as in normal PDP-8 programming, the letter *I* is used as a mnemonic to represent the presence of a 1 in bit 3.

Examples:

1. To store the contents of the AC in register 100 of Page 10 (absolute address 2100), using an effective address stored on the current page:

Absolute Address	*Contents*	*Action*
0410	DCA I 300	/OCTAL CODE: 3700. THE /EFFECTIVE ADDRESS IS
. . . .		
0500	2100	/CONTAINED IN REGISTER 500, /(PAGE ADDRESS 300)

2. To store the C(AC) in register 2100, using an effective address stored in Page 0:

Absolute Address	*Contents*	*Action*
0050	2100	/EFFECTIVE ADDRESS, STORED
....		/ON PAGE 0
0410	DCA I 50	/OCTAL CODE: 3450. (BIT 4 = 0)

TABLE 4-1. SUMMARY OF ADDRESSING METHODS IN PDP-8 MODE

Bit 3	Bit 4	Effective Address
0	0	The operand is in Page 0 at the address specified by bits 5 through 11.
0	1	The operand is in the current page at the address specified by bits 5 through 11.
1	0	The absolute address of the operand is taken from the contents of the location in Page 0 designated by bits 5 through 11.
1	1	The absolute address of the operand is taken from the contents of the location in the current page designated by bits 5 through 11.

4.6 AUTOINDEXING

The eight registers in locations 10-17 of Page 0 have a special function when indirectly addressed. The contents of such a register are first incremented by 1, and the result is taken as the effective address of the operand. This *autoindexing* feature allows the programmer to address a series of contiguous locations without extra address modification, as shown in the following example.

Example:

To obtain the sum of 100 numbers stored in registers 1000-1077.

```
Address
Label     Instruction     Operation
GO,       CLA             /CLEAR THE AC
          TAD LIST        /PUT 777 IN AC (ADDRESS-1 OF THE TABLE OF NUMBERS
          DCA 10          /DEPOSIT IN AUTOINDEX REGISTER 10. (CLEARS AC)
          TAD COUNT       /PUT -100 IN AC (COUNT OF ADDENDS IN TABLE)
          DCA INDEX       /DEPOSIT IN REGISTER FOR COUNTING
LOOP,     TAD I 10        /C(10) INCREMENTED BY 1, THEN USED AS
                          /EFFECTIVE ADDRESS TO GET ADDEND FROM TABLE
          ISZ INDEX       /INCREMENT COUNT. IF RESULT IS 0000, SKIP
                          /THE NEXT INSTRUCTION.
          JMP LOOP        /IF NOT FINISHED, GO BACK TO GET NEXT ADDEND
```

Address Label	*Instruction*	*Operation*
END	HLT	/WHEN FINISHED, STOP; AC CONTAINS THE SUM
. . . .		
LIST	777	/ADDRESS-1 OF TABLE OF ADDENDS
COUNT	-100	/COUNT OF TABLE ENTRIES
INDEX	0000	/HOLDS COUNT DURING EXECUTION OF PROGRAM

When register 10 is first accessed, its contents are incremented from 777 to 1000, then used as the effective address to obtain the first addend. The next time around the loop, the C(10) are again incremented by 1, to 1001, for the next operand. At the end of the sequence, C(10)=1077.

Section III. PDP-8 INSTRUCTIONS

4.7 MEMORY REFERENCE INSTRUCTIONS

There are six memory reference instructions: DCA, TAD, AND, ISZ, JMP, and JMS. All may use either direct or indirect addressing. When indirect addressing is specified, add 1.6 microseconds to the execution time.

DCA Deposit and Clear Accumulator

Form: DCA Y
Octal code: 3000 + Y
Execution time: 3.2 μsec
Operation: The contents of the AC are deposited in register Y; the AC is then cleared to 0000. The previous C(Y) are lost.

TAD 2's Complement Add To Accumulator

Form: TAD Y
Octal code: 1000 + Y
Execution time: 3.2 μsec
Operation: The contents of Y are added to the contents of the AC, using 2's complement addition. If there is a carry out of bit 0, the Link is complemented; otherwise, the Link is unchanged. The previous contents of the AC are lost; the contents of Y are not changed.

AND Logical AND To Accumulator

Form: AND Y
Octal code: 0000 + Y
Execution time: 3.2 μsec
Operation: The contents of the AC and the contents of Y are combined according to the Boolean AND relation, with the result left in the AC. The comparison is made on each bit pair independent of the other bits in the two operands. The truth table for the AND relation is shown below:

		$C(AC_j)$	
		0	1
$C(Y_j)$	0	0	0
	1	0	1

When corresponding bits of AC and Y are both 1, the result is 1. Otherwise, the result is 0. The previous C(AC) are lost; the C(Y) are unchanged.

ISZ Increment And Skip If Zero

Form: ISZ Y
Octal code: 2000 + Y
Execution time: 3.2 μsec
Operation: The contents of Y are incremented by 1. If the result is 0000, the next instruction in sequence is skipped; otherwise, the next instruction is executed. The contents of the AC are not affected.

JMP Jump

Form: JMP Y
Octal code: 5000 + Y
Execution time: 1.6 μsec
Operation: The address Y is placed in the PC, and the next instruction is taken from register Y; the program continues from that point. The contents of the AC are not affected.

JMS Jump To Subroutine

Form: JMS Y
Octal code: 4000 + Y
Execution time: 3.2 μsec
Operation: The contents of the PC are stored in Y. The address Y + 1 is placed in the PC, and the program continues from Y + 1. The contents of the AC are not affected. To return from the subroutine to the point at which the JMS was given (i. e., to the register immediately following the JMS), the instruction *JMP I Y* is executed. The contents of Y are taken as the effective address; since Y contains the PC stored at the time of the JMS, control returns to the calling program.

4.8 OPERATE CLASS INSTRUCTIONS

This class is divided into two groups, I and II. Group I instructions include miscellaneous operations on the Accumulator and Link. Group II instructions include skips, the program halt, and access to the console switches.

Operate class instructions are microprogrammed; they may be combined to provide several operations within a single instruction. However, combinations can be made only within a group; operations from different groups cannot be combined. To ease this restriction, the operation CLA (Clear the AC) is available in both groups. All Operate Class instructions require 1.6 microseconds for execution.

4.8.1 Operate Class: Group I

The microprogram structure of Group I instructions is shown in Figure 4-3. Any combination of these functions can be made, but the programmer must be aware of the order in which the operations are performed when the instruction is executed. This order is as follows:

1. CLA, CLL
2. CMA, CML
3. IAC
4. RAR, RAL, RTR, RTL

Certain combinations of Group I operations are common enough to be assigned separate mnemonics. These are described in paragraph 4.8.2.

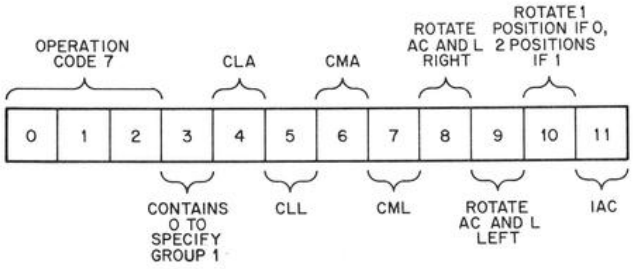


Figure 4-3. Group I Operate Class Instruction Format

NOP No Operation

Octal code: 7000
Operation: None. This instruction may be used to provide short delays (1.6 microsecond per instruction) or to hold a place for instructions to be inserted by the program.

CLA Clear Accumulator

Octal code: 7200
Operation: The contents of the AC are cleared to 0000.

CLL Clear Link

Octal code: 7100
Operation: The contents of the Link are cleared to 0.

CMA Complement Accumulator

Octal code: 7040
Operation: The 1's complement of the contents of the AC replace the original contents of the AC. Each bit that is 0 becomes 1, and vice versa.

CML Complement Link

Octal code: 7020
Operation: The contents of the Link are complemented.

IAC Increment Accumulator

Octal code: 7001
Operation: The contents of the AC are incremented by 1, using 2's-complement arithmetic. A carry out of bit 0 is lost. The Link is unaffected.

RAR Rotate Accumulator Right

Octal code: 7010
Operation: The contents of the AC and Link are rotated right one position. A digit rotated out of AC_{11} enters the Link; the bit rotated out of the Link enters AC_0. (See Figure 4-4.)

RTR Rotate Two Places Right

Octal code: 7012
Operation: The contents of the AC and Link, taken as a 13-bit register, are rotated two positions to the right. (See Figure 4-4.)

RAL Rotate Accumulator Left

Octal code: 7004
Operation: The contents of the AC and Link, taken as a 13-bit register, are rotated one place to the left. A bit leaving AC_0 enters the Link; a bit leaving the Link enters AC_{11}. (See Figure 4-4.)

RTL Rotate Two Places Left

Octal code: 7006
Operation: The contents of the AC and Link, taken as a 13-bit register, are rotated two positions left. (See Figure 4-4.)

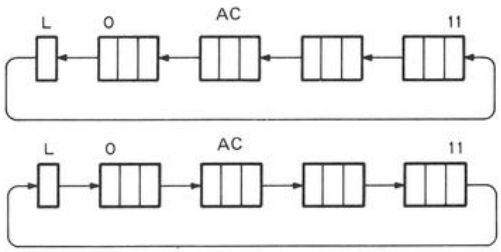


Figure 4-4. Rotation Scheme for RAR, RTR, RAL, RTL

4.8.2 Combined Operations: Group I

The following combined operations have been given separate mnemonics for programming convenience.

STA Set Accumulator (CLA + CMA)

Octal code: 7240
Operation: Clear, then complement the AC. Resulting C(AC) = 7777.

STL Set Link (CLL + CML)

Octal code: 7120
Operation: Clear, then complement the Link. Resulting C(L) = 1.

CIA Complement And Increment Accumulator (CMA + IAC)

Octal code: 7041
Operation: Complement the AC, then increment the result by 1. This gives the 2's complement of the original C(AC). The 2's complement of a number is defined as the 1's complement plus 1.

GLK Get Link (CLA + RAL)

Octal code: 7204
Operation: Clear the AC, then rotate one place left, thus putting the contents of the Link into AC_{11}. This instruction is useful in multiple precision arithmetic.

Other Useful Combinations – The programmer can place a number of selected constants in the AC by combining Group I operations, as shown:

Combination				*Resulting C(AC)*
CLA	IAC			0001
CLA	STL	RTL		0002
CLA	STL	IAC	RAL	0003
CLA	CLL	IAC	RTL	0004
CLA	STL	IAC	RTL	0006
STA	CLL	RAL		7776 (-2)
STA	CLL	RTL		7775 (-3)
CLA	STL	RAR		4000
CLA	STL	RTR		2000
CLA	IAC	STL	RTR	6000

4.8.3 Operate Class: Group II

The microprogram structure of Group II operations is shown in Figure 4-5. Any of these operations may be combined, but again, the programmer must be aware of the sequence of events. In addition, the sense of the skip instruction determines the manner in which combined skips are interpreted. If bit 8 is 0, the logical OR of the tested conditions will cause a skip; if the bit is 1, the logical AND of the conditions will cause the skip. In the first case, this means that the skip will occur if any one of the conditions tested is true; in the second case, the skip will occur only if all the conditions tested are true. The various combinations are described in paragraph 4.8.4.

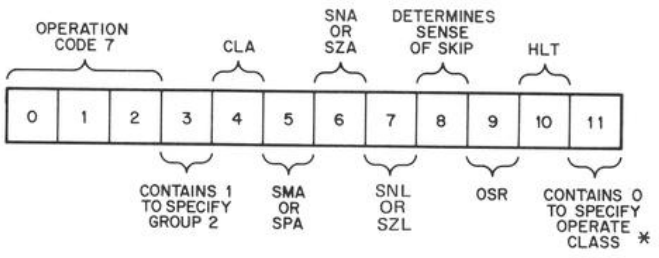


Figure 4-5. Group II Operate Class Instruction Format

The sequence of events in a Group II instruction is as follows:

1. Skips
2. CLA
3. OSR

HLT occurs after all other specified operations have been performed.

CLA Clear AC

Octal code: 7600
Operation: Clear the AC

SKP Skip Unconditionally

Octal code: 7410
Operation: The next instruction in the program sequence is skipped.

SNL Skip On Non-Zero Link

Octal : 7420
Skip Condition: The contents of the Link equal 1.

SZL Skip on Zero Link

Octal code: 7430
Skip condition: The contents of the Link equal 0.

SZA Skip On Zero Accumulator

Octal code: 7440
Skip condition: The contents of the AC equal 0000.

SNA Skip On Non-Zero Accumulator

Octal code: 7450
Skip condition: The contents of the AC are not equal to 0000.

SMA Skip On Minus Accumulator

Octal code: 7500
Skip condition: The contents of AC_0 equal 1. By convention, a negative number is one in which the most significant digit is 1. Thus, all numbers between 4000 and 7777, inclusive, are negative. The 2's complement of such a number is its positive counterpart. In this sense, 7777 is equivalent to -1; 4000 is equivalent to -4000. The 2's complement sum of a number and its 2's complement is always zero.

SPA Skip On Plus Accumulator

Octal code: 7510
Skip condition: The contents of AC_0 equal 0. By the convention described above, a number is positive if its most significant digit is 0.

OSR OR Switch Register With Accumulator

Octal code: 7404
Operation: The contents of the console switch register (Right Switches) are combined with the contents of the AC by the logical Inclusive OR relation; the result is left in the AC. If either bit of a corresponding pair is set to 1, the result is 1. The result is 0 only if both AC and SR bits are 0. This instruction is normally used with CLA to obtain the actual status of the Switches (see below).

HLT Halt

Octal code: 7402
Operation: The processor stops. The PC contains the address of the register following the HLT instruction. The contents of the other processor registers are not affected.

LAS Load Accumulator From Switches (CLA + OSR)

Octal code: 7604
Operation: Clear the AC, then OR the contents of the Right Switches with C(AC). This places the status of the switches in the AC. If the switch is set to 1, the corresponding AC bit is set to 1.

4.8.4 Combined Skips In Group II

The possible skip combinations are listed, with the conditions for a skip to occur.

Combination	*Octal Code*	*A skip will occur if*
SZA SNL	7460	C(AC) = 0000, or C(L) = 1, or both
SZA SMA	7540	C(AC) = 0000, or $C(AC_0) = 1$, or both
SMA SNL	7520	$C(AC_0) = 1$, or C(L) = 1, or both
SZA SMA SNL	7560	C(AC) = 0000, or $C(AC_0) = 1$, or C(L) = 1, or any, or all of these.
SNA SZL	7470	$C(AC) \neq 0$ and C(L) = 0
SNA SPA	7550	$C(AC) \neq 0$ and $C(AC_0) = 0$
SPA SZL	7530	$C(AC_0) = 0$ and C(L) = 0
SNA SPA SZL	7570	$C(AC) \neq 0000$ and $C(AC_0) = 0$ and C(L) = 0

If CLA is combined with any skip, the AC is cleared after the conditions have been tested.

4.8.5 Input/Output Transfer Class

These instructions, all of which have the basic operation code of 6000, are used to service all peripheral devices, enable and disable the program interrupt and the memory extension control, change from PDP to LINC programming mode, and provide maintenance operations for the LINCtape subprocessor. Most of these instructions are described in Chapter 5 with their associated devices. The program interrupt and memory extension control are discussed in Sections IV and VI of this chapter.

Mode Control – To change operating mode from PDP-8 to LINC, the following IOT instruction is used.

LINC Switch To LINC Mode

Octal code: 6141
Execution time: 4.25 μsec
Operation: Starting with the next succeeding instruction, the central processor will operate in LINC mode.

Section IV. PROGRAM INTERRUPT

4.9 OPERATION

To facilitate the handling of data transfers and the checking of peripheral device status, provision is made for interrupting a program when a given condition exists. In general, an interrupt occurs when a peripheral device flag is raised, i. e., when the device is available for service, when an operation has been completed, or when a specific condition, such as an alarm, occurs within the device.

The Program Interrupt is enabled or disabled by the program. When it is disabled, a device flag must be sensed by means of a skip; the program is not interrupted. When the interrupt is enabled, any device flag that is connected to the interrupt system will cause the following sequence of events to occur when the flag is raised:

1. The instruction in progress at the time of the interrupt request is completed.

2. The contents of the program counter are stored in register 0000, and 0001 is placed in the PC.

3. Processing continues beginning with the instruction in register 0001.

4. The interrupt facility is disabled.

The two IOT instructions which control the interrupt facility are described below.

ION Interrupt On

Octal code: 6001
Execution time: 4.25 μsec
Operation: The interrupt facility is enabled immediately after the instruction *following* the ION has been executed. If an interrupt request is waiting at the time of the ION, the interrupt will occur after the next instruction has been completed. The enabling is delayed in this manner so that an interrupt service routine can return to the interrupted program before a subsequent interrupt request destroys the contents of register 0000.

IOF Interrupt Off

Octal code: 6002
Execution time: 4.25 μsec
Operation: The interrupt facility is disabled. Subsequent requests will not cause an interrupt, although the flag causing the request may be sensed in the usual manner with an IOT skip.

4.10 USING THE INTERRUPT

Normally, when an interrupt occurs, the instruction in register 0001 is a JMP to an interrupt service routine, which examines the expected flags to determine which device or condition caused the request. The appropriate routine is called to service the device. During this time, the interrupt facility is disabled. When the device service routine is completed, control normally returns to the interrupt handling routine for restoring the interrupt facility and exiting to the main program. The last two instructions of such a routine would be:

```
ION          /ENABLE INTERRUPT FACILITY
JMP I 0      /RETURN TO MAIN PROGRAM AT THE ADDRESS
             /STORE 0 IN REGISTER 0000
```

The interrupt is not enabled until the JMP I 0 has been executed, so that the return to the main program is completed before a waiting request can cause another interrupt.

Section V. EXTENDED ARITHMETIC ELEMENT

4.11 OPERATION

The Extended Arithmetic Element (EAE), Type KE12, adds a complete automatic multiplication and division facility to the PDP-12. Programming is by means of PDP-8 mode instructions. The AC and MQ are used to accommodate full 24-bit products and dividends, and the remainder and quotient after a division. Shifting, normalizing, and register setup instructions are included.

All operands are treated as unsigned integers; the programmer must establish his own sign conventions. The normalizing instruction facilitates the writing of floating-point subroutines.

4.12 EAE INSTRUCTIONS

The EAE instruction set has a basic operation code of 7401; all functions are microprogrammed, as in Operate Class commands. The microprogram structure of the EAE class is shown in Figure 4-6.

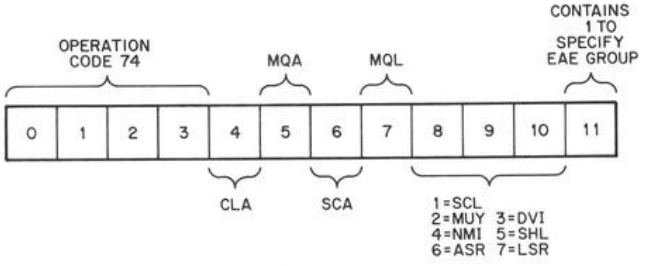


Figure 4-6. EAE Instruction Format

As with other microprogrammed instructions, EAE operations are performed in a given order. Operations can be combined in a single instruction, *except* that operations occurring at the same time cannot be combined with meaningful results.

The order of events is as follows:

1. CLA
2. MQA, MQL, SCA
3. SCL, MUY, DVI, NMI, SHL, ASR, LSR

CLA Clear AC

Octal code: 7601
Execution time: 1.6 μsec
Operation: Clear the AC. The MQ and Link are unaffected.

MQA Place MQ In AC

Octal code: 7501
Execution time: 1.6 μsec
Operation: The contents of the MQ are ORed into the AC. The C(MQ) are unchanged.

NOTE

All twelve bits are transferred; this is *not* identical to the LINC mode QAC instruction (page 3-21).

MQL Load MQ from AC

Octal code: 7421
Execution time: 1.6 μsec
Operation: The contents of the AC are placed in the MQ. The previous (CMQ) are lost. The AC is cleared at the end of the instruction.

SCA SCA Step Counter to AC

Octal code: 7441
Execution time: 1.6 μsec
Operation: The contents of the step counter are ORed with the contents of bits 7-11 of the AC; the result is left in the AC. To obtain the actual step count, this can be combined with CLA (combined operation code: 7641).

All the rest of the EAE instructions except NMI require two words; the first contains the operation to be performed, the second contains the operand. In the following descriptions, the notation p+1 designates the register containing the operand.

SCL Load Step Counter

Octal code: 7403
Execution time: 3.2 μsec
Operation: The *complement* of the contents of bits 7-11 of p+1 is placed in the SC.

MUY Multiply

Octal code: 7405
Execution time: 8.0 μsec
Operation: The number in the MQ is multiplied by the number in register p+1. At the conclusion of this command the Link contains a 0. The most significant 12 bits of the product are in the AC and the least significant 12 bits are in the MQ.

DVI Divide

Octal code: 7407
Execution time: 4.0 μsec to 9.0 μsec
Operation: The 24-bit dividend is held in the AC (most significant part) and MQ (least significant part); the divisor is in register p+1. At the conclusion of the division, the quotient is in the MQ, and the remainder in the AC. If the division was carried out, the Link is left clear. If either dividend or divisor is zero, the operation ends after one step, and the Link is set to 1, to indicate that a *divide overflow* occurred.

NMI Normalize

Octal code: 7411
Execution time: 1.6 μsec to 0.33 μsec/step
Operation: The AC and MQ are treated as a single 24-bit register. The combined contents of the AC and MQ are shifted left until $C(AC_0)$ differs from $C(AC_1)$.Bits shifted out of AC_0 enter the Link; bits shifted out of the Link are lost. Zeros enter MQ_{11} and are shifted up the registers. At the end of the operation, the SC contains the number of shifts performed, this being the exponent of the normalized floating point fraction. The shift path is shown in Figure 4-7.

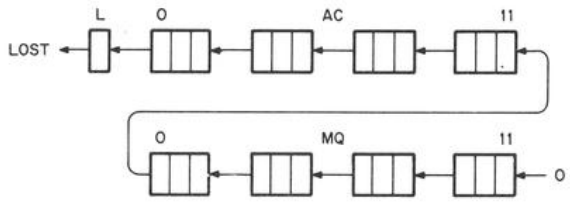


Figure 4-7. Shift Path for NMI, SHL

SHL Shift Left (Arithmetic)

Octal code: 7431
Execution time: 3.2 μsec + 0.33 μsec/step
Operation: The combined contents of the Link, AC and MQ are shifted left N+1 places, where N is the number contained in bits 7-11 of register p+1. Bits shifted out of the Link are lost; zeros are shifted into MQ_{11} and up the registers. The shift path is identical with that for NMI, as shown in Figure 4-8.

ARS Shift Right (Arithmetic)

Octal code: 7415
Execution time: 3.2 μsec + 0.33 μsec/step
Operation: The combined contents of the AC and MQ are shifted right N+1 places, where N is the number contained in bits 7-11 of register p+1. The sign bit (AC_0) is replicated in all vacated bit positions, and is also placed in the Link. Bits shifted out of MQ_{11} are lost, as is the previous C(L). The shift path is shown in Figure 4-8.

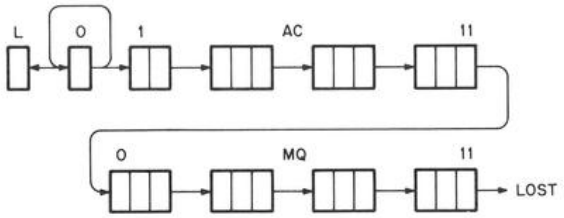


Figure 4-8. Shift Path for ASR

LSR Shift Right Logical

Octal code: 7421
Execution time: 3.2 μsec + 0.33 μsec/step
Operation: The contents of the AC are placed in the MQ. The previous C(MQ) are lost. The AC is cleared at the end of the instruction.

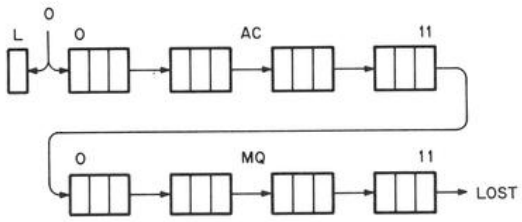


Figure 4-9. Shift Path for LSR

4.13 EAE PROGRAMMING

4.13.1 Multiplication

Multiplication is performed as follows:

1. Load the AC with the multiplier using the TAD instruction.
2. Transfer the contents of the AC into the MQ using the MQL command.
3. Give the MUY command.

Note that steps 2 and 3 can be combined into one instruction.

The contents of the MQ are then multiplied by the contents of the next successive core memory address (p+1). At the conclusion of the multiplication the most significant 12 bits of the product are held in the AC and the least significant bits are held in the MQ. This operation takes a maximum of 8.0 microseconds; at the end of this time the next instruction is executed.

The following program examples demonstrate the operation of the EAE multiplication:

Multiplication of 12-Bit Unsigned Numbers

Enter with a 12-bit multiplicand in AC and a 12-bit multiplier in core memory. Exit with high order half of product in a core memory location labelled HIGH, and with low order half of product in the AC. Program time is approximately 13 microseconds.

```
        MQL MUY         /LOAD MQ WITH MULTIPLICAND, INITIATE
                        /MULTIPLICATION
        MLTPLR          /MULTIPLIER
        DCA HIGH        /STORE HIGH ORDER PRODUCT
        MQA             /LOAD AC WITH LOW ORDER PRODUCT
```

Multiplication of 12-Bit Signed Numbers, 24-Bit Signed Product

Enter with a 12-bit multiplicand in AC and a 12-bit multiplier in core memory. Exit with signed 24-bit product in core memory locations designated HIGH LOW.

```
                CLL
                SPA                     /MULTIPLICAND POSITIVE?
                CMA  CML    IAC         /NO. FORM TWO'S COMPLEMENT
                MQL                     /LOAD MULTIPLICAND INTO MQ
                TAD  MLTPLR
                SPA                     /MULTIPLIER POSITIVE?
                CMA  CML    IAC         /NO. FORM TWO'S COMPLEMENT
                DCA  MLTPLR
                RAL
                DCA  SIGN               /SAVE LINK AS SIGN INDICATOR
                MUY                     /MULTIPLY
MLTPLR,         0                       /MULTIPLIER
                DCA  HIGH
                TAD  SIGN
                RAR                     /LOAD LINK WITH SIGN INDICATOR
                MQA
                SNL                     /IS PRODUCT NEGATIVE?
                JMP          LAST       /NO
                CLL  CMA     IAC        /YES
                DCA          LOW
                TAD          HIGH
                CMA
                SZL
                IAC
                DCA          HIGH
                SKP
LAST,           DCA          LOW
```

4.13.2 **Division**

Division is performed as follows:

1. Load the 12 least significant bits of the dividend into the AC using the TAD instruction, then transfer the contents of the AC into the MQ using the MQL command.

2. Load the 12 most significant bits of the dividend into the AC.
3. Give the DVI command.

The 24-bit dividend contained in the AC and MQ is divided by the 12-bit divisor contained in the next successive core memory address (p+1). This operation takes a maximum of 7.8 microseconds and is concluded with a 12-bit quotient held in the MQ, the 12-bit remainder in the AC, and the Link holding a 0 if divide overflow did not occur. To prevent divide overflow, the divisor in the core memory must be greater than the 12 bits of the dividend held in the AC. When divide overflow occurs, the Link is set and the division is concluded after only one cycle. Therefore the instruction following the divisor in core memory should be an SZL microinstruction to test for overflow. The instruction following the SZL may be a jump to a subroutine that services the overflow. This subroutine may cause the program to type out an error indication, rescale the divisor or the dividend, or perform other mathematical corrections and repeat the divide routine.

The following program examples demonstrate the use of the EAE in division.

Division of 12-Bit Unsigned Numbers

Enter with a 12-bit unsigned dividend in the AC and a 12-bit unsigned divisor in core memory. Exit with remainder in core memory location labeled REMAIN and with the quotient in the AC.

```
CLL
MQL DVI                 /LOAD MQ, INITIATE DIVISION
DIVISOR                 /DIVISOR
SZL                     /OVERFLOW?
JMP                     /YES, EXIT
DCA REMAIN
MQL                     /LOAD AC WITH QUOTIENT
```

Division Of A 12-Bit Signed Numbers

Enter with a 12-bit signed dividend in the AC and a 12-bit signed divisor in core memory. Exit with unsigned remainder in core memory location REMAIN and a 12-bit signed quotient in the AC.

```
CLL
SPA                     /DIVIDEND POSITIVE?
CMA CML IAC             /NO
MQL
TAD . +11
SPA                     /DIVISOR POSITIVE?
CMA CML IAC             /NO

DCA . +6
SNL                     /QUOTIENT NEGATIVE?
CMA                     /NO
CLL                     /SET SIGN INDICATOR
DCA SIGN
DVI                     /DIVISOR
DIVISOR                 /OVERFLOW
SZL                     /EXIT ON OVERFLOW
JMP
MQL
ISZ SIGN
CMA IAC
```

Section VI. EXTENDED MEMORY

When additional 4096-word memory banks are attached to the PDP-12, the Memory Extension Control provides access to the additional storage, both for programs and data. The registers of the Control are already built into the PDP-12; they are described in Section 3.22 in relation to LINC mode memory control. In PDP-8 mode, the functions of these registers are the same, but only a portion of each register is used. The Instruction Field (IF), Data Field (DF), and Instruction Field Buffer (IB) registers are each three bits long; the two low-order bits of the 5-bit total pertain only to LINC memory fields. The Save Field register (Interrupt Buffer) is only six bits long; in this case, the four high-order bits are unused.

4.14 REGISTERS

4.14.1 Instruction Field Register (IF), 3 Bits

These three bits serve as an extension of the PC for determining the 4096-word field from which executable instructions are to be taken. All direct memory references are made to registers in the Instruction Field. with one exception, all JMP and JMS instructions, whether direct or indirect, are registers within the Instruction Field. The exception is the first JMP or JMS executed after a CIF instruction is given. This causes the field to change.

4.14.2 Data Field Register (DF), 3 Bits

These three bits serve as an extension of the Memory Address register for determining which memory bank contains the operands to be accessed by indirect (only) memory references. The Data Field and Instruction Field may be set to the same bank.

4.14.3 Instruction Field Buffer (IB), 3 Bits

This serves as an input buffer for the IF. Except for a direct transfer from the console switches, all transfers into the IF must pass through the IB. When a CIF or RMF instruction is executed, information going to the IF is first placed in the IB. At the next occurrence of a JMP or JMS, the contents of the IB are transferred to the Instruction Field register, and programming continues in the new field, starting in the target register of the jump.

4.14.4 Save Field Register (SF), 6 Bits

Also called the Interrupt Buffer. When a program interrupt occurs, the contents of the IF and DF are stored in the Save Field register, as shown in Figure 4-10. After the interrupt has been serviced, an RMF instruction will cause the contents of the SF to be restored to the DF and IB. The SF can be examined by using the RIB instruction.

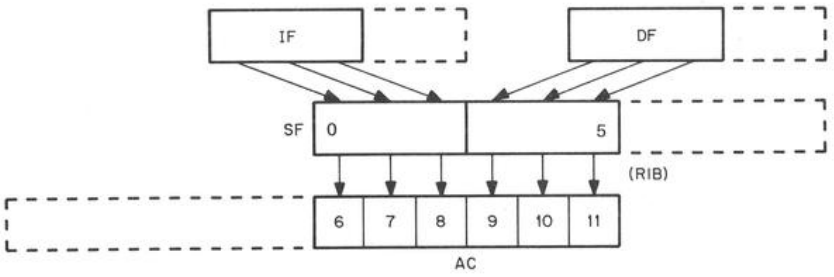


Figure 4-10. Data Path to SF and AC

4.14.5 Break Field Register (BF), 3 Bits

When an external device requires extended memory for the transfer of data using the Data Break Facility, the contents of the BF specify the memory bank to be accessed. The use of the register is described in detail in Chapter 5.

4.15 INSTRUCTIONS

All Extended Memory IOT instructions require 4.3 microseconds for execution.

CDF Change Data Field

Octal code: 62N1, $0 \leqslant N \leqslant 7$
Operation: The quantity N is transferred to the Data Field register. All subsequent indirect memory references by AND, TAD, ISZ, and DCA are to the new field.

CIF Change Instruction Field

Octal code: 62N2, $0 \leqslant N \leqslant 7$
Operation: The quantity N is transferred to the Instruction Field Buffer. At the occurrence of the next subsequent JMP or JMS instruction, whether direct or indirect, the contents of the IB are transferred to the IF. The effective address of the jump is placed in the PC, and the program continues from that address in the new Instruction Field.

In both CIF and CDF, the number N occupies bits 6-8 of the instruction code.

RDF Read Data Field

Octal code: 6214
Operation: The contents of the Data Field register are placed in bits 6, 7, and 8 of the AC. The other bits of the AC are unaffected.

RIF Read Instruction Field

Octal code: 6224
Operation: The contents of the Data Field register are placed in bits 6, 7, and 8 of the AC. The other bits of the AC are unaffected.

RIB Read Interrupt Buffer

Octal code: 6234
Operation: The contents of the Save Field register (Interrupt Buffer) are transferred to the AC, as follows: Bits 0-2 (IF) are placed in AC_{6-8}; bits 3-5 (DF) are placed in AC_{9-11}.

RMF Restore Memory Field

Octal code: 6244
Operation: The contents of the Save Field register are placed in the Instruction Field Buffer and DF as follows: Bits 0-2 (original Instruction Field) are transferred to the IB; bits 3-5 (original Data Field) are restored to the Data Field register. This instruction is used to restore the Memory Field registers after a program interrupt has been serviced. Normally, the next instruction after the RMF would be JMP I 0; the address of the interrupted program, stored in register 0000 of bank 0, is placed in the PC, and the contents of the IB are placed in the Instruction Field register; the program thus returns to the main program with the Memory Fields restored to their original values.

4.16 PROGRAMMING

All instructions, effective addresses, and directly-addressed operands are taken from the field specified by the contents of the Instruction Field Register. All indirectly-addressed operands are taken from (or are stored in) the field specified by the contents of the Data Field Register. The following chart shows the results of the four possible addressing combinations, when the IF and DF designate different memory fields.

Instruction Bits		*Fields*		*Effective Address*
Indirect	*Page*	*IF*	*DF*	
0	0	m	n	The operand is in Page 0 of Field *m* at the address specified by instruction bits 5-11.
0	1	m	n	The operand is in the current page of Field *m*.
1	0	m	n	The effective address of the operand is in Page 0 of Field *m* at the location specified by instruction bits 5-11. The operand is in Field *n*, in the location specified by the effective address.
1	1	m	n	The effective address is taken from the current page of Field *m*, at the location specified by instruction bits 5-11. The operand is in Field *n*.

4.16.1 Autoindexing

When any memory bank is used as an Instruction Field, registers 10-17 of that bank have autoindexing properties, just as the corresponding locations in field 0 do. This is necessary so that a program can operate correctly regardless of the actual memory bank assigned by the IF. When an autoindex register is indirectly addressed, the resulting effective address is used to obtain the operand from the Data Field specified by the DF.

Example:

C(IF) = 2. C(DF) = 4. C(AC) = 0.

In field 4: C(4326) = 1107
In field 2: C(0012) = 4325

The instruction, TAD I 12, is executed in field 2.

C(0012) + 1 → C(0012). Resulting effective address is 4326.
C(4326) in field 4 are added to the AC
C(AC) = 1107 when the instruction is completed.

4.16.2 Calling A Subroutine Across Fields

The problem is to let the subroutine know which field contains the calling program, so that it can return to the proper point when it's finished. This is most easily done by setting the DF to the same field as the IF, then setting

the IF to the field containing the subroutine, and executing a JMS to read the subroutine. The subroutine uses the DF to indirectly obtain data from the calling field, then transfers the C(DF) back to the IF Buffer to return to the calling program. The following example shows a general procedure for doing this.

```
/CALLING PROGRAM IN FIELD 2, SUBROUTINE IN FIELD 4
/CURRENT DATA FIELD IS 1
/CALLING SEQUENCE SAVES CURRENT DF, PUTS IF IN DF, CALLS
/SUBROUTINE. ON RETURN, ORIGINAL DF IS RESTORED

            ....
            CLA
            TAD RESDF       /CDF INSTRUCTION TO AC
            RDF             /C(DF) TO AC 6-8 FORMS CDF 10 (6211)
            DCA RESDF       /STORE IN SEQUENCE TO RESTORE DF
            TAD SETDF       CDF TO AC
            RIF             /C(IF) TO AC 6-8 FORMS CDF 20 (6221)
            DCA SETDF       /STORE IN SEQUENCE TO SET DF
SETDF       CDF             /SETS DF TO CURRENT IF
            CIF 40          /SET IF BUFFER TO SUBROUTINE FIELD 4
            JMS I SUBADR    /JUMP TO SUBROUTINE IN FIELD 4
RESDF       CDF 10          /RESTORES ORIGINAL DF (FIELD 1)
            ....
SUBADR      SUBRTN          /ABSOLUTE ADDRESS OF SUBROUTINE

/IN FIELD 4, THE SUBROUTINE HAS THE FOLLOWING GENERAL FORM

SUBRTN      0               /C(PC) FROM CALLING PROGRAM
            TAD RESIF       /CIF INSTRUCTION TO AC
            RDF             /C(DF) TO AC6-8 FORMS CIF 20 (6222)
            DCA RESIF       /STORE IN SEQUENCE TO RESTORE CALLING FIELD
            ....
            ....
            ....
RESIF       CIF 20          /SETS IF BUFFER TO RESTORE CALLING FIELD
            JMP I SUBRTN    /JUMP BACK TO CALLING PROGRAM
```

The original contents of the IF, placed in the DF by the calling program, are used to form a CIF instruction which is placed in the subroutine program sequence just before the exit, to restore the original calling field.

4.16.3 **Program Interrupt**

If, when the Interrupt facility is enabled, an interrupt request occurs, the contents of the IF and DF are saved in the Interrupt Buffer. The contents of the PC are stored in register 0000 of Field 0, and the next instruction is taken from register 0001 of Field 0. Regardless of the states of the Memory Field registers, a program interrupt always transfers control to Memory Field 0. When the interrupt has been serviced, the RMF instruction is used to restore the Memory Field registers to their original states. The last three instructions of a general interrupt service routine should be as follows:

```
            ....
            RMF             /RESTORES DF, PUTS ORIGINAL IF IN IF BUFFER
            ION             /ENABLE INTERRUPT
            JMP I 0         /SETS IF FROM IF BUFFER, AND RETURNS TO
                            /MAIN PROGRAM.
```

CHAPTER 5
INPUT/OUTPUT BUS AND PERIPHERALS

Because the processing power of the computer depends largely upon the range and number of peripheral devices that can be connected to it, the PDP-12 has been designed to interface readily with a broad variety of external equipment. The following chapter defines the interface characteristics of the computer to allow the reader to design and implement any electrical interfaces required to connect devices to the PDP-12.

The simple I/O technique of the PDP-12, the availability of DEC's FLIP CHIP logic circuit modules, and DEC's policy of giving assistance wherever possible allow inexpensive, straightforward device interfaces to be realized. Should questions arise relative to the computer interface characteristics, the design of interfaces using DEC modules, or installation planning, customers are invited to telephone any of the sales offices or the main plant in Maynard, Massachusetts. Digital Equipment Corporation makes no representation that the interconnection of its circuit modules in the manner described herein will not infringe on exisitng or future patent rights. Nor do the descriptions contained herein imply the granting of licenses to use, manufacture, or sell equipment constructed in accordance therewith.

The PDP-12 contains a processor and core memory composed of Digital's M Series TTL circuit modules. These circuits have an operating temperature exceeding the limits of 50°F to 110°F, so no air-conditioning is required at the computer site. Standard 115V, 50/60-CPS power operates an internal solid state power supply that produces all required voltages and currents. High-capacity high-speed I/O capabilities of the PDP-12 allow it to operate a variety of peripheral devices in addition to the standard Teletype keyboard/printer, tape reader, and tape punch. DEC options, consisting of an interface and normal data processing equipment, are available for connecting into the computer system. These options include a random access disc file, card equipment, line printers, magnetic tape transports, magnetic drums, analog-to-digital converters, CRT displays, and digital plotters. The PDP-12 system can also accept other types of instruments or hardware devices that have an appropriate interface. Up to 61 devices requiring three programmed command pulses, or up to 183 devices requiring one programmed command pulse can be connected to the computer. Interfacing of any devices to the computer requires no modifications to the processor and can be achieved in the field.

Control of some kind is needed to determine when an information exchange is to take place between the PDP-12 and peripheral equipment and to indicate the location(s) in the computer memory which will accept or yield the data. Either the computer program or the device external to the computer can exercise this control. Transfers controlled by the computer, hence under control of its stored program, are called programmed data transfers.

Transfers made at times controlled by the external devices through the data break facility are called data break transfers.

Programmed Data Transfers

The majority of I/O transfers occur under control of the computer program. To transfer and store information under program control requires about six times as much computer time as under data break control. In terms of real time, the duration of a programmed transfer is rather small, due to the high speed of the computer, and is well beyond that required for laboratory or process control instrumentation.

To realize full benefit of the built-in control features of the PDP-12 programmed I/O transfers should be used in most cases. Controls for devices using programmed data transfers are usually simpler and less expensive than controls for devices using data break transfers. Using programmed data transfer facilities simultaneous operation of devices is limited only by the relative speed of the computer with respect to the device speeds, and the search time required to determine the device requiring service. Analog-to-digital converters, digital-to-analog converters, digital plotters, line printers, message switching equipment, and relay control systems typify equipment using only programmed data transfers.

Data Break Transfers

Devices which operate at very high speed or which require very rapid response from the computer use the data break facilities. Use of these facilities permits an external device, almost arbitrarily, to insert or extract-words from the computer core memory, bypassing all program control logic. Because the computer program has no cognizance of transfers made in this manner, programmed checks of input data are made prior to use of information received in this manner. The data break is particularly well-suited for devices that transfer large amounts of data in block form, e. g., random access disc file, high-speed magnetic tape systems, high-speed drum memories, or CRT display systems containing memory elements.

Program Interrupt

It is sometimes very useful for a program to be able to initiate operation of an I/O device and then continue with execution of programming which is not immediately related to the input-output operation, rather than to wait for the device to become ready to transfer data. In this mode of operation, the device itself through use of the PDP-12 *Program Interrupt* facility initiates execution of the programming to transfer data to or from the computer. When the device *requires service*, i.e., when the device is ready to transfer data, it transmits an *interrupt request* signal to the computer. This signal causes the execution of the program currently underway to be interrupted, and program control to be transferred to a specific memory location. This location having been filled with a JMP to the starting point of a program to control the data transfer.

After completion of the data transfer, the interrupted program is resumed. The information needed to return to the interrupted program is saved at the time the program interrupt occurs. The program interrupt hardware is designed so that interrupt requests from devices may be ignored if the program so desires.

Logic Symbols

The PDP-12 uses TTL logic internally. In order to discuss some of the internal logic pertaining to interfacing, it is necessary to understand the TTL symbology used in the PDP-12. The logic symbols are shown in Figure 5-1.

Signal Names

All signals not originating at a flip-flop output are true when the line is at the level indicated by the suffix H (high) or L (low). Thus a line labeled IOP 1 H is high when the pulse is being generated and is otherwise ground. Similarly, AC CLEAR L is at ground for assertion, and positive otherwise.

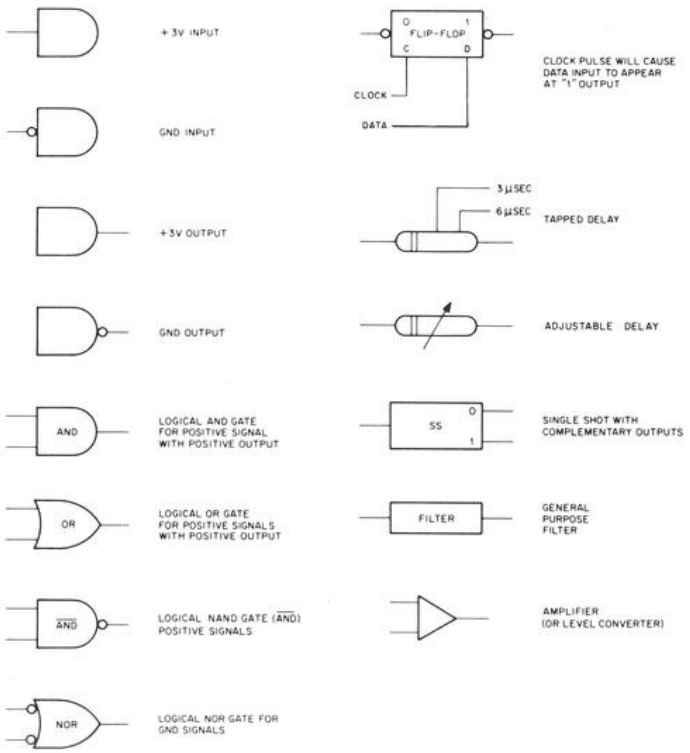


Figure 5-1. Logic Symbols

Signals originating at flip-flops are fefined in terms of the flip-flop state. The following table illustrates the convention.

Signal Name	*State of MB Flip-Flop*	*Signal Voltage*
MB 03 (0)H	0	+3V
MB 03 (0)L	0	0V
MB 03 (1)L	1	0V
MB 03 (1)H	1	+3V

Note: The line MB 03(0)H is the same line as MB 03(1)L and MB 03(1)H is the same line as MB 03(0)L.

5.1 PROGRAMMED DATA TRANSFERS AND I/O CONTROL

The majority of I/O transfers take place under control of the PDP-12 program taking advantage of control elements built into the computer. Although programmed transfers take more computer and actual time than do data break transfers, the timing discrepancy is insignificant, considering the high speed of the computer with respect to most peripheral devices. The maximum data transfer rate for programmed operations of 12-bit words is 148 kc when no status checking end transfer check, etc., is done. This speed is well beyond the normal rate required for typical laboratory or process control instrumentation.

The PDP-12 is a parallel-transfer machine that distributes and collects data in bytes of up to twelve bits. All programmed data transfers take place through the accumulator, the 12-bit arithmetic register of the computer. The computer program controls the loading of information into the accumulator (AC) for an output transfer, and for storing information in core memory from the AC for an input transfer. Output information in the AC is power amplified and supplied to the interface connectors for bussed connection to many peripheral devices. Then the program-selected device can sample these signal lines to strobe AC information into a control or information register. Input data arrives at the AC as pulses received at the interface connectors from bussed outputs of many devices. Gating circuits of the program-selected device produce these pulses. Command pulses generated by the device flow to the input/output skip facility (IOS) to sample the condition of I/O device flags. The IOS allows branching of the program based upon the condition or availability of peripheral equipment, effectively making programmed decisions to continue the current program or jump to another part of the program, such as a subroutine that services an I/O device.

The bussed system of input/output data transfers imposes the following requirements on peripheral equipment.

a. The ability of each device to sample the select code generated by the computer during IOT instructions and when selected, to be capable of producing sequential IOT command pulses in accordance with the computer-generated IOP pulses. Circuits which perform these functions in the peripheral device are called the Device Selector (DS).

b. Each device receiving output data from the computer must contain gating circuits at the input of a receiving register capable of strobing the AC signal information into the register when triggered by a command pulse from the DS. Gating is also recommended at the input to the peripheral device in order to minimize loading on the BAC signal lines.

c. Each device which supplies input data to the computer must contain gating circuits at the output of the transmitting register capable of sampling the information in the output register and supplying a pulse to the computer input bus when triggered by a command pulse from the DS.

d. Each device should contain a busy/done flag (flip-flop) and gating circuits which can pulse the computer input/output skip bus upon command from the DS when the flag is set in the binary 1 state to indicate that the device is ready to transfer another byte of information.

Figure 5-3 shows the information flow within the computer which effects a programmed data transfer with input/output equipment. All instructions stored in core memory as a program sequence are read into the memory buffer register (MB) for execution. The transfer of the operation code in the three most significant bits (bits 0, 1, and 2) of the instruction into the instruction register (IR) takes place and is decoded to produce appropriate control signals. The computer, upon recognition of the operation code as an IOT instruction, enters a 4.25 μsec expanded computer cycle and enables the IOP generator to produce time sequenced IOP pulses as determined by the three least significant bits of the instruction (bits 9, 10, and 11 in the MB). These IOP pulses and the buffered output of the select code from bits 3-8 of the instruction word in the MB are bussed to device selectors in all peripheral equipment. Figure 5-4 indicates the timing of programmed data transfers and Figure 5-2 shows the decoding of the IOT instruction.

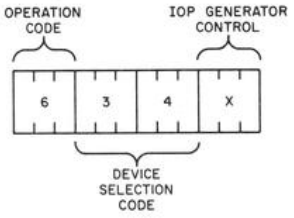


Figure 5-2. IOT Instruction Decoding

Devices which require immediate service from the computer program, or which take an exorbitant amount of computer time to discontinue the main program until transfer needs are met, can use the program interrupt (PI) facility. In this mode of operation, the computer can initiate operation of I/O equipment and continue the main program until the device requests servicing. A signal input to the PI requesting a program interrupt causes storing of the conditions of the main program and initiates a subroutine to service the device. At the conclusion of this subroutine, the main program is reinstated until another interrupt request occurs.

5.1.1 Timing and IOP Generator

When the IR decoder detects an operation code of 6000_8 it identifies an IOT instruction and the computer generates a slow cycle. The Slow Cycle signal ANDS with TP4 to generate I/O Start and Sets the I/O PAUSE flip-flop. The logic of the IOP generator consists of a re-entrant delay chain which generates three time states. These time states are gated with MB bits 11, 10 and 9 to generate IOP 1, IOP 2 and IOP 4 respectively. Note that an IOP is generated only if the corresponding MB bit is set although the I/O timing remains constant. At the end of each IOP, the state of the I/O interface is sampled by an I/O strobe pulse.

Following the end of IOP 4 time, the PAUSE flip-flop is reset and the normal timing chain is restarted.

Unlike PDP-8/I, the PDP-12 makes no timing distinction between internal I/O functions and normal I/O. Thus all I/O instructions cause the slow cycle.

Instruction Bit	*IOP Pulse*	*IOT Pulse*	*Event Time*	*Used Primarily For, But Not Restricted To*
11	IOP 1	IOT 1	1	Sampling Flags, Skipping.
10	IOP 2	IOT 2	2	Clearing Flags, Clearing AC.
9	IOP 4	IOT 4	3	Reading Buffers, Loading Buffers and Clearing Buffers.

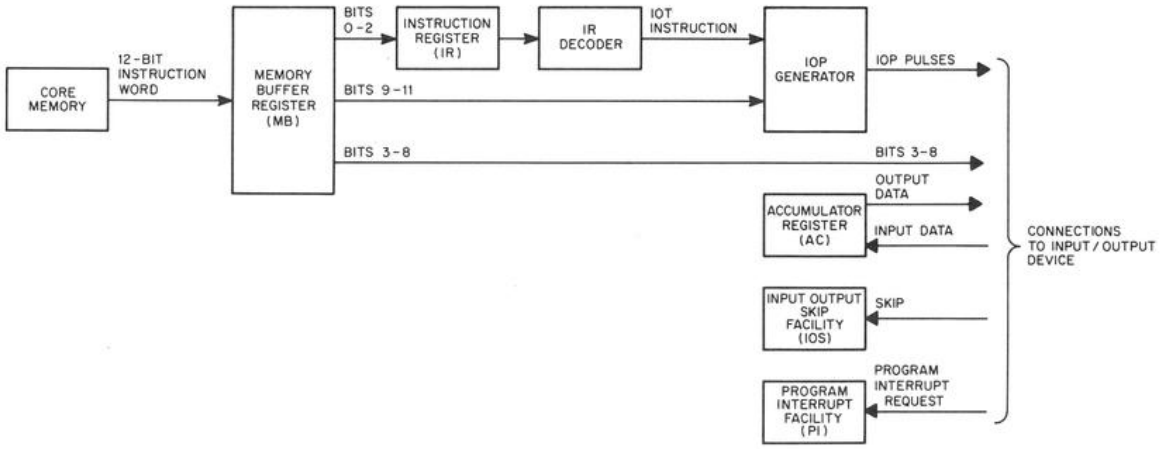


Figure 5-3. Programmed Data Transfer Interface Block Diagram

5.1.2 Device Selector (DS)

Bits 3 through 8 of an IOT instruction serve as a device or subdevice select code. Bus drivers in the processor buffer both the binary 1 and 0 output signals of MB_{3-8} and distribute them to the interface connectors for bussed connection to all device selectors. Each DS is assigned a select code and is enabled only when the assigned code is present in the MB. When enabled, a DS regenerates IOP pulses as IOT command pulses and transmits these pulses to skip, input, or output gates within the device and/or to the processor to clear the AC.

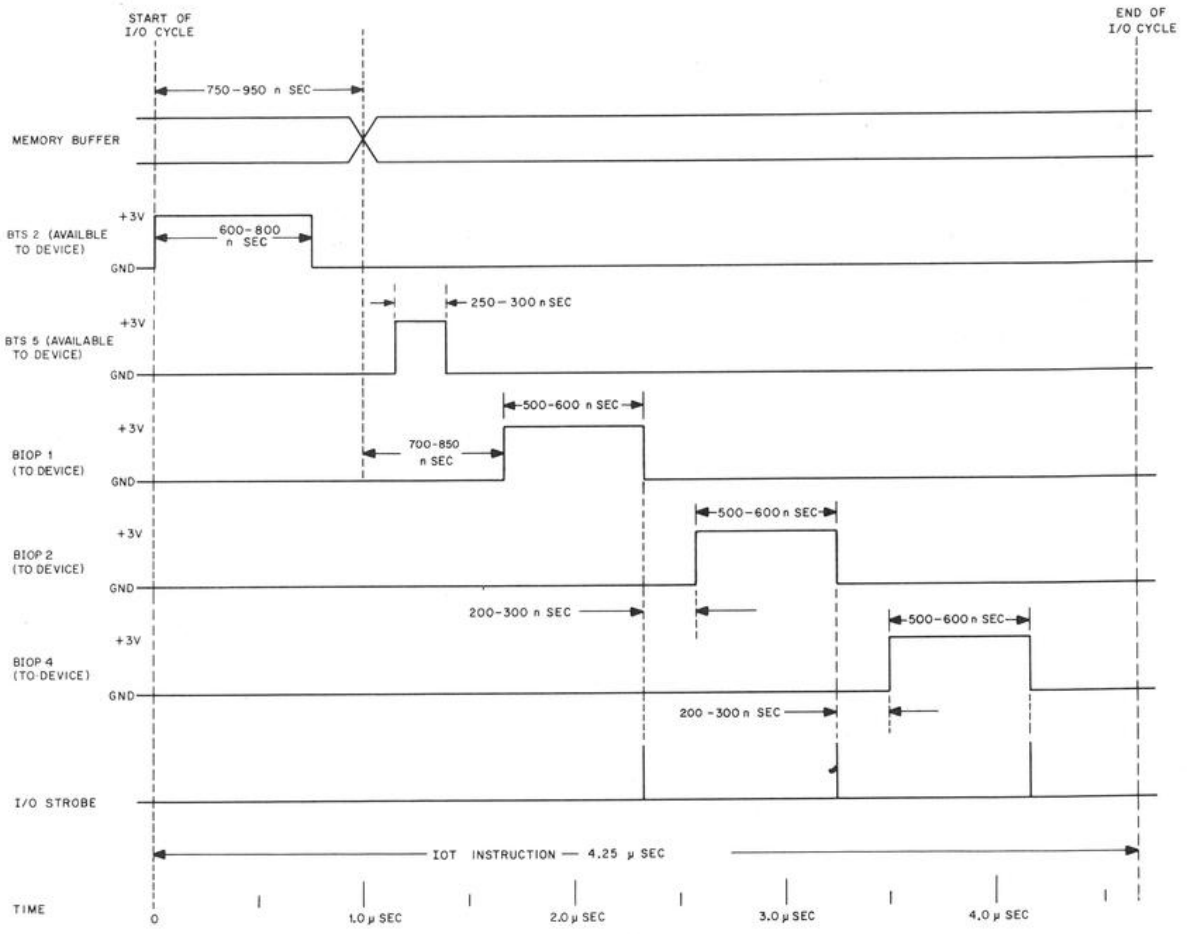


Figure 5-4. Programmed Data Transfer Timing

Each group of three command pulses requires a separate DS channel, and each DS channel requires a different select code (or I/O device address). One I/O device can, therefore, use several DS channels. Note that the processor produces the pulses identified as IOP 1, IOP 2, and IOP 4 and supplies them to all device selectors. The device selector produces pulses IOT 1, IOT 2, and IOT 4 which initiate a transfer or effect some control. Figure 5-5 shows generation of command pulses by several DS channels.

The logical representation for a typical channel of the DS, using channel 34, is shown in Figure 5-6. A 6-input NAND gate wired to receive the appropriate signal outputs from the MB_{3-8} for select code 34 activates the channel. In the DS module, the NAND gate contains 8 input terminals; 6 of these connect to the complementary outputs of MB_{3-8}, and 2 are open to receive subdevice or control condition signals as needed. Either the 1 or the 0 signal from each MB bit is connected to the NAND gate when establishing the select code. The ground level output of the NAND gate indicates when the IOT instruction selects the device, and can therefore enable circuit operations with the device. This output also enables three power NAND gates, each of which produces an output pulse if the corresponding IOT pulse occurs. The positive output from each gate is an IOT command pulse identified by the select code and the number of the initiating IOP pulse. Three inverters receive the positive IOT pulses to produce complementary IOT output pulses. An amplifier module can be connected in each channel of the DS to provide greater output drive.

5.1.3 Input/Output Skip (IOS)

Generation of an IOS pulse can be used to test the condition or status of a device flag, and to continue to or skip the next sequential instruction based upon the results of this test. This operation is performed by a 2-input AND gate in the device connected as shown in Figure 5-7. One input of the skip gate receives the status level (flag output signal), the second input receives an IOT pulse, and the output drives the computer skip (designated SKIP BUS L) to ground when the skip conditions are fulfilled. The state of the skip bus is sampled at the end of each IOT. If the bus has been driven to ground, the content of the program counter is incremented by 1 to advance the program count without executing the instruction at the current program count. In this manner an IOT instruction can check the status of an I/O device flag and skip the next instruction if the device requires servicing. Programmed testing in this manner allows the routine to jump out of sequence to a subroutine that services the device tested.

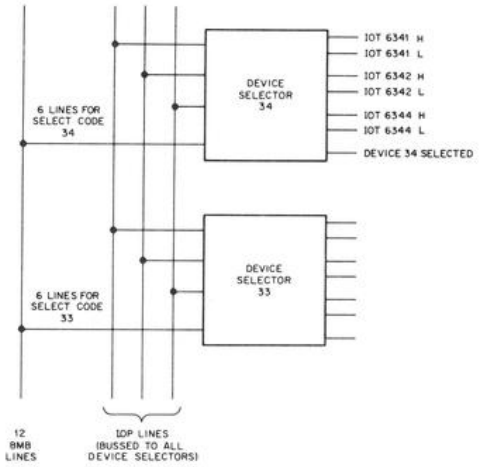


Figure 5-5. Generation of IOT Command Pulses by Device Selectors

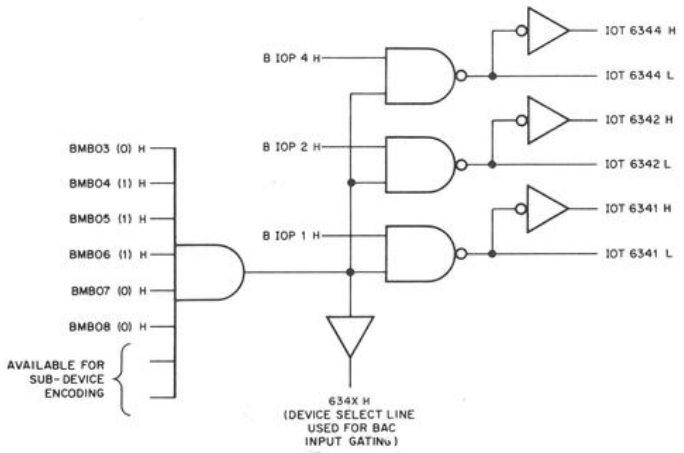


Figure 5-6. Typical Device Selector (Device 34)

Assuming that a device is already operating, a possible program sequence to test its availability is:

```
100,    6342    /SKIP IF DEVICE 34 IS READY
101,    5100    /JUMP .-1
102,    5XXX    /ENTER SERVICE ROUTINE FOR DEVICE 34
```

When the program reaches address 100, it executes an instruction skip with 6342. The skip occurs only if device 34 is ready when the IOT 6342 command is given. If device 34 is not ready, the flag signal disqualifies the skip gate, and the skip pulse does not occur. Therefore, the program continues to the next instruction which is a jump back to the skip instruction. In this example, the program stays in this waiting loop until the device is ready to transfer data, at which time the skip gate in the device is enabled and the skip pulse is sent to the computer IOS facility. When the skip occurs, the instruction in location 102 transfers program control to a subroutine to service device 34. This subroutine can load the AC with data and transfer it to device 34, or can load the AC from a register in device 34 and store it in some known core memory address.

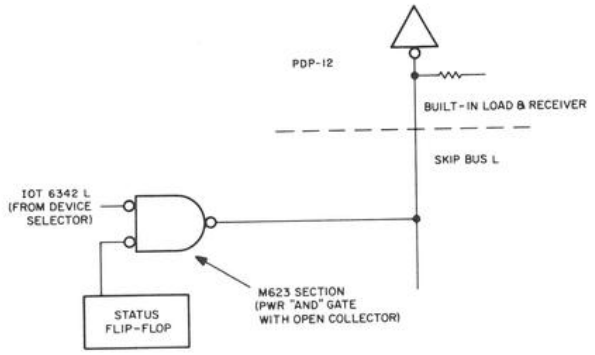


Figure 5-7. Use of IOS to Test the Status of an External Device

5.1.4 Accumulator

The binary 1 output signal of each flip-flop of the AC, buffered by a bus driver, is available at the interface connectors. These computer data output lines are bus connected to all peripheral equipment receiving programmed data output information from the PDP-12. A terminal on each flip-flop of the AC is connected to the interface connectors for bussing to all peripheral equipment supplying programmed data input to the PDP-12. An IOP that drives the AC input bus terminal to ground causes setting of the corresponding AC flip-flop to the binary 1 state. Output and input connections to the accumulator appear in Figure 5-8.

The status of the link bit is not available to enter into transfers with peripheral equipment (unless it is rotated into the AC). A bus driver continuously buffers the output signal from each AC flip-flop. These buffered accumulator (BAC) signals are available at the interface connectors.

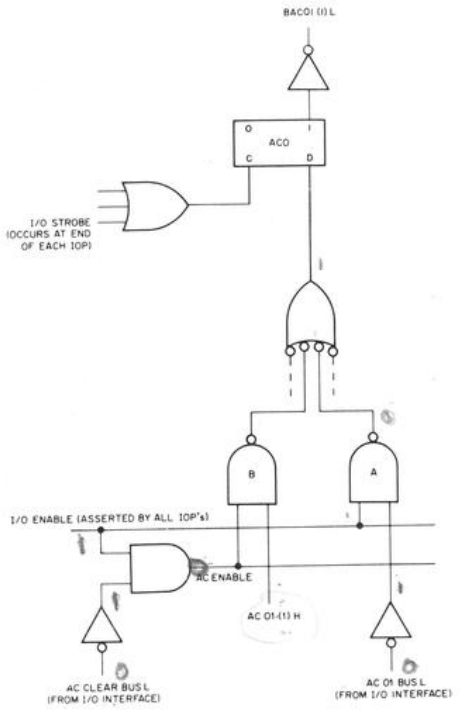


Figure 5-8. Accumulator Input or Output

5.1.5 **Input Data Transfers**

When ready to transfer data into the PDP-12 accumulator, the device sets a flag connected to the IOS. The program senses the ready status of the flag and issues an IOT instruction to read the content of the external device buffer register into the AC. If the AC CLEAR BUS L is not asserted, the resultant word in the AC is the inclusive OR of the previous word in the AC and the word transferred from the device buffer register. The AC CLEAR BUS L may also be used as an I/O AC clear by activating only this line from a separate IOT.

The illustration in Figure 5-9 shows that the accumulator has an input bus for each bit flip-flop. Setting a 1 into a particular bit of the accumulator necessitates grounding of the interface input bus by the standard interface gate. In the illustration, the 2-input AND gates set various bits of the accumulator. In this case an IOT pulse is AND combined with the flip-flop state of the external device to transfer into the accumulator. (The program need not include a clear AC command prior to loading in this manner.)

Following the transfer (possibly in the same instruction) the program can issue a command pulse to initiate further operation of the device and/or clear the device flag.

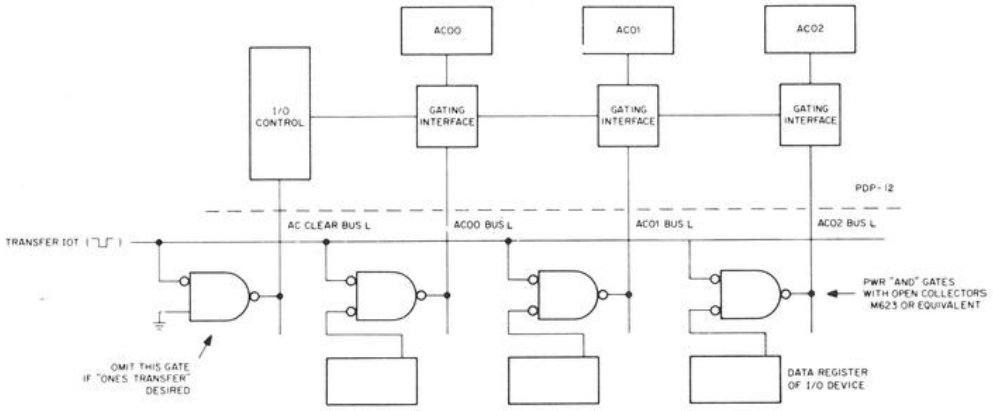


Figure 5-9. Loading Data into the Accumulator from an External Device

5.1.6 **Output Data Transfers**

The AC is loaded with a word (e.g., by a CLA TAD instruction sequence); then the IOT instruction is issued to transfer the word into the control or data register of the device by an IOT pulse (e.g., IOP 2), and operation of the device is initiated by another IOT pulse (e.g., IOP 4). The data word transferred in this manner can be a character to be operated upon, or can be a control word sampled by a status register to establish a control mode.

The BAC lines should be gated by the select code at each device to prevent excessive loading. A special module, the M101, is provided for this purpose.

Since the BAC interface bus lines continually present the status of the AC flip-flops, the receiving device can strobe them to sense the value in the accumulator. In Figure 5-10 a strobe pulse samples six bits of the accumulator to transfer to an external 6-bit data register. Since this is a jam transfer, it is not necessary to clear the external data register. The gates driving the external data register are part of the external device and are not supplied by the computer. The data register can contain any number of flip-flops up to a maximum of twelve. If more than twelve flip-flops are involved, two or more transfers must take place. Obviously the strobe pulse shown in Figure 5-10 must occur when the data to be placed in the external data register is held in the accumulator. This pulse therefore must be under computer control to effect synchronization with the operation or program of the computer.

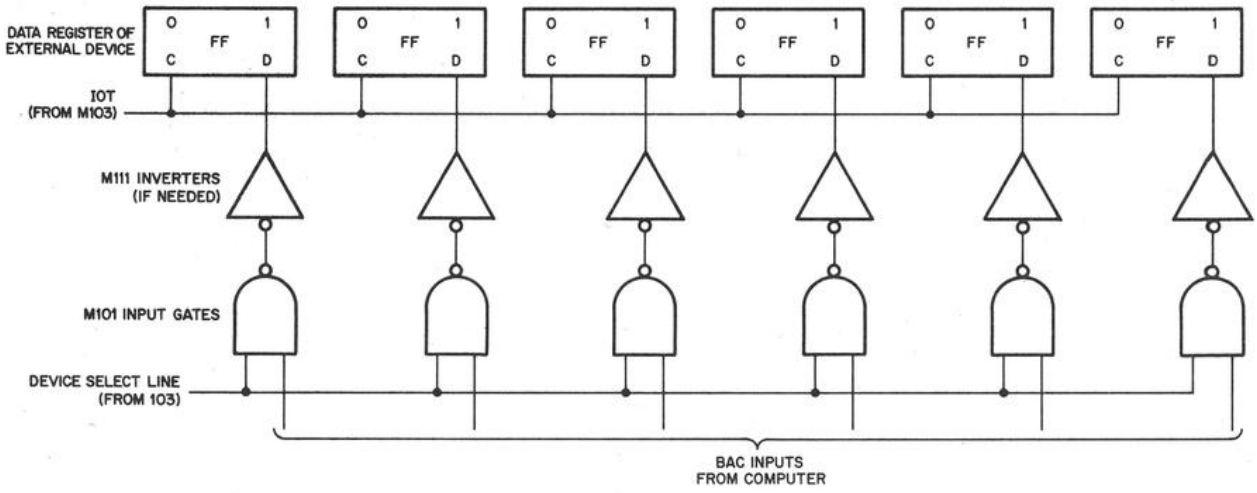


Figure 5-10. Loading a Six-Bit Word into an External Device from the Accumulator

5.1.7 **Program Interrupt (PI)**

When a large amount of computing is required, the program should initiate operation of an I/O device then continue the main program, rather than wait for the device to become ready to transfer data. The program interrupt facility, when enabled by the program, relieves the main program of the need for repeated flag checks by

allowing the ready status of I/O device flags to automatically cause a program interrupt. When the program interrupt occurs, program control transfers to a subroutine that determines which device requested the interrupt and initiates an appropriate service routine.

In the example shown in **Figure 5-11**, a flag signal from a status flip-flop operates a standard gate with no internal load. When the status flip-flop indicates the need for device service, the inverter drives the Program Interrupt Request bus to ground to request a program interrupt.

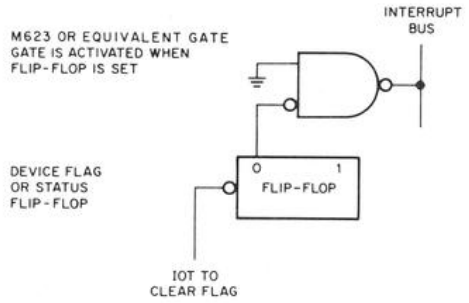


Figure 5-11. Program Interrupt Request Signal Origin

If only one device is connected to the PI facility, program control can be transferred directly to a routine that services the device when an interrupt occurs. This operation occurs as follows, (example in PDP-8 mode):

Tag	*Address*	*Instruction*	*Remarks*
	1000	.	/MAIN PROGRAM
	1001	.	/MAIN PROGRAM CONTINUES
	1002	.	/INTERRUPT REQUEST OCCURS
			/INTERRUPT OCCURS
	0000		/PROGRAM COUNT (PC = 1003) IS
			/STORED IN 0000
	0001	JMP SR	/ENTER SERVICE ROUTINE
SR	2000	.	/SERVICE SUBROUTINE FOR
		.	/INTERRUPTING DEVICE AND
			/SEQUENCE TO RESTORE AC, AND
	3001	.	/RESTORE LINK IF REQUIRED

Tag	*Address*	*Instruction*	*Remarks*
	3002	RMF	/RESTORE MEMORY FIELDS
	3003	ION	/TURN ON INTERRUPT
	3004	JMP 1 0000	/RETURN TO MAIN PROGRAM
	1003	.	/MAIN PROGRAM CONTINUES
	1004	.	
		.	
		.	

In most PDP-12 systems numerous devices are connected to the PI facility, so the routine beginning in core memory address 0001 must determine which device requested an interrupt. The interrupt routine determines the device requiring service by checking the flags of all equipment connected to the PI and transfers program control to a service routine for the first device encountered that has its flag in the state required to request a program interrupt. In other words, when program interrupt requests can originate in numerous devices, each device flag connected to the PI must also be connected to the IOS.

Multiple Use of IOS and PI

In common practice, more than one device is connected to the PI facility. In the basic PDP-12, the teletype flags are already connected. Therefore, since the computer receives a request that is the inclusive OR of requests from all devices connected to the PI, the IOS must ibentify the device making the request. When a program interrupt occurs, a routine is entered from address 0001 in PDP-8 mode (0041 in LINC mode) to sequentially check the status of each flag connected to the PI and to transfer program control to an appropriate service routine for the device whose flag is requesting a program interrupt. Figure 5-12 shows IOS and PI connections for two typical devices.

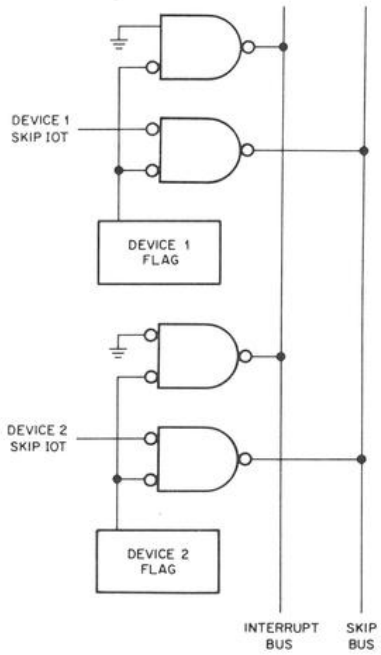


Figure 5-12. Multiple Inputs to IOS and PI Facilities

The following program example illustrates how the program interrupt routine determines the device requesting service (example in PDP-8 mode):

```
Tag             Address     Instruction         Remarks

                1000            .           /MAIN PROGRAM
                1001            .           /MAIN PROGRAM CONTINUES
                1002            .           /INTERRUPT REQUEST OCCURS
                                            /INTERRUPT OCCURS
                0000                        /STORE PC (PC = 1003)
                0001        JMP FLG CK      /ENTER ROUTINE TO DETERMINE
                                            /WHICH DEVICE CAUSED INTERRUPT
FLG CK,                     IOT 6341        /SKIP IF DEVICE 34 IS REQUESTING
                            SKP             /NO – TEST NEXT DEVICE
                            JMP SR34        /ENTER SERVICE ROUTINE 34
                            IOT 6441        /SKIP IF DEVICE 44 IS REQUESTING
                            SKP             /NO – TEST NEXT DEVICE
                            JMP SR44        /ENTER SERVICE ROUTINE 44
                            IOT 6541        /SKIP IF DEVICE 54 IS REQUESTING
                            SKP             /NO – TEST NEXT DEVICE
                            JMP SR44        /ENTER SERVICE ROUTINE 54
                                .
                                .
                                .
```

Assume that the device that caused the interrupt is an input device (e.g., tape reader). The following example of a device service routine might apply:

```
Tag             Instruction         Remarks

SR,             DAC TEMP            /SAVE AC
                IOT XX              /TRANSFER DATA FROM DEVICE
                                    /BUFFER TO AC
                DAC I 10            /STORE IN MEMORY LIST
                ISZ COUNT           /CHECK FOR END
                SKP                 /NOT END
                JMP END             /END. JUMP TO ROUTINE TO HANDLE
                                    /END OF LIST CONDITION
                    .
                    .               /RESTORE LINK AND OTHER STATUS IF REQUIRED
                    .
                TAD TEMP            /RELOAD AC
                RMF                 /RESTORE MEMORY FIELDS
                ION                 /TURN ON INTERRUPT
                JMP I 0             /RETURN TO PROGRAM
```

If the device that caused the interrupt was essentially an output device (receiving data from computer), the IOT – then – DAC I 10 sequence might be replaced by a TAD I 10 – then – IOT sequence.

5.2 DATA BREAK TRANSFERS

The data break facility allows an I/O device to transfer information directly with the PDP-12 core memory on a cycle-stealing basis. Up to seven devices can connect to the data break facility through the optional Data Multiplexer Type DM01. The data break is particularly well-suited for devices which transfer large amounts of information in block form.

Peripheral I/O equipment operating at high speeds can transfer information with the computer through the data break facility more efficiently than through programmed means. The combined maximum transfer rate of the data break facility is 7.5 million bits per second. Information flow to effect a data break transfer with an I/O device appears in Figure 5-13.

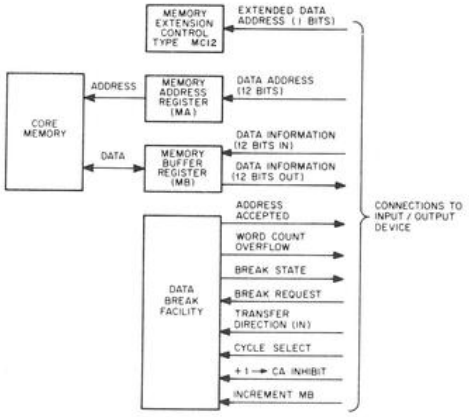


Figure 5-13. Data Break Transfer Interface Block Diagram

In contrast to programmed operations, the data break facilities permit an external device to control information transfers. Therefore, a data-break device interfaces require more control logic circuits, causing a higher cost than programmed-transfer interfaces.

Data breaks are of two basic types: single-cycle and three-cycle. In a single-cycle data break, registers in the device (or device interface) specify the core memory address of each transfer and count the number of transfers to determine the end of data blocks. In the three-cycle data break two computer core memory locations perform these functions, simplifying the device interface by omitting two hardware registers.

In general terms, to initiate a data break transfer of information, the interface control must do the following:

a. Specify the affected address in core memory.

b. Provide the data word by establishing the proper logic levels at the computer interface (assuming an input data transfer), or provide readin gates and storage for the word (assuming an output data transfer).

c. Provide a logical signal to indicate direction of data word transfer.

d. Provide a logical signal to indicate single-cycle or three-cycle break operation.

e. Request a data break by supplying a proper signal to the computer data break facility.

5.2.1 Single-Cycle Data Breaks

Single-cycle breaks are used for input data transfers to the computer, output data transfers from the computer, and memory increment data breaks. Memory increment is a special data break in which the content of a memory address is read, incremented by 1, and rewritten at the same address. It is useful for counting iterations or external events without disturbing the computer program counter (PC) or Accumulator (AC) registers.

5.2.2 Input Data Transfers

Figure 5-14 illustrates timing of an input transfer data break. The address to be affected in core is normally provided in the device interface in the form of a 12-bit flip-flop register (data break address register) which has been preset by the interface control by programmed transfer from the computer.

External registers and control flip-flops supplying information and control signals to the data break facility and other PDP-12 interface elements are shown in Figure 5-15. The input buffer register (IB in Figure 5-15) holds the 12-bit data word to be written into the computer core memory location specified by the address contained in the address register (AR in Figure 5-14).

Appropriate output terminals of these registers are connected to the computer to supply ground potential to designate binary 1's. Since most devices that transfer data through the data break facility are designed to use either single-cycle or three-cycle breaks, but not both, the Cycle Select signal can usually be supplied from a stable source (such as a ground connection or a +3v clamped load resistor) rather than from a bistable device as shown in Figure 5-15.

Other portions of the device interface, not shown in Figure 5-15, establish the data word in the input buffer register, set the address into the address register, set the direction flip-flop to indicate an input data transfer, and control the break request flip-flop. These operations can be performed simultaneously or sequentially, but all transients should occur before the data break request is made. Note that the device interface need supply only static levels to the computer, minimizing the synchronizing logic circuits necessary in the device interface.

When the data break request arrives, the computer completes the current instruction, generates an Address Accepted pulse (at TP1, the beginning of the break cycle) to acknowledge receipt of the request, then enters the Break state to effect the transfer. The Address Accepted pulse can be used in the device interface to clear the break request flip-flop, increment the content of the address register, etc. If the Break Request signal is removed before TP4 time of the data break cycle, the computer performs the transfer in one 1.6 μsec cycle and returns to programmed operation.

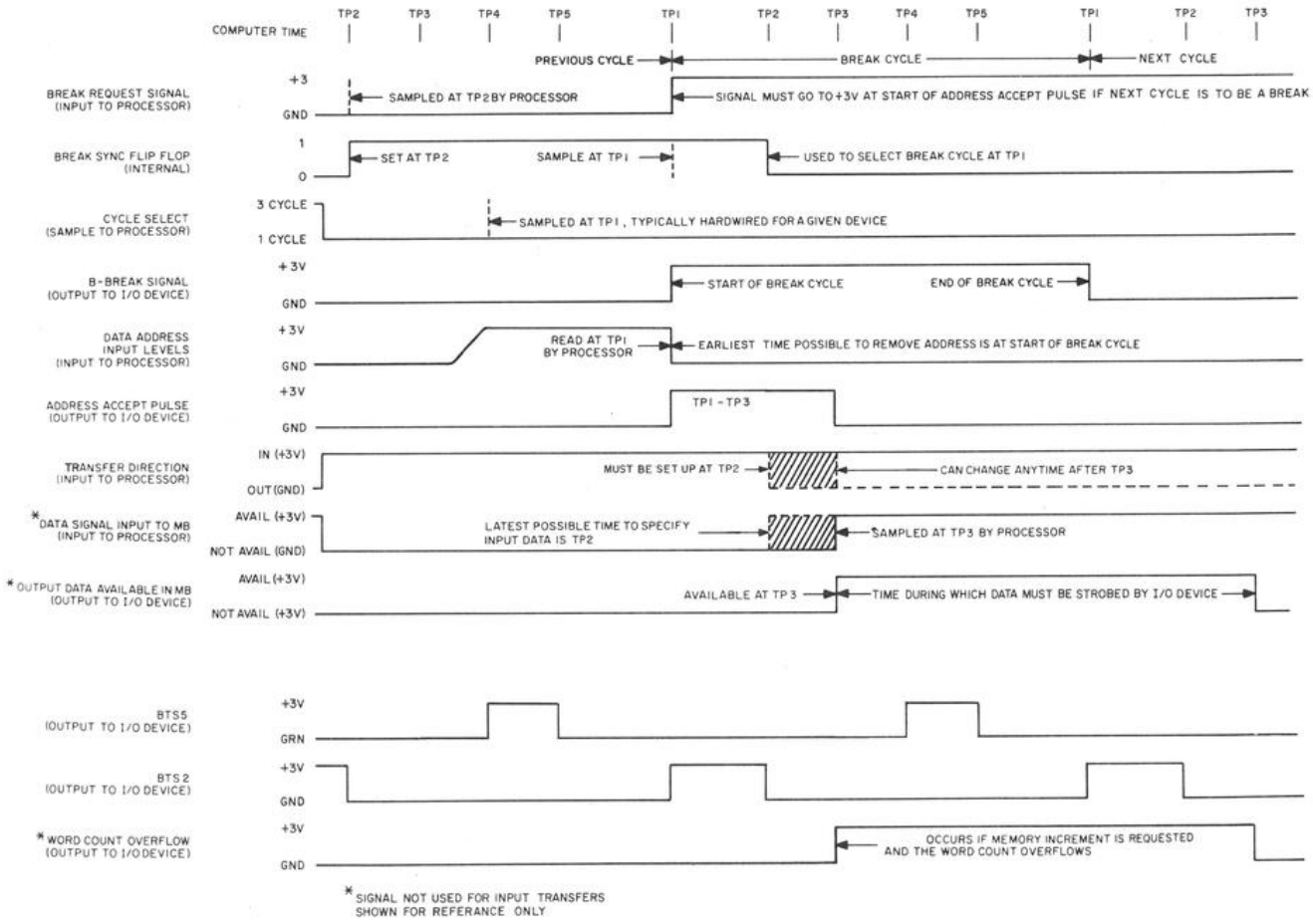


Figure 5-14. Single Cycle Data Break Input Transfer Timing Diagram

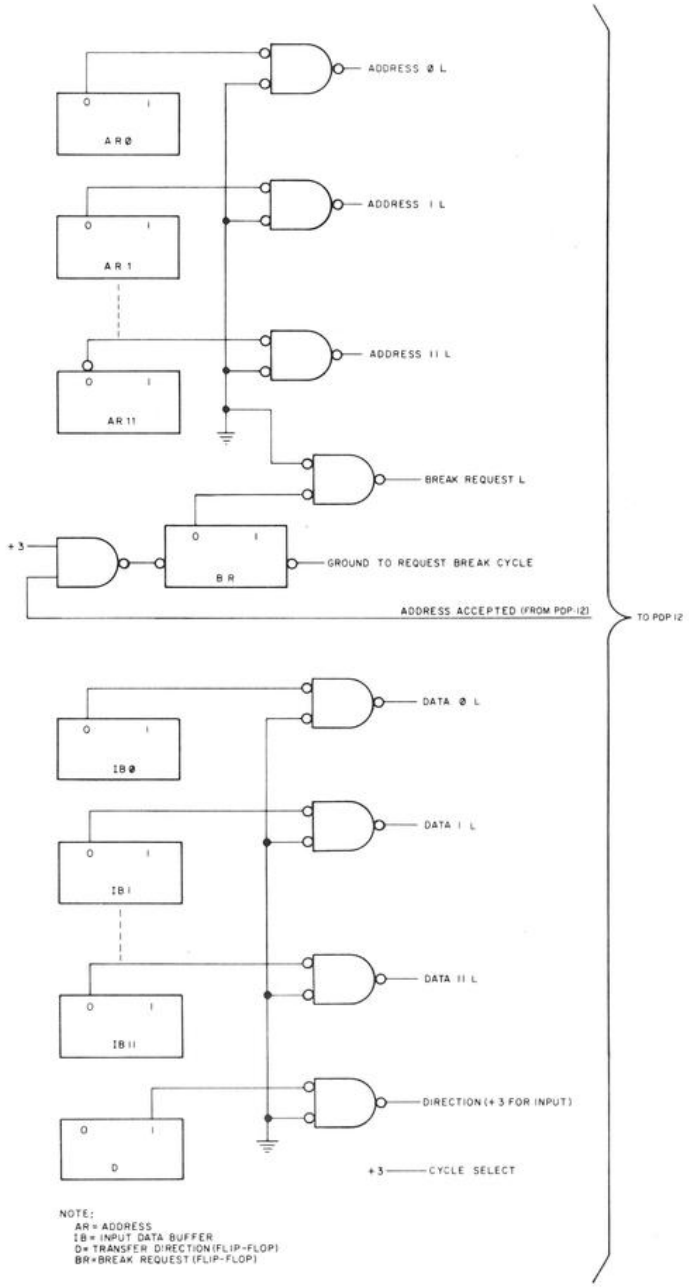


Figure 5-15. Device Interface Logic for Single-Cycle Data Break Input Transfer

5.2.3 Output Data Transfers

Timing of operations occurring in a single-cycle output data break is shown in Figure 5-16. Basic logic circuits for the device interface used in this type of transfer are shown in Figure 5-17. Address and control signal generators are similar to those discussed previously for input data transfers, except that the Transfer Direction signal must be at ground potential to specify the output transfer of computer information. An output data register (OB in Figure 5-17) is usually required in the device interface to receive the computer information. The device must supply strobe pulses for all data transfers out of the computer (programmed or data break) since circuit configuration and timing characteristics differ in each device.

When the data break request arrives, the computer completes the current instruction and generates an Address Accepted pulse as it enters the Data Break cycle. At TP1 time the address supplied to the PDP-12 is loaded into the MA, and the Break state is entered. Not more than 900 nsec after TP1 (at TP3 time), the content of the device-specified core memory address is read and available in the MB' (This word is automatically rewritten at the same address during the last half of the Break cycle and is available for programmed operations when the data break is finished.) Data Bit signals are available as static levels of ground potential for binary 0's and +3v for binary 1's. The MB is changed at TP3 time of each computer cycle, so the data word is available in the MB for approximately 1.6 μsec to be strobed by the device interface.

Generation of the strobe pulse by the device interface can be synchronized with computer timing through use of timing pulses BTS2 or BTS5, which are available at the computer interface. In addition to a timing pulse (delayed or used directly from the computer), generation of this strobe pulse should be gated by condition signals that occur only during the Break cycle of an output transfer. Figure 5-17 shows typical logic circuits to effect an output data transfer. In this example, BTS5 and B BREAK set the BREAK ENABLE flip-flop which remains set for one computer cycle (unless successive cycles are requested). This enabling signal samples the buffered MB lines into the data inputs of a D type flip-flop. At BTS2 time the data will be clocked into the Output Buffer flip-flops. Note that BTS2 can generate a strobe pulse only during a BREAK ENABLE cycle. Interface input gates are M101; output bus drivers are M623.

By careful design of the input and output gating, one register can serve as both the input and the output buffer register. Most DEC options using the data break facility have only one data buffer register with appropriate gating to allow it to serve as an output buffer when the Transfer Direction signal is at ground potential or as an input buffer when the Transfer Direction signal is +3v.

5.2.4 Memory Increment

In this type of data break the content of core memory at a device-specified address is read into the MB, is incremented by 1, and is rewritten at the same address within one 1.6-μsec cycle. This feature is particularly useful in building a histogram of a series of measurements, such as in pulse-height analysis applications. For example, in a computer-controlled experiment that counts the number of times each value of a parameter is measured, a data break can be requested for each measurement, and the measured value can be used as the core memory address to be incremented (counted).

Signal interface for a memory increment data break is similar to an output transfer data break except that the device interface generates an Increment MB signal and does not generate a strobe pulse (no data transfer occurs between the PDP-12 and the device). Timing of memory increment operations appear in Figure 5-18.

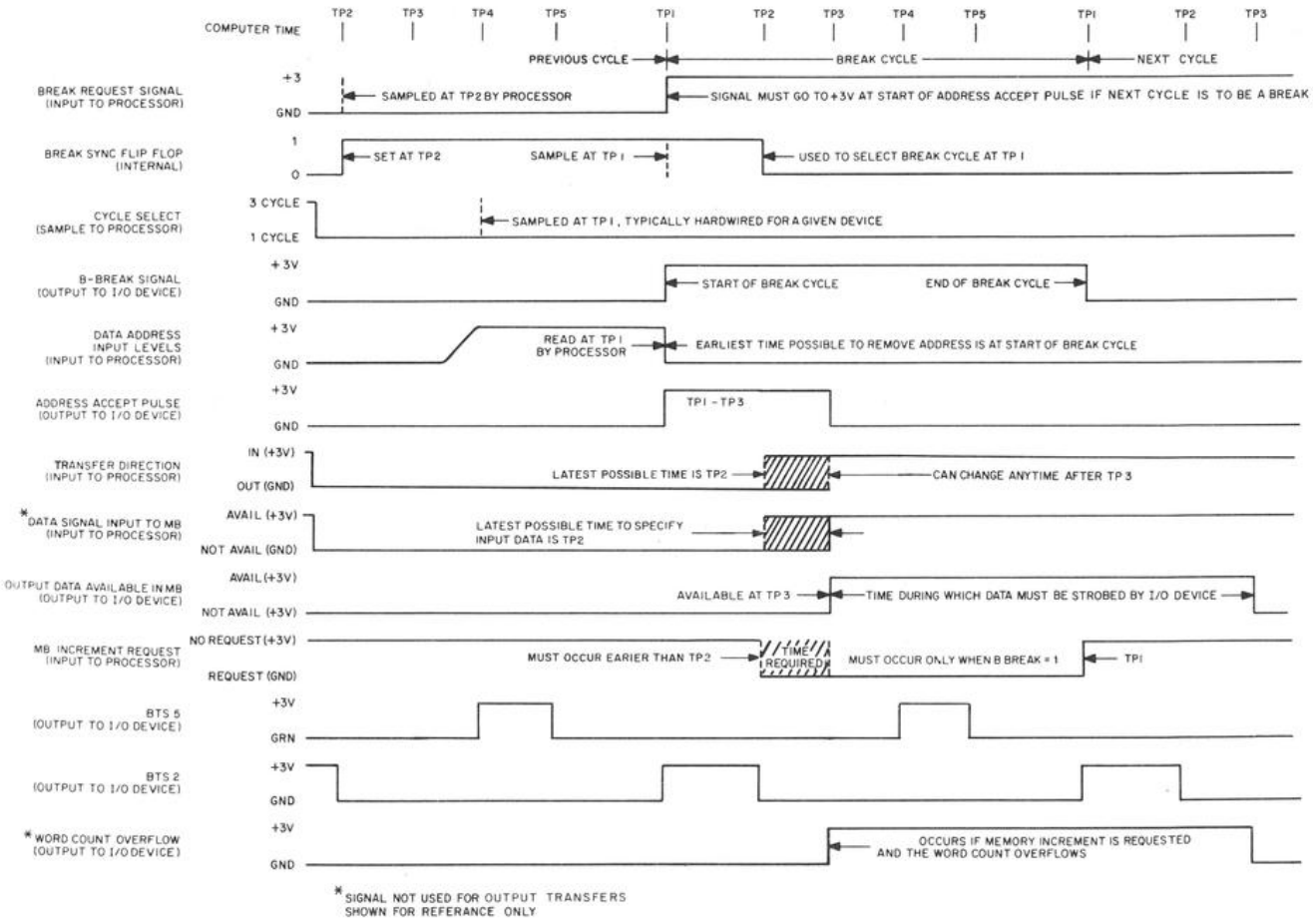


Figure 5-16. Single Cycle Data Break Output Transfer Timing Diagram

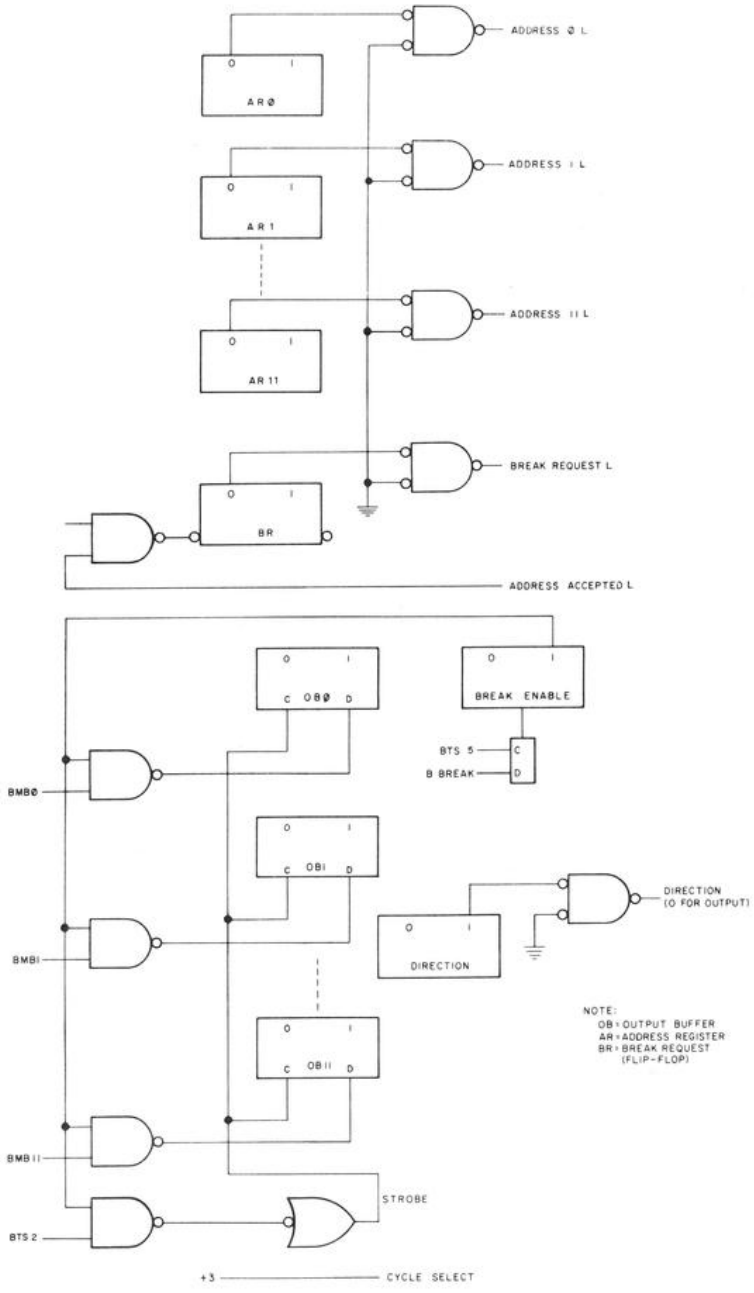


Figure 5-17. Device Interface Logic for Single Cycle Data Break Output Transfer

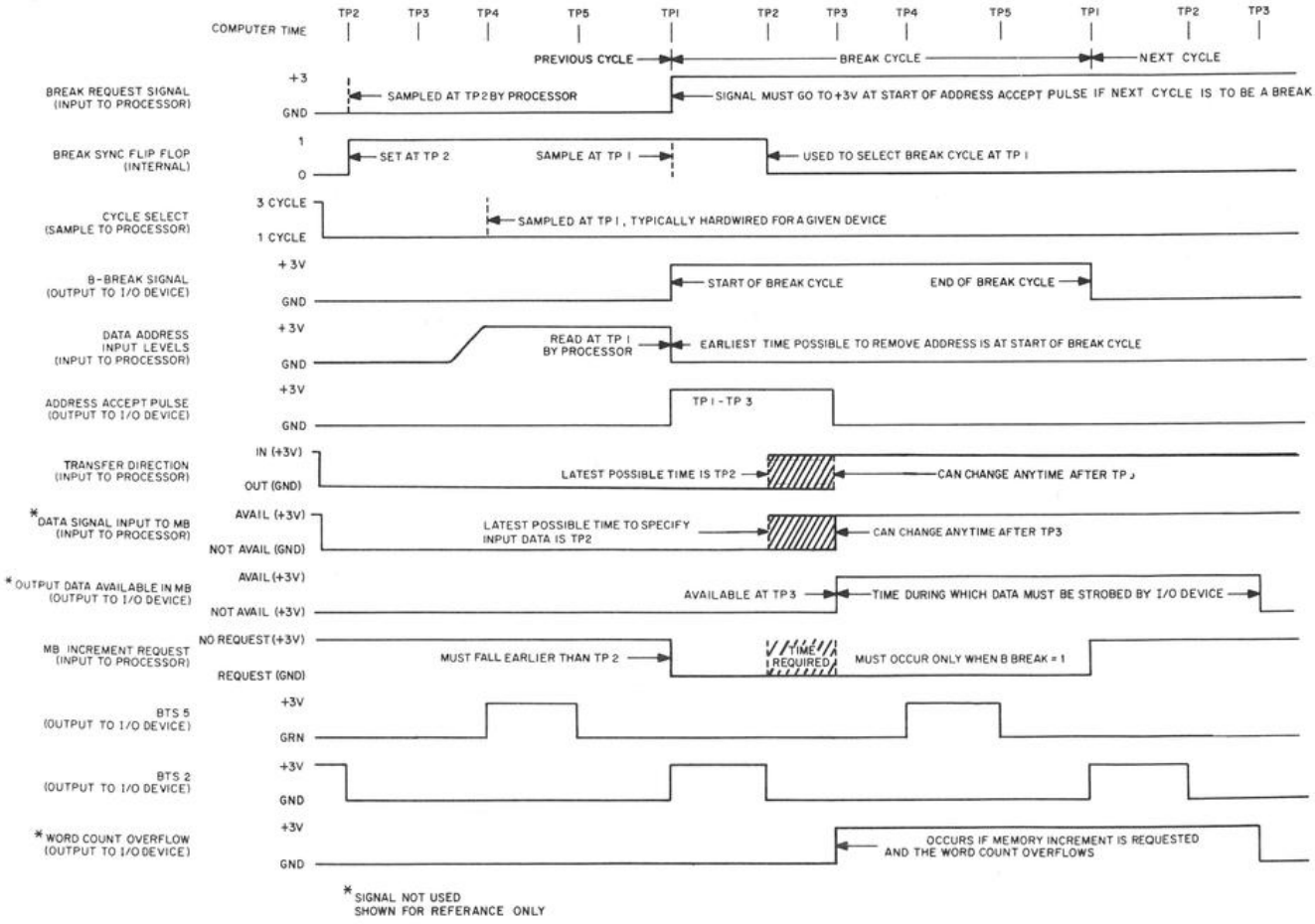


Figure 5-18. Memory Increment Data Break Timing Diagram

An interface for a device using memory increment data breaks must supply twelve Data Address signals, a Transfer Direction signal, a Cycle Select signal, and a Break Request signal to the computer data break facility as in an output transfer data break. In addition, a ground potential increment MB signal must be provided at least 250 nanoseconds before TP3 time of the Break cycle. The signal can be generated in the device interface by AND combining the B Break Computer Output signal, the output transfer condition of the Transfer Direction signal, and the Condition signal in the device that indicates that an increment operation should take place. When the computer receives this Increment MB signal, it forces the MB control element to generate a Carry Insert signal at TS3 time to increment the content of the MB.

5.2.5 Three-Cycle Data Breaks

Timing of input or output 3-cycle data breaks is shown in Figure 5-19. The 3-cycle break uses the block transfer control circuits of the computer. The block transfer control provides an economical method of controlling the flow of data at high speeds between PDP-12 core memory and fast peripheral devices, e.g., drum, disc, magnetic tape and line printers, allowing transfer rates in excess of 208 kc.

The 3-cycle data break facility provides separate current address and word count registers in core memory for the connected device, thus eliminating the necessity for flip-flop registers in the device control. When several devices are connected to this facility, each is assigned a different set of core locations for word count and current address, allowing interlaced operations of all devices as long as their combined rate does not exceed 208 kc. The device specifies the location of these registers in core memory, and thus the software remains the same regardless of what other equipment is connected to the machine. Since these registers are located in core memory, they may be loaded and unloaded directly without the use of IOT instructions. In a procedure where a device request to transfer data to or from core memory, the 3-cycle data break facility performs the following sequence of operations:

a. An address is read from the device to indicate the location of the word count register. This address is always the same for a given device; thus it can be wired in and does not require a flip-flop register.

b. The content of the specified address is read from memory and 1 is added to it before rewriting. If the content of this register becomes 0 as a result of the addition, a WC Overflow pulse will be transmitted to the device. To transfer a block of N words, this register is loaded with −N during programmed initialization of the device. After the block has been fully transferred this pulse is generated to signify completion of the operation.

c. The next sequential location is read from memory as the current address register. Although the content of this register is normally incremented before being rewritten, an increment CA inhibit (+1 → CA Inhibit) signal from the device may inhibit incrementation. To transfer a block of data beginning at location A, this register is program initialized by loading with A-1.

d. The content of the previously read current address is transferred to the MA to serve as the address for the data transfer. This transfer may go in either direction in a manner identical to the single-cycle data break system.

The 3-cycle data break facility uses many of the gates and transfer paths of the single-cycle data break system, but does not preclude the use of standard data break devices. Any combination of 3-cycle and single-cycle data break devices can be used in one system, as long as a multiplexer channel is available for each. Two additional control lines are provided with the 3-cycle data break. These are:

Word Count Overflow - A level change from GND to +3v, from TP3 to TP3, is transmitted to the device when the word count becomes equal to zero.

Increment CA Inhibit - When ground potential, this device-supplied signal inhibits incrementation of the current address word.

In summary, the 3-cycle data break is entered similarly to the single-cycle data break, with the exception of supplying a ground-level Cycle Select signal to allow entry of the WC (Word Count) state to increment the fixed

core memory location contining the word count. The device requesting the break supplies this address as in the 1-cycle data break, except that this address is fixed and can be supplied by wired ground and +3 signals, rather than from a register. Following the WC state a CA (Current Address) state is entered in which the core memory location following the WC address is read, incremented by one, restored to memory, and used as the transfer address (by MB = > MA). Then the normal B (Break) state is entered to effect the transfer.

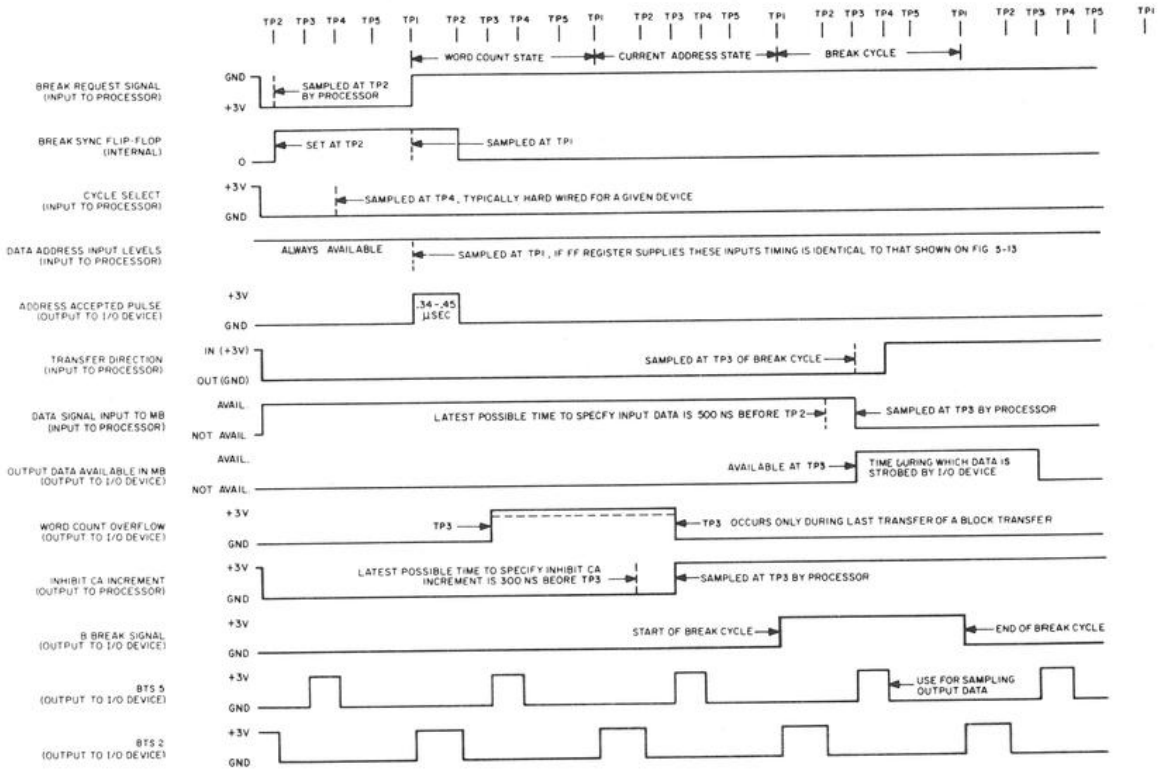


Figure 5-19. Three Cycle Data Break Timing Diagram

5.3 INTERFACE DESIGN AND CONTRUCTION

This section describes the interface modules of the PDP-12, available modules, interface conventions, and interface connections.

5.3.1 PDP-12 Interface Modules

PDP-12 interfacing is constructed of Digital FLIP CHIP modules. The Digital Logic Handbook describes more than 150 of these modules, all of their component circuits, and the associated accessories; i. e., power supplies and mounting panels. The user should study this catalog carefully before beginning the design of a special interface.

The interface modules of the PDP-12 are the M111, M906, M516, M660 and M623 modules. Interface signals to the computer use either a combination of the M111 and M906 modules or the M516 module. Interface signals from the computer will originate from a combination of M623 and M906 modules for data signals, and M660 modules for timing signals.

M111/M906 Positive Input Circuit (See Figure 5-20)

The M111 Inverter module is used in conjunction with the M906 Cable Terminator module which clamps the input to prevent excursions beyond +3 volts and ground. The M906 also provides the pullup resistors to +5 volts.

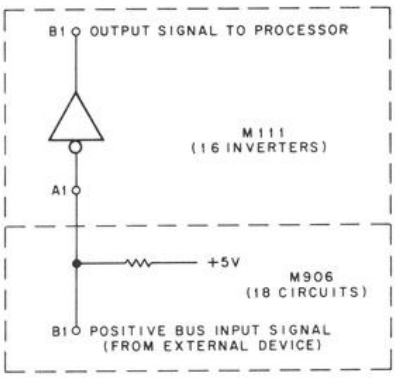


Figure 5-20. Typical M111/M906 Positive Input Circuit

M516 Positive Bus Receiver Input Circuit (See Figure 5-21)

Six four input NAND gates with overshoot and undershoot clamp on one input of each gate. Pullup resistors to +5v are also provided.

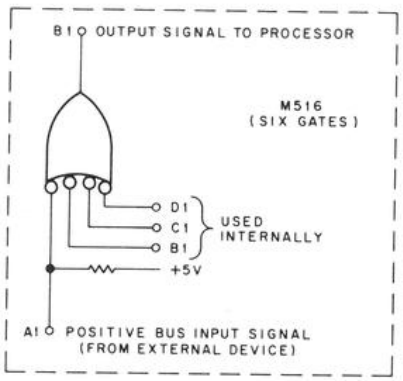


Figure 5-21. Typical M516 Positive Bus Receiver Input Circuit

M623/M906 Positive Output Circuit (See Figure 5-22)

The M623 Bus Driver module contains twelve circuits with negative NAND's. Used in conjunction with the M906 Cable Terminator module, the output is clamped to prevent excursions beyond +3 volts and ground. Output can drive +5 milliamperes at the high level and sink 20 milliamperes at the low level.

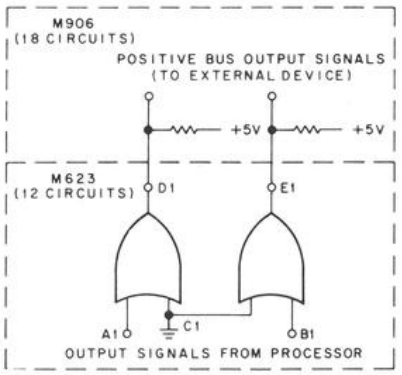
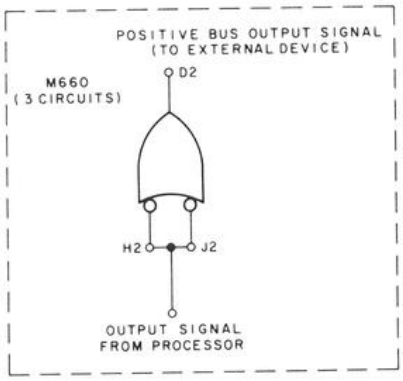


Figure 5-22. Typical M623/M906 Positive Output Circuit

M660 Bus Driver Output Circuit (See Figure 5-23)

Three circuits which provide low impedance 100 ohm terminated cable driving capability using M Series levels or pulses of duration greater than 100 nanoseconds. Output can drive + 5 ma at the high level and sink 20 ma at the low level, in addition to termination current required by the G717 termination module. The M660 module is used in the PDP-12 for the following output signals:

IOP 1, IOP 2, IOP 4, TS 2, TS5



Module Selection for Interface Circuits of Peripheral Equipment

Two FLIP CHIP modules are of particular interest in the design of equipment to interface to the PDP-12. Complete details on these and other FLIP CHIP modules can be found in the Digital Logic Handbook.

M103 Device Selector (See Figure 5-24)

The M103 selects an input/output device according to the code in the instruction word (being held in the memory buffer during the IOT cycle). M103 module includes diode protection clamps on input lines so that it may be used directly on the PDP-12 positive bus.

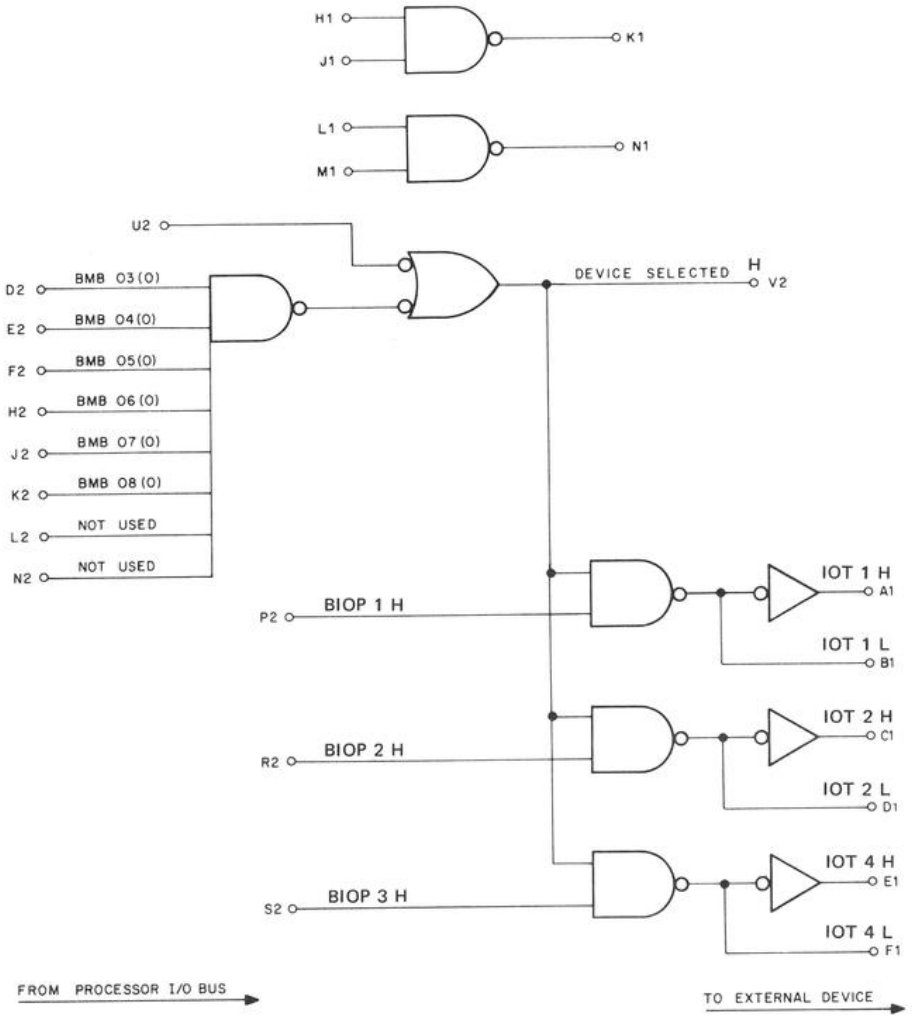


Figure 5-24. M103 Device Selector Logic Circuit

M101 Bus Data Interface (See Figure 5-25)

Fifteen two-input NAND gates with one input of each gate tied to a common line. For use in strobing data off of the PDP12 I/O bus. M101 module includes diode protection clamps on input lines so that it may be used directly on the PDP-12 positive bus.

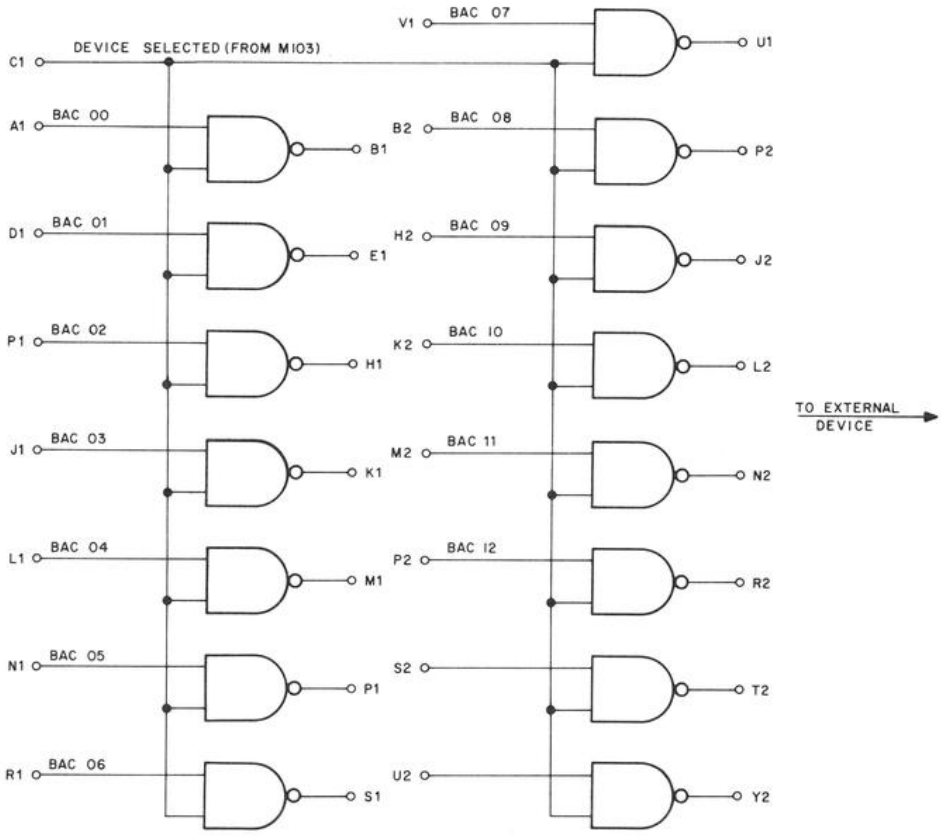


Figure 5-25. M101 Bus Data Interface Logic Circuit

5.3.2 M Series Flip Chip Modules

The following is a list of M Series modules available from Digital Equipment Corporation that can be used in designing special interfaces and special devices. The majority of these modules are described in the Digital Logic Handbook. For those that cannot be found in the Handbook, contact the nearest Digital representative.

M Series Module Summary

Type		*Description*
M002	15 Loads	Fifteen +3 volt sources each capable of driving ten unit loads. Can be used for typing off unused inputs.
M040	Solenoid Driver	Output ratings of -70 volts and 0.6 amp allow these two drivers to be used with a variety of medium current loads.
M050	Indicator Driver	Output ratings of -20 volts and 50 ma. Allow any of the twelve circuits on this module to drive a variety of incandescent lamps. These drivers can also be used as slow speed open collector PNP level shifters to -3 volt systems.
M101	Bus Data Interface	Fifteen two-input NAND gates with one input of each gate tied to a common line. For use in strobing data off of the PDP-8/I or PDP-12 I/O bus. Pin compatible with M111.
M103	Device Selector	Similar to W103 only for use with PDP-8/I and PDP-12 options. Output pulses are not regenerated but only buffered.
M111	Inverters	Sixteen inverter circuits with a fan-in of one unit load and fan-out of ten unit loads.
M112	NOR Gates	Ten positive NOR gates with a fan-in of one unit load and fan-out of ten unit loads.
M113	NAND Gates	Ten two input positive NAND gates with a fan-in of one unit load and fan-out of ten unit loads.
M115	NAND Gates	Eight three-input positive NAND gates with a fan-in of one unit load and a fan-out of ten unit loads.
M117	NAND Gates	Six four-input positive NAND gates with a fan-in of one unit load and a fan-out of ten unit loads.
M119	NAND Gates	Three eight-input positive NAND gates with a fan-in of one unit load and a fan-out of ten unit loads.
M121	AND/NOR Gates	Six gates which perform the positive logic function AB + CD. Fan-in on each input is one unit load and gate fan-out is ten unit loads.

M141	NAND/OR Gates	Twelve two input positive NAND gates which can be used in a wired OR manner. Gates are grouped in a 4-4-3-1 configuration with a fan-in of one unit load and a fan-out which depends on the number of gates ORed together.
M160	AND/NOR GATES	Three general purpose multi-input gates which can be used for system input selection. Fan-in is one unit load and fan-out is ten unit loads.
M161	Binary to Octal/ Decimal Decoder	A binary to eight line or BCD to ten line decoder. Gating is provided so that up to six binary bits can be decoded using only M161's. Accepts a variety of BCD codes.
M162	Parity Circuit	Two circuits each of which can be used to generate even or odd parity signals for four bits of binary input.
M169	Gating Module	Four circuits which can be used for input selection. Each circuit is of an AND/OR configuration with four two input AND gates.
M202	Triple J-K Flip-Flop	Three J-K flip-flops with multiple input AND gates on J and K. Versatile units for many control or counter purposes. All direct set and clear inputs are available at module pins.
M203	Set-Reset-Flip-Flops	Eight single-input set-reset flip-flops for use as buffer storage. Each circuit has a fan-in one unit load and a fan-out of ten unit loads.
M204	Genera Purpose Buffer and Counter	Four J-K flip-flops which can be interconnected as a ripple or synchronous counter or used as general control elements.
M206	General Purpose Flip-Flops	Six D type flip-flops which can be used in shift registers counters, buffer registers and general purpose control functions.
M207	General Purpose Flip-Flops	Six single input J and K type flip-flops for use in shift-registers, ripple counters, and general purpose control functions.
M208	8-Bit Buffer/	An internally connected 8-bit buffer or shift register. Provisions are made for gated single-ended parallel load, bipolar parallel output, and serial input.
M211	Binary Up/Down Counter	A six bit binary up/down ripple counter with control gates for direction changes via a single control line.
M212	Left-Right Shift Register	An internally connected left-right shift register. Provisions are made for gated single-ended parallel load, bipolar parallel output, and serial input.

M213	BCD Up/Down Counter	One decade of 8421 up or down counting is possible with this module. Provisions are made for parallel loading, bipolar output and carry features.
M230	Binary to BCD Shift Register Converter	One decade of a modified shift register which allows high speed conversion (100 nsec per binary bit) of binary data to 8421 BCD code. System use of this module requires additional modules.
M302	Dual Delay Multivibrator	Two pulse or level triggered one-shot delays with output delay adjustable from 50 nsec to 7.5 msec. Fan-in is 2.5 unit loads and fan-out is 25 unit loads.
M310	Delay Line	Fixed tapped delay line with delay adjustable in 50 nsec increments from 50 nsec to 500 nsec. Two digital output amplifiers and one driver are included.
M360	Variable Delay	Continuously variable delay line with a range of 50 nsec to 500 nsec Module includes delay line drivers and digital output amplifiers.
M401	Variable Clock	A gateable RC clock with both positive and negative pulse outputs. The output frequency is adjustable from 10mHz tobe-low 100 Hz.
M405	Crystal Clock	Stable system clock frequencies from 5KHz to 10MHz are available with this module. Frequency drift at either the positive or negative pulse output is less than .01% of the specified frequency.
M410	Resonant Reed	A stable low frequency reed control clock similar to the M452. Stability in the range 0°C to 70°C is better than 0.15%. For use with communications systems and available with only standard teletype and data set frequencies.
M452	Teletype Clock	Provides 880Hz, 440Hz, and 220Hz square waves necessary for clocking and M706 and M707 in a 110 band teletype system.
M501	Schmitt Trigger	Provides a regenerative characteristics necessary for switch filtering, pulse shaping, and contact closure sensing. This circuit can be AND/OR expanded.
M502	Negative Input Converter	Pulses as short as 35 nsec can be level shifted from -3volt systems to standard M Series levels by the two circuits in this converter. This module can also drive low impedance terminated cables.

M506	Negative Input Converter	This converter will level shift pulses as short as 100 nsec from -3volt systems to M Series levels. Each of the six circuits on this module provide a low impedance output for driving unterminated long lines.
M507	Bus Converter	Six inverting level shifters which accept -3 and GRD, as inputs and have an open collector NPN transistor at the output. Output rise is delayed by 100 nsec for pulse spreading.
M516	Positive Bus Receiver	Six four input NAND gates with overshoot and undershoot clamps on one input of each gate. In addition, one input of each gate is tied to +3 volts with the lead brought out to a connector pin.
M602	Pulse Amplifier	The Two pulse amplifiers in this module provide standard 50 nsec or 110 nsec pulses for M Series systems.
M617	Four Input Power NAND Gate	Six four-input positive NAND gates with a fan-in of one unit load are a fan-out of 30 unit loads.
M623	Bus Driver	Twelve circuits organized in a manner similar to two R123 gates. Input gates are negative NAND's and the open collector NPN outputs can drive 100 ma at ground.
M627	NAND Power Amplifier	Six four-input high speed positive NAND gates with a fan-in of 2.5 unit loads and a fan-out of 40 unit loads.
M650	Negative Output Converter	The three non-inverting level shifters on this module can be used to interface the positive levels or pusles (duration greater than 100 nsec) of K and M Series to -3 volt logic systems.
M652	Negative Output Converter	These two circuits provide high-speed non-inverting level shifting for pulses as short as 35 nsec or levels from M Series to -3 volt systems. The output can drive low impedance terminated cables.
M660	Bus Driver	Three circuits which provide low impedance 100 ohm terminated cable driving capability using M Series levels or pulses of duration greater than 100 nsec. Output drive capability is 50 ma at +3 volts or ground.
M661	Positive Level Driver	Three circuits which provide low impedance unterminated cable driving. Characteristics are similar to M660 with the exception that +3 volts drive is 5 ma.
M730	8/I Bus Positive Output Interfacer	General Purpose positive bus output module for use in interfacing many positive level (0 to +20 volt) systems to the PDP-8/I or PDP-12. Module includes device selector, 12 bit parallel output buffer and adjustable timing pulses.

M731	8/I Bus Negative Output Interfacer	Identical to M730 except outputs are level shifted for 0 to -20 volt) systems to the PDP-8/I or PDP-12. Module includes device selector, 12 bit parallel input buffer, and adjustable timing pulses.
M733	8/I Bus Negative	Identical to M732 except inputs are level shifted from negative voltage systems.
M901	Flexprint Cable	Double-sided 36 pin flexprint cable connector. All pins are available for signals or grounds. Pins A2, B2, U1, and V1 have 10 Ω resistors in series.
M902	Resistor Termintor	Double-sided 36 pin terminator module with 100 Ω terminations on signal leads. Alternate ground are provided as in the M903 and M904.
M903	Connector	Double-sided 36 pin flexprint cable connector with alternate grounds for I/O bus cables.
M906	Cable Terminator	18 load resistors clamped to prevent excursions beyond +3 volts and ground. It may be used in conjunction with the M623 to provide cable driving ability.

5.3.3 **Construction of Interfaces**

This section will provide the interface designer with additional information on design procedures, module layout, wiring, and cable selection. Additional help may be obtained from local DEC sales offices.

Physical

The PDP-12 was designed to provide the user maximum ease and flexibility in implementing special interfaces. External devices and interfaces are constructed and mounted outside of the basic machine, thereby eliminating the necessity for modifications to the basic processor. All signals to and from the computer are carried on coaxial or flexprint cables.

To implement several devices, the cables parallel connect each peripheral in a serial type form (see Figure 5-26). Three dual cables are used for program interrupt cable connections in (or out). Two additional dual cables are used for a total of five, when Data Break devices are implemented.

Module Layout

In general, module layout is done based on the functional elements within a system and is primarily a matter of common sense.

Digital has, however, layout conventions for I/O cabling to extend devices. The interface designer may wish to use these conventions as a guide. The general rule is DO NOT DEAD END THE I/O BUS. This means that parallel connections should always be made at each device to handle possible future expansion.

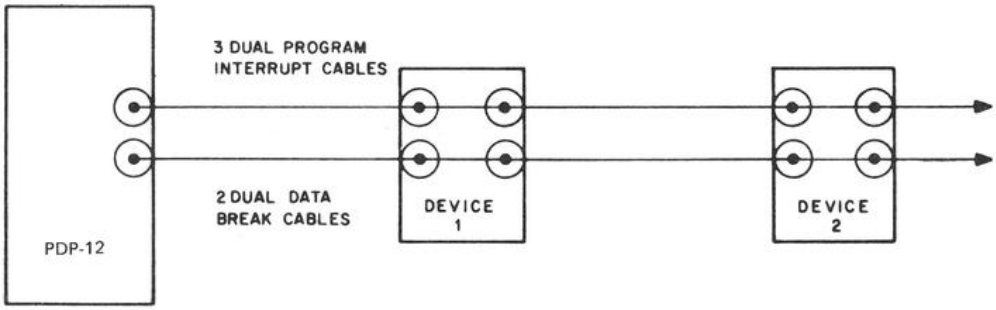


Figure 5-26. I/O Bus Configuration

Figure 5-27 shows the I/O cable connections in an option mounting panel. Module slot locations 1 through 3 (looking at the wiring pin side) in an option mounting panel are reserved for program transfer cable connections in (or out). Module slot locations 4 to 5 are reserved for Data Break cable connections in (or out). Slot 6 is used for Sense lines.

Module slot locations 1 through 6 in the bottom half of the option mounting panel are wired in parallel with the top module slot locations 1 through 6. To continue the I/O cabling to the next device, the bottom slots are used and the I/O cable connections are exactly the same as mentioned above.

Cable Selection

Two types of cables are recommended for I/O interface connections.

One is 9 conductor coax cable. This cable protects systems from radiated noise and cross talk between individual lines. Coax cable used and sold by Digital has the following nominal specs:

Z_o	=	0.095 ohm/foot nominal
C	=	95 ± 5
L	=	13.75 pf/foot approx. (unterminated)
R	=	124 Nhy/foot approx.
Y	=	79% of velocity of light, approx. (1.5 nsec/ft.)

The other cable is 19 conductor (9 signals and 10 grounds), # 30 gauge flat copper flexprint cable.

The total length of I/O cabling, from the PDP-12 to the last device, can be a maximum of 50 feet. This can be 50 feet of coax or a combination of coax and flexprint, in which case the flexprint cannot exceed a total of 15 feet.

Connector Selection - Of the many connectors available in the module product line, several have particular application to I/O connectors. Price and ordering information is available on these and other connectors in the Digital Logic Handbook. Of particular interest are the M903 and M904 connectors described in the subsequent paragraphs.

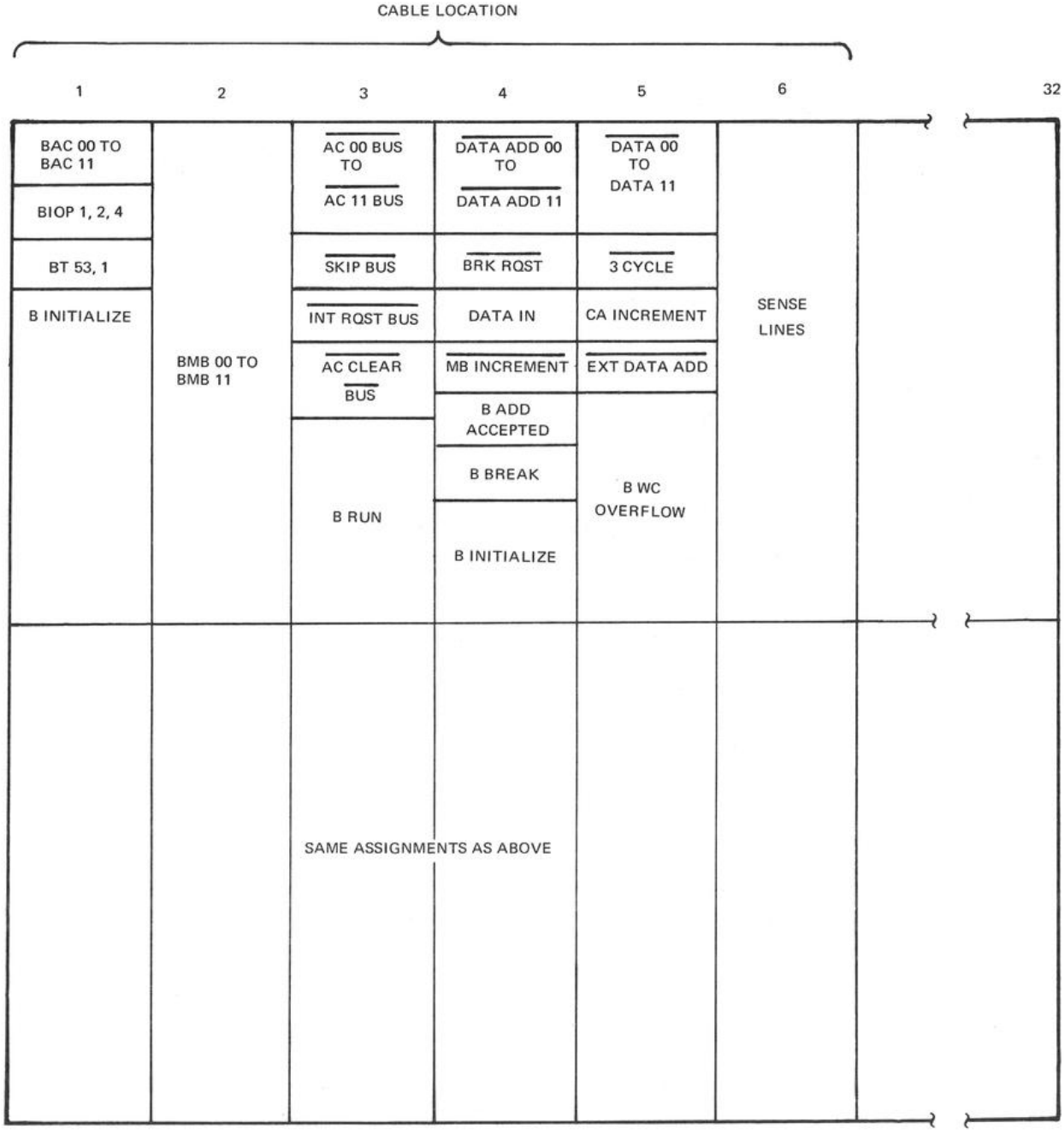


Figure 5-27. I/O Cable Connections

a. *M903 Connector* - Double-sided 36 pin flexprint cable connector with alternate grounds for I/O bus cables. (Two flexprint cables are utilized with this connector module.)

b. *M904 Connector* - Double-sided 36 pin coaxial cable connector with alternate grounds for I/O bus cables. (Two coax cables are utilized with this connector module).

(1) *Signals*:

B1, D1, E1, H1, J1, L1, M1, P1, S1,
D2, E2, H2, K2, M2, P2, S2, T2, V2

(2) *Grounds*

A1, C1, F1, K1, N1, R1, T1,
C2, F2, J2, L2, N2, R2, U2

Signal Terminating - The G717 module is used for terminating the following signals:

IOP 1, IOP 2, IOP 4, TS 2, TS 5.

This module contains five 100 ohm terminating resistors and should be located in the last device of the I/O cabling scheme.

Wiring Hints - These suggestions may help reduce mounting panel wiring time. They are not intended to replace any special wiring instructions given on individual module data sheets or in application notes. For fastest and neatest wiring, the following order is recommended.

1) All power wiring (Pins A2, B2, C2, T1) and any horizontally bussed signal wiring. Use Horizontal Bussing Strips, Type 933. (Pin B2 is bussed with -15V for modules requiring -15V).

2) Vertical grounding wires interconnecting each chassis ground with pins C2 & T1 grounds. Start these wires at the uppermost mounting panel and continue to the bottom panel. On the first and last blocks of the mounting panel, connected the grounds to the chassis.

(3) All other ground wires. Always use the nearest ground pin, unless a special grounding pin has been provided in the module.

(4) Wire all signal wires in convenient order. Point-to-point wiring produces the shortest wire lengths, goes in the fastest, is easiest to trace and change, and generally results in better appearance and performance than cabled wiring. Point-to-point wiring is strongly urged.

The recommended wire size for use with the H803 mounting blocks and H911 mounting panel is #30. Larger or smaller wire may be used depending on the number of connections to be made to each lug. Solid wire and a heat resistant insulation (Kynar) is recommended. The H803 mounting blocks are only available with wire wrap pins which necessitates the use of a wire wrap tool. (Digital can supply #30 gauge wire in 1000 foot rolls.)

Adequate grounding is essential. In addition to the connections between mounting panels mentioned above, there must be continuity of grounds between cabinets and between the logic assembly and any equipment with which the logic communicates.

When wire wrapping is done on a mounting panel containing modules, the wire wrap tool should be grounded except when all modules are removed from the mounting panel. This procedure should be followed, because even with completely isolated tools, such as those operated by batteries or compressed air, a static charge can often build up and burn out semiconductors.

Cooling - The low power consumption of M Series modules results in a total of about 15 watts dissipation in a typical H911 mounting panel with 64 modules. Convection cooling is sufficient for a few mounting panels, but forced air cooling should be used when a very large system is built.

5.3.4 **IOT Allocations**

IOT	*OPTION*
00	Interrupt
01	High Speed Reader Type PR12
02	High Speed Punch Type PP12
03	Teletype Keyboard/Reader
04	Teletype Teleprinter/Punch
05	Displays, Types VC8/I and KV8/I
06	Displays, Types VC8/I and KV86I
07	Displays, Types VC8/I, and Light Pen Type 370
10	Power Fail Option KP8/L
11	Teletype System Type PT08
12	Teletype System Type PT08
13	Rea Time Clock Type KW12
14	
15	
16	
17	
20	Memory Extension Control Option Type MC12
21	Memory Extension Control Option Type MC12
22	Memory Extension Control Option Type MC12
23	Memory Extension Control Option Type MC12
24	Memory Extension Control Option Type MC12
25	Memory Extension Control Option Type MC12
26	Memory Extension Control Option Type MC12
27	Memory Extension Control Option Type MC12
30	User Interfaces
31	User Interfaces
32	User Interfaces
33	User Interfaces
34	User Interfaces
35	User Interfaces
36	User Interfaces
37	User Interfaces
40	Teletype System Type PT08
41	Teletype System Type PT08
42	Teletype System Type PT08
43	Teletype System Type PT08
44	Teletype System Type PT08

TERMINAL PORT

45	Teletype System Type PT08
46	Teletype System Type PT08
47	Teletype System Type PT08
50	Incremental Plotter Type XY12
51	Incremental Plotter Type XY12
52	Incremental Plotter Type XY12
53	General Purpose A/D Converters and Multiplexers, Types AF01A, AM02A, AMO3A and AF04A Scanning Digital Voltmeter
54	General Purpose A/D Converters and Multiplexers, Types AF01A, AM02A, AM03A and AF04A Scanning Digital Voltmeter
55	D/A Converter Type AA01A
56	D/A Converter Type AA01A
57	D/A Converter Type AA01A, Sample and Hold Control Type AC01A and AF04A Scanning Digital Voltmeter
60	Random Access Disk File and Control Type DF32 and Synchronous Modem Interface Type DP01A
61	Random Access Disk File and Control Type DF32 and Synchronous Modem Interface Type DP01A
62	Random Access Disk File and Control Type DF32 and Synchronous Modem Interface Type DP01A
63	Card Reader Type CR12
64	Synchronous Modem Interface Type DP01A
65	Synchronous Modem Interface Type DP01A
66	Synchronous Modem Interface Type DP01A
67	Card Reader Type CR12 and Synchronous Modem Interface Type DP01A
70	Automatic Mag Tape Type TC58
71	Automatic Mag Tape Type TC58
72	Automatic Mag Tape Type TC58
73	Automatic Mag Tape Type TC58
74	Automatic Mag Tape Type TC58
75	
76	DECtape Control TC01
77	DECtape Control TC01

5.3.5 **Interface Connections**

All interface connections to the PDP-12 are made at assigned module receptacle connectors in the Processor Mounting Frame. Capital letters designate vertical rows of modules within a mounting frame. The letters increase from right to left when viewed from the wiring side. Module receptacles are numbered from top to bottom within a row. Terminals are assigned capital letters from right to left with the letters G, I, O, and Q omitted. Double sided connectors or modules are used with the suffic number *1* used to designate the top side of a module and the suffix number *2* used to designate the bottom side.

The module receptacles and assigned use for interface signal connections are:

Receptacle

N13	SENSE LINES
N14	AC, IOP, TIMING OUTPUTS

N15	MB OUTPUTS
N16	AC, SKIP, INT. REQUEST INPUTS
N17	DATA BREAK ADDRESS INPUTS
N18	DATA BREAK DATA INPUTS

Terminals A1, C1, F1, K1, N1, R1, T1, C2, F2, J2, L2, N2, R2, and U2 of these receptacles are grounded within the computer and terminals B1, D1, E1, H1, J1, L1, M1, P1, S1, D2, E2, H2, K2, M2, P2, S2, T2, and V2 carry signals. Terminal A2 is +5 VDC and Terminal B2 is -15 VDC. These terminals mate with either M903 or M904 Cable Connectors.

Interface connection to the PDP-12 can be established for all peripheral equipment by making series cable connections between devices. In this manner only one set of cables is connected to the computer and two sets are connected to each device: one receiving the computer connection from the computer itself or the previous device, and one passing the connection to the next device. Where physical location of equipment does not make series bus connections feasible, or when cable length becomes excessive, additional interface connectors can be provided near the computer. All logic signals passing between the PDP-12 and input/output equipment are positive voltage levels, allowing direct TTL logic interface with appropriate diode clamp protection.

Positive level for a low logic state is 0 to 0.4 volts. Positive level for a high logic state is +2.4 to +3.6 volts.

The following table present cable connections to the PDP-12 I/O Bus. A signal is true when its polarity matches the suffix character of its name (i.e., I00 BAC 00 (1) H will be high when AC 00 (1) and a program interrupt will be requested when the line EXT INT RQST BUS L is pulled low.)

SIGNAL	CONNECTION	SIGNAL	CONNECTION
IOB XL OO H	N13B1	IOB XL 09 H	N13D2
IOB XL O1 H	N13D1	IOB XL 10 H	N13E2
IOB XL 02 H	N13E1	~~NOT USED~~	N13H2
IOB XL 03 H	N13H1	NOT USED	N13K2
IOB XL 04 H	N13J1	NOT USED	N13M2
IOB XL 05 H	N13L1	NOT USED	N13P2
IOB XL 06 H	N13M1	NOT USED	N13S2
IOB XL 07 H	N13P1	NOT USED	N13T2
IOB XL 08 H	N13S1	NOT USED	N13V2
IOO BAC 00 (1) H	N14B1	IOO BAC 09 (1) H	N14D2
IOO BAC 01 (1) H	N14D1	IOO BAC 10 (1) H	N14E2
IOO BAC 02 (1) H	N14E1	IOO BAC 11 (1) H	N14H2
IOO BAC 03 (1) H	N14H1	IOO BIOP 1 H	N14K2
IOO BAC 04 (1) H	N14J1	IOO BIOP 2 H	N14M2
IOO BAC 05 (1) H	N14L1	IOO BIOP 4 H	N14P2
IOO BAC 06 (1) H	N14M1	IOO BTS 5 (1) H	N14S2
IOO BAC 07 (1) H	N14P1	IOO BTS 2 (1) H	N14T2
IOO BAC 08 (1) H	N14S1	IOO BA INITIALIZE H	N14V2

SIGNAL	CONNECTION	SIGNAL	CONNECTION
IOO BMB 00 (1) H	N15B1	IOO BMB 06 (0) H	N15D2
IOO BMB 01 (1) H	N15D1	IOO BMB 06 (1) H	N15E2
IOO BMB 02 (1) H	N15E1	IOO BMB 07 (0) H	N15H2
IOO BMB 03 (0) H	N15H1	IOO BMB 07 (1) H	N15K2
IOO BMB 03 (1) H	N15J1	IOO BMB 08 (0) H	N15M2
IOO BMB 04 (0) H	N15L1	IOO BMB 08 (1) H	N15P2
IOO BMB 04 (1) H	N15M1	IOO BMB 09 (1) H	N15S2
IOO BMB 05 (0) H	N15P1	IOO BMB 10 (1) H	N15T2
IOO BMB 05 (1) H	N15S1	IOO BMB 11 (1) H	N15V2
EXT IO BUS 00 L	N16B1	EXT IO BUS 09 L	N16D2
EXT IO BUS 01 L	N16D1	EXT IO BUS 10 L	N16E2
EXT IO BUS 02L	N16E1	EXT IO BUS 11 L	N16H2
EXT IO BUS 03L	N16H1	EXT SKIP BUS L	N16K2
EXT IO BUS 04L	N16J1	EXT INT RQST BUS L	N16M2
EXT IO BUS 05L	N16L1	EXT AC CLEAR BUS L	N16P2
EXT IO BUS 06 L	N16M1	IOO B RUN (0) H	N16S2
EXT IO BUS 07 L	N16P1	NOT USED	N16T2
EXT IO BUS 08 L	N16S1	NOT USED	N16V2
EXT DATA ADD 00 L	N17B1	EXT DATA ADD 09L	N17D2
EXT DATA ADD 01 L	N17D1	EXT DATA ADD 10 L	N17E2
EXT DATA ADD 02 L	N17E1	EXT DATA ADD 11 L	N17H2
EXT DATA ADD 03 L	N17H1	EXT BREAK RQST L	N17K2
EXT DATA ADD 04 L	N17J1	EXT DATA IN H	N17M2
EXT DATA ADD 05 L	N17L1	IOO BREAK (0) H	N17P2
EXT DATA ADD 06 L	N17M1	IOO ADD ACCEPTED (0) H	N17S2
EXT DATA ADD 07 L	N17P1	EXT INCREMENT MB L	N17T2
EXT DATA ADD 08 L	N17S1	IOO BB INITIALIZE H	N17V2
EXT DATA 00 L	N18B1	EXT DATA 09 L	N18D2
EXT DATA 01 L	N18D1	EXT DATA 10 L	N18E2
EXT DATA 02 L	N18E1	EXT DATA 11 L	N18H2
EXT DATA 03 L	N18H1	EXT 3 CYCLE L	N18K2
EXT DATA 04 L	N18J1	IOB CA INCREMENT H	N18M2
EXT DATA 05 L	N18L1	IOO WC OVERFLOW (0) H	N18P2
EXT DATA 06 L	N18M1		N18S2
EXT DATA 07 L	N18P1	EXT DATA ADD 02 L	N18T2
EXT DATA 08 L	N18S1	EXT DATA ADD 01 L	N18V2
		EXT DATA ADD 00 L	

Calcomp Memory N19
M-617 Board

5.4 STANDARD I/O BUS PERIPHERALS

This section contains description for a number of standard I/O Bus peripherals which are available on the PDP-12. The ASR33, DP12, KW12, and XY12 are previewed in the basic PDP-12. The LP12, PR12, PP12, PC12, and CR12 are available in the BA12 option expander. Other options contain their own logic and mounting hardware.

5.4.1 Teletype Model 33 ASR and Control

The standard Teletype Model 33 ASR (automatic send-receive) can be used to type in or print out information at a rate of up to ten characters per second, or to read in or punch out perforated paper tape at a ten characters per second rate. Signals transferred between the 33 ASR and the control logic are standard serial, 11 unit code Teletype signals. The signals consist of marks and spaces which correspond to idle and bias curren in the Teletype, and to zeros and ones in the control and computer. The start mark and subsequent eight character bits are one unit of time duration and are followed by the stop mark which is two units. The control circuitry for this device is located in the PDP-12 central processor.

The 8-bit code used by the Model 33 ASR Teletype unit is the American Standard Code for Information Interchange (ASCII) modified. To convert the ASCII code to Teletype code add 200 octal (ASCII + 200_8 = Teletype). This code is read in the reverse of the normal octal form used in the PDP-12 since bits are numbered from right to left, from 1 through 8, with bit 1 having the least significance. Therefore perforated tape is read:

8 7 6 5 4 S 3 2 1

Most Significant Octal Bit — Least Significant Octal Bit

The Model 33 ASR set can generate all assigned codes except 340 through 374 and 376. Generally codes 207, 212, 215, 240 through 337, and 377 are sufficient for Teletype operation. The Model 33 ASR set can detect all characters, but does not interpret all of the codes that it can generate as commands. The standard number of characters printed per line is 72. The sequence for proceeding to the next line is a carriage return followed by a line feed (as opposed to a line feed followed by a carriage return). Appendix 3 lists the character code for the Teletype. Punched tape format is as follows:

Tape Channel	8 7 6 5 4 S 3 2 1
Binary Code (1 = Punch)	1 0 1 1 0 1 0 0
Octal Code	2 6 4

Teletype Control

Serial information read or written by the Teletype unit is assembled or disassembled by the control for parallel transfer to the accumulator of the processor. The control also provides the program flags whcih cause a program interrupt or an instruction skip based upon the availability of the Teletype and the processor as a function of the program.

In all programmed operation, the Teletype unit and control are considered as a Teletype in (TTI) as a source of input intelligence from the keyboard or the perforated-tape reader and is considered a Teletype out (TTO) for computer output information to be printed and/or punched on tape. Therefore, two device selectors are used, the

select code of 03 initiates operations associated with the keyboard/reader, and the device selector, assigned the select code of 04, performs operations associated with the teleprinter/punch. Parallel input and output functions are performed by corresponding IOT pulses produced by the two device selectors. Pulses produced by IOP1 pulse trigger skip gates; pulses produced by the IOP2 pulse clear the control flags and/or the accumulator; and pulses produced by the IOP4 pulse initiate data transfers to or from the control.

Keyboard Reader

The keyboard and tape reader control contains an 8-bit buffer (TTI) which assembles and holds the code for the last character struck on the keyboard or read from the tape. Teletype characters from the keyboard/reader are received serially by the 8-bit shift register TTI. The code of a teletype character is loaded into the TTI so that spaces correspond with binary zeros and holes (marks) correspond to binary ones. Upon program command the content of the TTI is transferred in parallel to the accumulator.

When a Teletype character starts to enter the TTI the control de-energizes a relay in the Teletype unit to release the tape feed latch. When released, the latch mechanism stops tape motion only when a complete character has been sensed, and before sensing of the next character is started.

A keyboard flag is set to a binary one, and causes a program interrupt when an 8-bit computer character has been assembled in the TTI from a TEletype character. The program must sense the condition of this flag with a KSF microinstruction, and if the flag is set, issue a KRB microinstruction which clears the AC, clears the keyboard flag, transfers the content of the TTI into the AC, and enables advance of the tape feed mechanism.

Instructions for use in supplying data to the computer from the Teletype are:

KSF Skip on Keyboard Flag

Octal Code: 6031
Event Time: 1
Execution Time: 4.25 μsec
Operation: The keyboard flag is sensed, and if it contains a binary 1 the content of the PC is incremented by one so that the next sequential instruction is skipped.
Symbol: If Keyboard Flag = 1, then PC + 1 = > PC

KCC Clear Keyboard Flag

Octal Code: 6032
Event Time: 2
Execution Time: 4.25 μsec
Operation: The AC is cleared in preparation for another microinstruction to transfer a character from the TTI into the AC. The keyboard flag is also cleared, this allows the hardware to begin assembling the next input character in the TTI. If there is tape in the reader and the reader is on, the character over the read head will be loaded into the TTI and the tape advanced one frame. If there is no tape or the reader is turned off (STOP or FREE) the character struck on the keyboard will be assembled into the TTI. In either case, when the character is completely assembled in the TTI the hardware causes the keyboard flag to be set to a binary 1.

Symbol: 0 = > AC
O = > Keyboard flag allowing the hardware to cause:
Keyboard/Tape Character = > TTI
1 > Keyboard flag when done

KRS Read Keyboard Buffer Static

Octal Code: 6034
Event Time: 3
Execution Time: 4.25 μsec
Operation: The content of the TTI is transferred into bits 4 through 11 of the AC. This is a static command in that neither the AC nor the keyboard flag is cleared.
Symbol: TTI V AC_{4-11} => AC_{4-11}

KRB Read Keyboard Buffer Dynamic

Octal Code: 6036
Event Time: 2, 3
Execution Time: 4.25 μsec
Operation: This microinstruction combines the functions of the KCC and KRS. The AC and keyboard flag are both cleared and the content of the TTI is transferred into bits 4-11 of the AC. Clearing the keyboard flag allows the hardware to begin assembling the next input character into the TTI (as discussed with the KCC). When the character is completely assembled in the TTI, the hardware causes the flag to be set indicating it again has a character ready for transfer.

Symbol: 0 => AC C(TTI) V $C(AC_{4-11})$ => AC_{4-11}
0 => Keyboard Flag allowing the hardware to cause:
Keyboard/Tape Character => TTI
1 => Keyboard flag when done.

The following are examples of possible sequences of instruction to read a character into the AC from the teletype:

```
                .
                .
LOOK,           KSF             /SKIP IF FLAG = 1
                JMP LOOK        /JMP BACK & TEST FLAG AGAIN
                KRB             /TRANSFER TTI CONTENTS INTO AC
```

This sequence waits for the TTI to set its flag, indicating that it has a character ready to be transferred. It then skips to the KRB command which causes the character to be read into the AC from the TTI.

By making this sequence of instructions a subroutine of a larger program, it can be accessed each time an input character is desired.

```
                KCC             /CLEAR TTI FLAG
                .
                .
                .
READ,           0               /STORE DC HERE FOR RETURN ADDRESS
                KSF             /SKIP IF /FLAG = 1
                JMP.-1          /TEST FLAG AGAIN
                KRB             /READ CHARACTER INTO AC
                JMP I READ      /EXIT TO MAIN PROGRAM
```

Teleprinter/Punch

On program command a character is sent in parallel from the accumulator (AC) to the TTO shift register for transmission to the teleprinter/punch unit. The control generates the start space, then shifts the eight character bits serially into the printer selector magnets of the teletype unit, and then generates the stop marks. This transfer of information from the TTO into the teleprinter/punch unit is accomplished at the normal teletype rate and requires 100 milliseconds for completion. The flag in the teleprinter control is again set to a 1 when the last of the character code has been sent to the teleprinter/punch, indicating that the TTO is ready to receive a new character from the AC. The flag is connected to both the program interrupt synchronization element and the instruction skip element. Unless using the interrupt, the program must check the flag and, upon detecting the ready or set (binary 1) condition of the flag by means of the TSF microinstruction, the program must issue a TLS microinstruction which clears the flag and sends a new character from the AC to the TTO to be shifted out to the teleprinter/punch. The process of sending a character to the TTO from the AC is a great deal shorter than that of shifting the character out to the teleprinter/punch, therefore, the program must account for the time differential by waiting for flag to be set (1) before issuing a TLS.

Instructions for use in outputing data to the teletype are as follows:

TSF Skip on Teleprinter Flag

Octal code: 6041
Event time: 1
Execution time: 4.25 μsec
Operation: The teleprinter flag is sensed, and if it contains a binary 1 the content of the PC is incremented by one so that the next sequential instruction is skipped.
Symbol: If Teleprinter Flag = 1, then PC + 1 = > PC

TCF Clear Teleprinter Flag

Octal code: 6042
Event time: 2
Execution time: 4.25 μsec
Operation: The teleprinter flag is cleared to 0.
Symbol: 0 = > Teleprinter Flag

TPC Load Teleprinter and Print

Octal code: 6044
Event time: 3
Execution time: 4.25 μsec
Operation: The contents of bits 4-11 of the AC are sent to the TTO, then the hardware starts shifting the character out to the printer/punch unit. This microinstruction does not clear the teleprinter flag.
Symbol: $C(AC_{4-11}) = >$ TTO causing:
C(TTO) = > printed and (if punch on) punched

TLS Load Teleprinter Sequence

Octal code: 6046
Event time: 2, 3
Execution time: 4.25 μsec

Operation: This microinstruction combines the functions of the TCF and the TPC. The teleprinter flag is cleared (set to 0) then the contents of bits 4-11 of the AC are sent to the TTO, where the hardware shifts the character out to the printer/punch unit. When the printer/punch has finished outputting the character and is ready for another character, the hardware has again raised the teleprinter flag (set it to a 1) to indicate this free condition. The whole operation, from the time at which the TLS has cleared the flag and sent out the character until the time at which the hardware finishes with the character and sets the flag to a 1 again, requires 100 milliseconds with the time required for the character to travel from the TTO to the paper being considerably greater than that required for it to be sent from the computer to the TTO.

Symbol: 0 = > Teleprinter flag
C(AC 4-11) = > TTO causing:
C(TTO) = > Printed and (if punch on) punched
1 = > Teleprinter flag when done

The following are examples of possible ways to use these instructions to output a character to the teletype. The last is recommended:

```
                .
                .
                .
                CLA
                TAD X           /PUT CHARACTER CODE INTO AC FROM LOACATION X
                TLS             /LOAD TTO FROM AC & PRINT/PUNCH
FREE,           TSF             /TEST FLAG TO SEE IF DONE PRINTING, SKIP IF = 1
                JMP FREE        /TEST FLAG AGAIN
                CLA             /CLEAR CHARACTER CODE FROM AC
                .
                .
                .               continue program
```

This sequence sends one character code to the TTO and waits for it to finish printing/punching before continuing program. It does not require that the flag to be set, in order to output the character. By making this sequence of instructions a subroutine of a larger program, it can be accessed (by a JMS) each time a character is to be output. Assume that the subroutine is entered with the character code in the AC:

```
TYPE,           0
                TLS             /LOAD TTO FROM AC AND PRINT/PUNCH
                TSF             /TEST FLAG' SKIP IF = 1
                JMP.-1          /JMP BACK & TEST FLAG AGAIN AND AGAIN
                CLA             /CLEAR CHARACTER FROM AC
                JMP I TYPE      /EXIT TO MAIN PROGRAM
                .
                .
                .
```

By rearranging this subroutine the present time spent waiting for the character to be output and the flag to be set to 1 (100 milliseconds) can be used to continue the calculations, etc, of the main program thus making more efficient use of time.

```
TYPE,       0
            TSF             /TEST FLAG TO SEE IF PRINTER FREE, SKIP IF YES OR
            JMP.-1          /WAIT TIL IT IS BY TESTING AGAIN AND AGAIN
            TLS             /OUTPUT CHARACTER
            CLA
            JMP I TYPE      /EXIT TO CONTINUE PROGRAM
            .
            .
            .
```

This subroutine tests the flag first and waits only if a previous character is still being output. It clears the AC and exits immediately after sending the character to the TTO and is continuing to run the user's program instead of waiting while the teletype (a much slower device) is off typing/punching the last character. The PDP-12 clears all flags which are on the clear flag bus (this includes teletype flags) when key IO PRESET is depressed. This means that the user program must account for setting the teleprinter flag initially and after each TCF if (any) or else the program will hang up in the wait loop of the print routine. The only way to set the flag to a 1 is through issuing a microinstruction which leaves the flag set when done. This instruction should appear among the first few executed and must appear before any attempt to output a character.

The following example initializes the flag with a TLS as the first instruction of the program and makes optimum use of the time that would be spent waiting for the teletype to finish.

```
BEGIN,      TLS             /INITIALIZE TELEPRINTER FLAG
            .
            .
            .
TYPE,       0
            TSF             /SKIP IF FLAG = 1 or . . .
            JMP.-1          /WAIT UNTIL IT IS LOAD TTO &
            TLS             /TYPE CHARACTER
            CLA
            JMP I TYPE      /EXIT & CONTINUE PROGRAM WHILE
            .               /TELETYPE IS FINISHING CHARACTER
            .
            .
```

5.4.2 TTY/DATA Phone Interface (DP12)

The DP12 is an eight bit serial interface to be used with Teletypes or Data Phones. The DP12A is 110 Baud Teletype interface. The DP12B is a crystal clock controlled interface for which the user supplies the BAUD rate between 20 and 100,000 Baud. The DP12B has input and output signals essentially equivalent to EIA Standard

Programming.

The programming of the DP12 is identical to that of the system teletype but uses I/O codes 40 and 41.

The Teletype interrupt disable function of the LINC ESF instruction applies to the DP12 also.

5.4.3 Teletype Option (Type PT08)

The Teletype facility of the basic computer can be expanded to accommodate several Mode 33 or Model 35 Automatic Send Receive or Keyboard Send Receive units with the PT08 option. A PT08 option allows a Teletype to be interfaced to the PDP-12. Each Teletype line added contains logic elements that are functionally identical to those of the basic Teletype control. Therefore, instructions and programming for each PT08 are similar to those described previously for the basic Teletype unit. The following device select codes have been assigned for 5 PT08 options.

Line Unit	*Select Codes*
1	42 and 43
2	44 and 45
3	46 and 47
4	40 and 41 ← Same as Dataphone (see p. 5-51)
5	11 and 12

Instruction mnemonics for Teletype equipment in the PT08 system are not recognized by the program assembler (PAL III) and must be defined by the programmer. Mnemonic codes can be defined by the mnemonic code of the comparable basic Teletype microinstruction, suffixed with PT and the line number. For example, the following instructions can be defined for line 3:

Mnemonic	Octal	Operation
TSFPT3	6441	Skip if teleprinter 3 flag is a 1.
TCPPT3	6442	Clear teleprinter 3 flag.
TPCPT3	6444	Load teleprinter 3 buffer (TT03) from the content of AC4-11 and print and/or punch the character.
TLSPT3	6446	Load TT03 from the content of AC4-11, clear teleprinter 3 flag, and print and/or punch the character.
KSFPT3	6451	Skip if keyboard 3 flag is a 1.
KCCPT3	6452	Clear AC and clear keyboard 3 flag.
KRSPT3	6454	Read keyboard 3 buffer (TTI3) static. The content of TTI3 is loaded into AC4-11 by an OR transfer.
KRBPT3	6456	Clear the AC, clear keyboard 3 flag, and read the content of TTI3 into AC4-11.

5.4.4 KW12 Real Time Clock

The KW12 is a prewired PDP-12 option including a kit of modules and an input control panel. The input panel mounts behind the vertical door on the left front of the PDP-12. The clock consists of a 12-bit counter with overflow and a 12-Bit Buffer Preset Register. This register buffers information transfers between the clock counter and the Processor Accumulator. The components of the KW12 include:

Time Base

The programmable time base provides count pulses to the counter register at any of the following rates derived from a 400 kHz crystal clock:

400 kHz
100 kHz
10 kHz
1 kHz
100 Hz

Input channel 1 may be used to enable an external source to drive the counter. The programmable selection of the rate is accomplished with the three rate bits of the clock control register.

Input Synchronizers

There are three input channels which are used to convert external events into a sychronized control and status signal for the clock. Each input channel consists of an input Schmitt trigger with pulse generator, five flip-flops and associated control gating. The function of the Schmitt trigger and pulse generator is to convert the crossing of a preselected voltage threshold by an external signal into a single event (pulse). This Schmitt trigger has level and slope selection available on the front control panel. This provides selection of any threshold between ±2v and either positive or negative going slope. The Schmitt trigger has a hysteresis of 0.3v. Refer to Figure 5-28.

The five flip-flops of each of the Input Circuits are: INPUT ENABLE, INPUT, PRE-EVENT, EVENT, and ENABLE EVENT INTERRUPT. Figure 5-29 shows a simplified Input Synchronizer Logic diagram.

Input Enable Flip-Flop - This flip-flop gates on and off input signals to the clock. It is set and cleared under program control.

Input Flip-Flop - The INPUT flip-flop is set by an external signal from the Schmidt trigger input or under program control with the CLLR instruction. The INPUT flip-flop provides synchronization between external timing and internal clock timing.

Pre-Event Flip-Flop - The input is strobed into the PRE-EVENT flip-flop by the strobe 1 signal (400 kHz). The strobe 2 signal occurs 1 μsec after strobe 1, this clears the INPUT flip-flop if EVENT is on a 0.

Event Flip-Flop - The EVENT flip-flop is loaded with the PRE-EVENT flip-flop on the next strobe 1 and continues to remain set during succeeding strobe 1 pulses. When EVENT is loaded PRE-EVENT is cleared.

The occurrence of strobe 2 with EVENT (0) and PRE-EVENT (1) is the actual single event used by other parts of the clock logic such as counting and transfers from COUNTER TO BUFFER register.

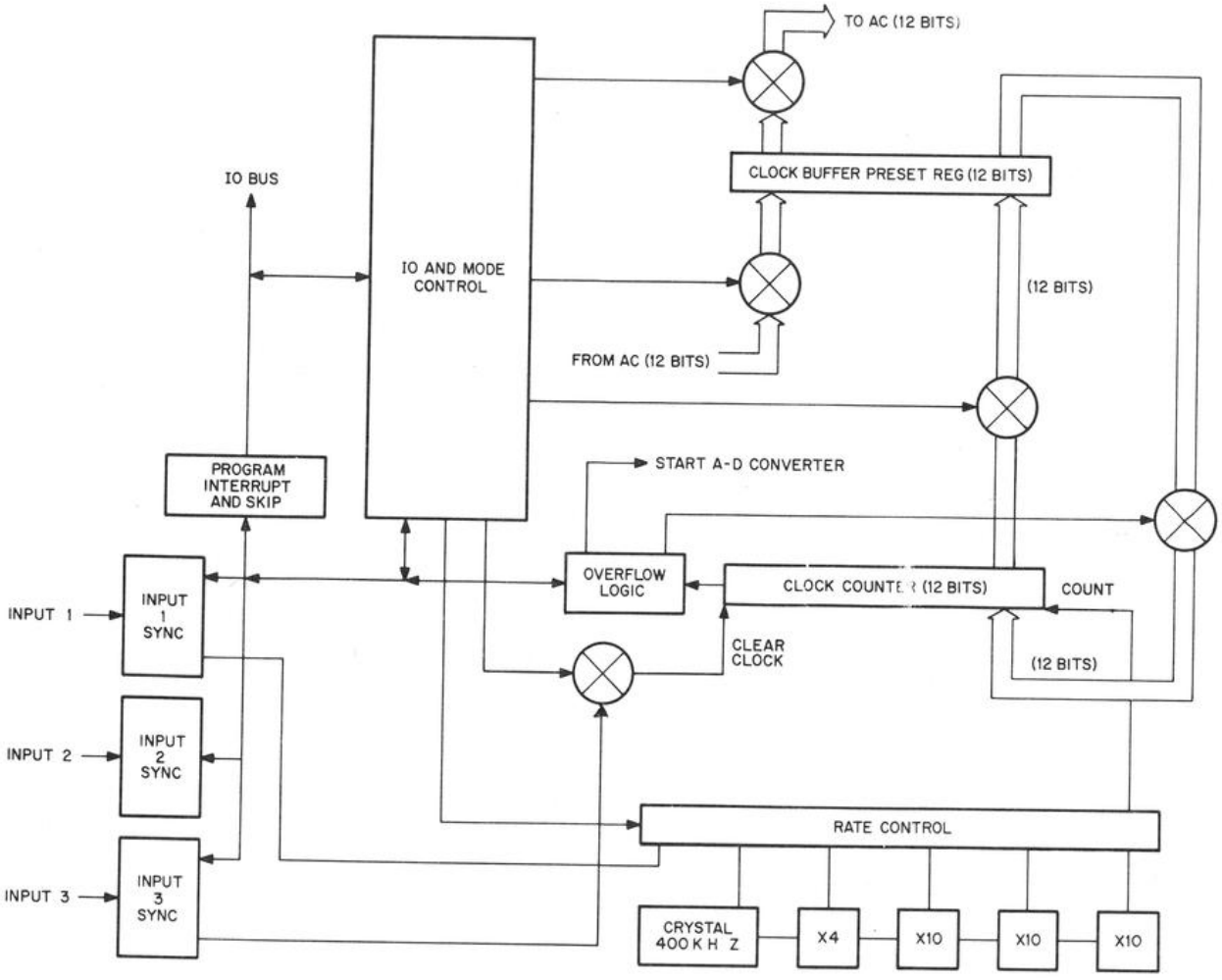


Figure 5-28. KW12 Clock Organization

The EVENT and PRE-EVENT flip-flops can be loaded into the AC under program control. When this transfer occurs the corresponding INPUT, PRE-EVENT and EVENT flip-flops are cleared.

If a second input occurs before EVENT is cleared then both the PRE-EVENT and EVENT flip-flops will remain set as an error indication.

Enable Event Interrupt - This flip-flop connects the EVENT flip-flop to both the Skip and Program Interrupt busses. It is set and cleared under program control.

Counter Register

The COUNTER register is a 12-bit counter which can be loaded from the BUFFER-PRESET register and can be transferred into the BUFFER-PRESET register. The counter is usually left free running at the program selected rate and its contents read through the Buffer Preset register.

Overflow Flip-Flop - The OVERFLOW flip-flop is set by the most significant bit of the counter register going from 1 to 0.

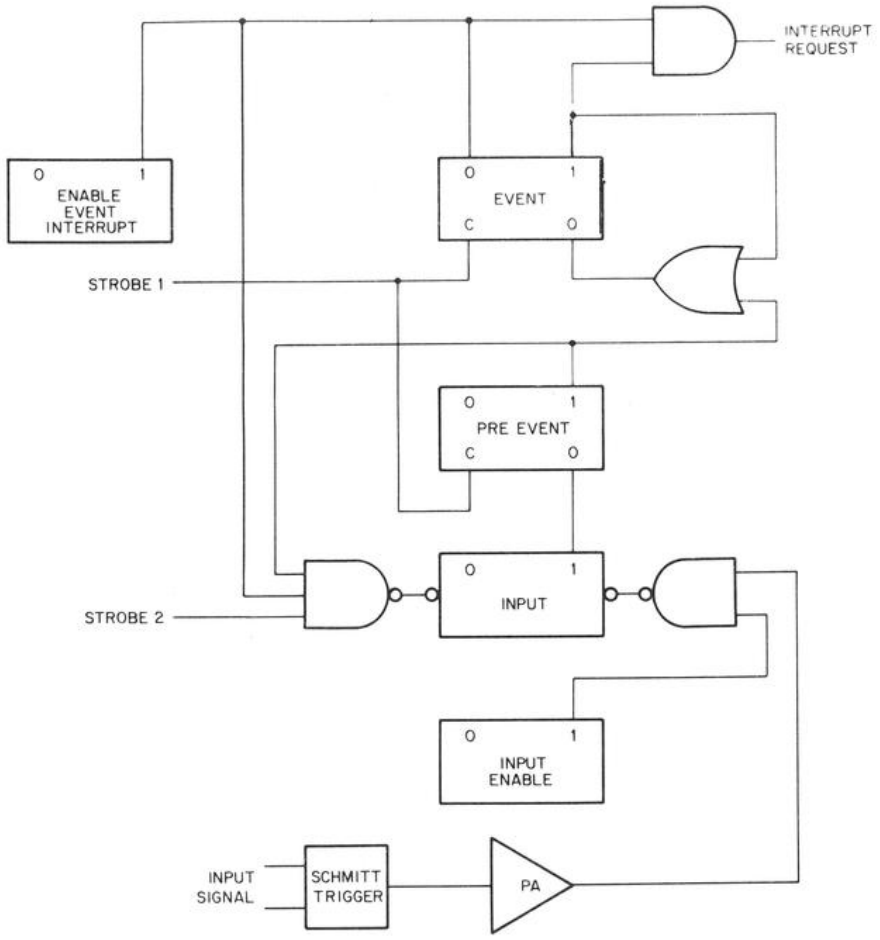


Figure 5-29. Simplified Input Synchronizer

Buffer-Preset Register - This 12-bit register is used to buffer the current count in the clock register at the occurrence of an *event* when operating with Mode 1 (1). With Mode 1 (0) and Mode 2 (1) the Buffer Preset register holds the number to be transferred into the counter when overflow occurs. The Buffer Preset Register can be loaded into the AC or the AC can be transferred into the Buffer-Preset register.

Use Of Clock With A-D

With Mode 0 (1) the occurrence of overflow is used to start an A-D conversion if the A-D is in the Fast Sample mode. With clock Mode 0 (1) the A-D is only triggered by the clock. When a SAM instruction is given the conversion just completed is transferred to the AC and the new MPX channel is selected.

If the SAM instruction is given prior to completion of the conversion triggered by the clock, the processor waits in TIME STATE 5 until the conversion is complete.

Clock Input Panel

The input panel for the clock is located behind the door on the left side of the front of the PDP-12. Each input channel has three input binding posts (input high, input low, and ground). The input is differential, ±2v range, input resistance greater than 10,000 ohms, and protected against inputs up to ±100v. Associated with each input is a level control and a slope control. The slope determines the slope of the input signal which will cause a trigger pulse. The level control selects the input voltage at which the trigger pulse will be generated. The trigger pulse will set the associated input flip-flop of the clock if that input channel is enabled.

KW12 Clock Instructions

The KW12 clock is controlled by PDP-12 IOT instructions. These instructions can be used from either PDP-8 mode directly or in conjunction with the LINC IOB instruction. Execution time for the clock IOT's is 4.25 microseconds. When used with IOB the total instruction time is 5.9 microseconds.

CLSK Skip On Clock Interrupt

Octal code: 6131
Event time: 1
Execution time: 4.25 μsec
Operation: Skip if clock interrupt condition exists. The interrupt conditions are as follows:

a. Enable Event 1 Interrupt (1) and Event 1 (1).

b. Enable Event 2 Interrupt (1) and Event 2 (1).

c. Enable Event 3 Interrupt (1) and 3 (1).

d. Enable Overflow Interrupt (1) and Overflow (1).

CLLR Load Clock Control Register 1

Octal code: 6132
Event time: 2
Execution time: 4.25 μsec
Operation: The contents of the AC register is transferred to the clock control register. Three bits are used to provide simulated data input to each of the three Event input channels. The AC is unchanged.

Bit	
00	Count Rate Register Bit 0
01	Count Rate Register Bit 1
02	Count Rate Register Bit 2
03	Mode Control Register Bit 0
04	Mode Control Register Bit 1
05	Mode Control Register Bit 2
06	~~Not used~~ Mode Control Reg. Bit 3
07	Simulate Input to Channel 1
08	Not used

Bit (cont)

09	Simulate Input to Channel 2
10	Not used
11	Simulate Input to Channel 3

The rate of the counter register input count pulses is determined by the contents of the Count Rate register.

Count Rate Register	*Frequency of Count Pulses*
000	Stop
001	400 kHz
010	100 kHz
011	10 kHz
100	1 kHz
101	100 Hz
110	Rate of Input Channel 1
111	Stop

NOTE

When Channel 1 is used as the time base for the counter, the Event flag is automatically cleared and Channel 1 Interrupt Enable would normally be left off.

The contents of the Mode Control Register determines the method by which the clock system operates.

Mode Control Register

000	Counter runs selected rate. Overflow occurs every 4096 counts. The overflow flag remains set until cleared with 6135 instruction.
001	Counter runs at selected rate. When overflow occurs the contents of the Clock Buffer-Preset register are automatically transferred to the Counter which continues. The Overflow flag remains set until cleared with a 6135 instruction.
010	Counter runs at selected rate. On the occurrence of an *Input Event* the contents of the counter are automatically transferred to the Buffer Preset register and the counter continues to count.
011	This is identical to Mode 10 except that the Clock Counter register is cleared after its contents have been transferred to the Buffer Preset register on Event 3. Event 1 and 2 remain only to cause transfer from the clock counter to the Buffer Preset register.

Mode Control Register (cont)

100
101
110
111

When Mode 0 is set to a 1 the occurrence of Overflow is used to trigger the A-D converter if A-D control also has the FAST-SAMPLE flip-flop set. This allows analog to digital conversion to take place under the automatic timing control of the clock. For details of the Analog-to-Digital converter, see Section 3.30. The remaining two Mode Control bits are decoded exactly as above.

XXX1 In Fast sample mode sample command conc thru evq + 2

CLAB AC to Buffer Preset Register

Octal code: 6133
Event time: 2
Execution time: 4.25 μsec
Operation: Transfer AC to Buffer-Preset register. The previous contents of the Buffer Preset register are lost and the AC is unchanged.

CLEN Load Clock Enable Register

Octal code: 6134
Event time: 3
Execution time: 4.25 μsec
Operation: The contents of the AC register are transferred to the Clock Enable register. The function of each bit is given below:

Bit	
00	Not used
01	Not used
02	Not used
03	Not used
04	Not used
05	Enable Interrupt when overflow (1)
06	Enable Interrupt on Event 1
07	Enable Input Channel 1
08	Enable Interrupt on Event 2
09	Enable Input Channel 2
10	Enable Interrupt Event 3
11	Enable Input Channel 3

CLSA Clock Status to AC

Octal code: 6135
Event time: 1.3
Execution time: 4.25 μsec
Operation: This instruction is used to interrogate the clock input and overflow status flip-flops. The clock status information is inclusive ORed into the AC. Then the status bits for which the AC is set are cleared. This assures 1 and only 1 occurrence of an EVENT being transferred to the program.

CLSA Clock Status to AC (cont)

AC Bit	*Status Condition*
00	Overflow Flip-Flop
01	Not used
02	Not used
03	Not used
04	Not used
05	Not used
06	Event 1
07	Pre-Event 1
08	Event 2
09	Pre-Event 2
10	Event 3
11	Pre-Event 3

Buffer Preset Register to AC

Octal code: 6136
Event time: 2, 3
Execution time: 4.25 μsec
Operation: The AC register is cleared and the contents of the Clock Buffer Preset register are transferred into the AC.

CLCA Counter to AC

Octal code: 6137
Event time: 1, 2, 3
Execution time: 4.25 μsec
Operation: The AC register is cleared and the contents of the Clock Counter are transferred to the Buffer Preset register. Then the contents of the Buffer Preset register are transferred into the AC.

5.4.5 Incremental Plotter and Control (Type XY12)

Four models of California Computer Products Digital Incremental Recorder can be operated from a Digital Type XY12 Incremental Plotter Control. Characteristics of the four recorders are:

CCP Model	*Step Size (inches)*	*Speed (steps/minute)*	*Paper Width (inches)*
563	0.01 or 0.005	12,000	31
565	0.01 or 0.005	18,000	12

The principles of operation are the same for each of the four models of Digital Incremental Recorders. Bidirectional rotary step motors are employed for both the X and Y axes. Recording is produced by movement of a pen relative to the surface of the graph paper, with each instruction causing an incremental step. X-axis deflection is produced by motion of the drum; Y-axis deflection, by motion of the pen carriage. Instructions are used to raise and lower the pen from the surface of the paper. Each incremental step can be in any one of eight directions through appropriate combinations of the X and Y axis instructions. All recording (discrete points, continuous curves, or symbols) is accomplished by the incremental stepping action of the paper drum and pen carriage. Front panel controls permit single-step or continuous-step manual operation of the drum and carriage, and manual control of the pen solenoid. The recorder and control are connected to the computer program interrupt and instruction skip facility.

Instructions for the recorder and control are:

PLPU Pen Up

Octal code: 6504
Event time: 3
Execution time: 4.25 μsec
Operation: The plotter pen is raised from the surface of the paper.
Symbol: None

PLPR Pen Right

Octal code: 6511
Event time: 1
Execution time: 4.25 μsec
Operation: The plotter pen is moved to the right in either the raised or lowered position.
Symbol: None

PLDU Drum Up

Octal code: 6512
Event time: 2
Execution time: 4.25 μsec
Operation: The plotter paper drum is moved upward. This command can be combined with the PLPR and PLDD commands.
Symbol: None

PLDD Drum Down

Octal code: 6514
Event time: 3
Execution time: 4.25 μsec
Operation: The plotter paper drum is moved downward.
Symbol: None

PLPL Pen Left

Octal code: 6521
Event time: 1
Execution time: 4.25 μsec
Operation: The plotter pen is moved to the left in either the raised or lowered position.
Symbol: None

PLUD Drum Up

Octal code: 6522
Event time: 2
Execution time: 4.25 μsec
Operation: The plotter paper drum is moved upward. This command is similar to the command 6512 except that it can be combined with the PLPL or PLPD commands.
Symbol: None

PLPD Pen Down

Octal code: 6524
Event time: 3
Execution time: 4.25 μsec
Operation: The plotter pen is lowered to the surface of the paper.
Symbol: None

Program sequence must assume that the end location is known at the start of a routine since there is no means of specifying an absolute pen location in an incremental plotter. Pen location can be preset by the manual controls on the recorder. During a subroutine, the PDP-12 can track the location of the pen on the paper by counting the instructions that increment position of the pen and the drum.

5.4.6 High-Speed Perforated Tape Reader and Control (Type PR 12)

This device senses 8-hole perforated paper or Mylar tape photoelectrically at 300 characters per second. The reader control requests reader movement, transfers data from the reader into the reader buffer (RB), and signals the computer when incoming data is present. Reader tape movement is started by clearing the reader flag. Data is assembled in the Reader Buffer from the perforated tape. The Reader Buffer is transferred into bits 4 through 11 of the accumulator under program control. The reader flag is connected to the computer program interrupt and instruction skip facilities, and is cleared by IOT pulses. The control circuitry for this device is located in the BA12 Option Mounting Panel. Computer instructions for the reader are:

RSF Skip on Reader Flag

Octal code: 6011
Event time: 1
Execution time: 4.25 μsec
Operation: The reader flag is sensed, and if it contains a binary 1 the content of the PC is incremented by one so that the next sequential instruction is skipped.
Symbol: If Reader Flag = 1, then PC + 1 => PC

RRB Read Reader Buffer

Octal code: 6012
Event time: 2
Execution time: 4.25 μsec
Operation: The content of the reader buffer is transferred into bits 4 through 11 of the AC and the reader flag is cleared. This command does not clear the AC.
Symbol: RB V AC_{4-11} => AC_{4-11}
0 => Reader Flag

RFC Reader Fetch Character

Octal code: 6014
Event time: 3
Execution time: 4.25 μsec
Operation: The reader flag and the reader buffer are both cleared, one character is loaded into the reader buffer from tape, and the reader flag is set when this operation is completed.
Symbol: 0 => Reader Flag, RB
Tape Data => RB
1 => Reader Flag when done

A program sequence loop to read a character from perforated tape can be written as follows:

```
        RFC       /FETCH CHARACTER FROM TAPE
LOOK,   RSF       /SKIP WHEN RB FULL
        JMP LOOK
        CLA
        RRB       /LOAD AC FROM RB
```

5.4.7 High-Speed Tape Punch and Control (Type PP 12)

This option consists of a Royal McBee paper tape punch that perforates 8-hole tape at a rate of 50 characters per second. Information to be punched on a line of tape is loaded in an 8-bit punch buffer (PB) from AC bits 4 through 11. The punch flag becomes a 1 at the completion of punching action, signaling that new information may be transferred into the punch buffer, and punching initiated. The control circuitry for this device is located in the BA12 Option Mounting Panel. The punch flag is as described for the Teletype unit. The punch instructions are:

PSF Skip on Punch Flag

Octal code: 6021
Event time: 1
Execution time: 4.25 μsec
Operation: The punch flag is sensed, and if it contains a binary 1 the content of the PC is incremented by one so that the next sequential instruction is skipped.
Symbol: If Punch Flag = 1, then PC + 1 = > PC

PCF Clear Punch Flag

Octal code: 6022
Event time: 2
Execution time: 4.25 μsec
Operation: Both the punch flag and the punch buffer are cleared in preparation for receiving a new character from the computer.
Symbol: 0 = > Punch Flag, PB

PPC Load Punch Buffer and Punch Character

Octal code: 6024
Event time: 3
Execution time: 4.25 μsec
Operation: An 8-bit character is transferred from bits 4 through 11 of the AC into the punch buffer and then this character is punched. This command does not clear the punch flag or the punch buffer.
Symbol: AC_{4-11} V PB = > PB

PLS Load Punch Buffer Sequence

Octal Code: 6026
Event Time 2, 3
Execution Time: 4.25 μsec
Operation The punch flag and punch buffer are both cleared, the content of bits 4 through 11 of the AC is transferred into the punch buffer, the character in the PB is punched in tape, and the punch flag is set when the operation is completed.
Symbol: 0 = > Punch Flag, PB
AC4-11 = > PB
1 = > Punch Flag when done

A program sequence loop to punch a character when the punch buffer is "free" can be written as follows:

```
FREE,    PSF        /SKIP WHEN FREE
         JMP FREE
         PLS        /LOAD PB FROM AC AND PUNCH CHARACTER
```

5.4.7 Card Reader and Control (Type CR12)

The Card Reader and Control Type CR12 reads 12 row, 80 column punched cards at a nominal rate of 200 cards per minute. Cards are read by column, beginning with column 1. One select instruction starts the card moving past the read station. Once a card is in motion, all 80 columns are read. Data in a card column is photo-electrically sensed. Column information is read in one of two program selected modes: alphanumeric and binary. In the alphanumeric mode, the 12 information bits in one column are automatically decoded and transferred into the least significant half of the accumulator as a 6-bit Hollerith code. In the binary mode, the 12 bits of a column are transferred directly into the accumulator so that the top row (12) is transferred into AC0 and the bottom row (9) is transferred into AC11. A punched hole is interperted as a binary 1 and no hole is interpreted as a binary 0.

Three program flags indicate card reader conditions to the computer. The data ready flag rises and requests a program interrupt when a column of information is ready to be transferred into the AC. A read alphanumeric or read binary command must be issued within 1.4 milliseconds after the data ready flag rises to prevent data loss. The card done flag rises and requests a program interrupt when the card leaves the read station. A new select command must be issued immediately after the card done flag rises to keep the reader operating at maximum speed. Sensing of this flag can eliminate the need for counting columns, or combined with column counting can provide check for data loss. The reader-not-ready flag can be sensed by a skip command to provide indication of card reader power off, pick failure, a dark check indication, a stacker failure, hopper empty, stacker full, Sync failure or light check indication. When this flag is raised, the reader cannot be selected and select commands are ignored. The reader-not-ready flag is not connected to the program interrupt facility and cannot be cleared under program control. Manual intervention is required to clear the reader-not-ready flag. Instructions for the CR8/I are:

RCSF Skip on Data Ready

Octal Code: 6631
Event Time: 1
Execution Time: 4.25 μsec
Operation: The content of the data ready flag is sensed, and if it contains a 1 (indication that information for one card column is ready to be read) the content of the PC is incremented by one so the next sequential instruction is skipped.
Symbol: If Data Ready Flag = 1, then PC + 1 = > PC

RCRA Read Alphanumeric

Octal Code: 6632
Event Time: 2
Execution Time: 4.25 μsec
Operation: The 6-bit Hollerith code for the 12 bits of a card column are transferred into bits 6 through 11 of the AC, and the data ready flag is cleared.
Symbol: AC6-11 V Hollerith = > AC_{6-11}
0 = >Data Ready Flag

RCRB Read Binary

Octal Code: 6634
Event Time: 3
Execution Time: 4.25 μsec
Operation: The 12-bit binary code for a card column is transferred directly into the AC, and the data ready flag is cleared. Information from the card column is transferred into the AC so that card row 12 enters ACO, row 11 enters AC_1 ,row 0 enters AC_2 ,. . . and row 9 enters AC11.
Symbol: AC V Binary Code = > AC
0 = > Data Ready Flag

RCSD Skip on Card Done Flag

Octal Code: 6671
Event Time: 1
Execution Time: 4.25μsec
Operation: The content of the card done flag is sensed, and if it contains a 1 (indication that the card has passed the read station) the content of the PC is incremented to skip the next sequential instruction.
Symbol: If Card Done Flag = 1, then PC + 1 = > PC

RCSE Select Card Reader and Skip if Ready

Octal code: 6672
Event Time: 2
Execution Time: 4.25 μsec
Operation: The status of the card reader is sensed. If the reader is ready, the PC is incremented to skip the next sequential instruction, a card is started toward the read station from the input hopper and the card done flag is cleared.
Symbol: If reader flag = 1, then PC + 1 = > PC
O = > Card Done Flag

RCRD Clear Card Done Flag

Octal code: 6674
Event time: 3
Execution time: 4.25 μsec
Operation: The card done flag is cleared. This command allows a program to stop reading at any point in a card deck.
Symbol: 0 = > Card Done Flag

A logical instruction sequence to read cards is:

```
START,      RCSE          /START CARD MOTION AND SKIP IF READY
            JUMP NOT RDY  /JUMP TO SUBROUTINE THAT TYPES OUT
                          /"CARD READER MANUAL INTERVENTION
                          /REQUIRED" OR HALTS
NEXT,       RCSF          /DATA READY?
            JMP DONE      /NO, CHECK FOR END OF CARD
            RCRA or RCRB  /YES, READ ONE CHARACTER OR ONE
                          /COLUMN AND CLEAR DATA READY
```

```
        DCA 1 STR       /STORE DATA
DONE,   RCSD            /END OF CARD?
        JMP NEXT        /NO, READ NEXT COLUMN
        JMP OUT         /YES, JUMP TO SUBROUTINE THAT CHECKS
                        /CARD COUNT OR REPEATS AT START FOR
                        /NEXT CARD
```

No validity checking is performed by the CR12. A programmed validity check can be made by reading each card column in both the aphanumeric and binary mode (within the 1.4 millisecond time limitation), then performing a comparison check.

Before commencing a card reading program, load the input hopper with cards and press Motor Start and Read S Start pushbuttons. The function of the manual controls and indicators are as follows (as they appear from left to right on the card reader):

Control or Indicator	*Function*
A – POWER switch	On-Off toggle switch. Applies power to all circuits except drive motor.
B – MOTOR START	Momentary action pushbutton, with separate indicator. Applies power to main drive motor. Motor start is also used as a reset to clear error indicators and therefore will not operate if there is an unremedied condition such as: 1. Input hopper is empty. 2. Output hopper is full. 3. All photo cells are not lit. 4. Internal power supplier is not operational.
C – READ START	Momentary action pushbutton, with separate indicator. Causes ready line to go high, which enables card reading under control of the external read command. If read command is open or high, card reading begins immediately at full rated speed.
D - READ STOP	Momentary action pushbutton, with separate indicator. Inhibits further card reading until READ START switch is pressed again. Ready line goes low and READ STOP condition is indicated. Does not stop drive motor. However, a READ STOP condition is indicated anytime the drive motor is stopped.
E - INDICATORS	Several detection circuits are incorporated in the card reader. Whenever any red indicator lights, the drive motor is stopped after allowing the completion of the current card cycle.
1. PICK FAIL Indicator	Lights when a card fails to enter the read station after two successive pick attempts.
2. DARK CHECK Indicator	After the card enters the read station, a check is made at the hypothetical 0th and 81st hole positions to be sure all photocells are dark. If not, the DARK CHECK indicator lights and data outputs are inhibited immediately.

Controls or Indicator	*Function*
3. STACKER FAIL Indicator	When three cards have passed the read station and none have been stacked, a STACK FAIL is indicated. Prevents more than three cards from being in the track at once.
4. HOPPER EMPTY Indicator	Indicates input hopper is empty.
5. STACKER FULL Indicator	Switch closure detects when approximately 400 cards are in the stacker hopper.
6. SYNC FAIL Indicator	Internal timing signals are derived from an oscillator which is sync'ed to the track speed. If the sync signal is lost, a SYNC FAIL is indicated.
7. LIGHT CHECK Indicator	All photo cells must always be lit except during the time a card is being read. The detector is inhibited each time a card enters the read station until position (count of) 84. If a card fails to leave the read station by this time, a LIGHT CHECK is indicated.

5.4.8 Digital-To-Analog Converter (Type AA01A)

The general-purpose Digital-to-Analog Converter Type AA01A converts 12-bit binary computer output numbers to analog voltages. The basic option consists of three channels, each containing a 12-bit digital buffer register and a digital-to-analog converter (DAC). Digital input to all three registers is provided, in common, by one 12-bit input channel which receives bussed output connections from the accumulator. Appropriate precision voltage reference supplies are provided for the converters.

One IOT microinstruction simultaneously selects a channel and transfers a digital number into the selected register. Each converter operates continuously on the content of the associated register to provide an analog output voltage.

Type AA01A options can be specified in a wide range of basic configurations; e.g., with from one to three channels, with or without output operational amplifiers, and with internally or externally supplied reference voltages. Configurations with double buffer registers in each channel are also available.

Each single-buffered channel of the equipment is operated by a single IOT command. Select codes of 55, 56, and 57 are assigned to the AA01A, making it possible to operate nine single-buffered channels or various configurations of double-buffered channels. A typical instruction for the AA01A is:

DAL1 Load Digital-to-Analog Converter 1

Octal Code:	6551
Event Time:	1
Execution Time:	4.25 μsec
Operation:	The content of the accumulator is loaded into the digital buffer register of channel 1.
Symbol:	AC => DAC1

5.4.9 Random Access Disk File (Type DF32)

Operating through the 3-cycle data-break channel, the DF32 provides 32,768 13-bit words (12 bits plus parity) of storage, and is economically expandable to 131,072 using Expander Disk Type DS32.

Transfer rate of the DF32 is 66 μsec per word; average access time is 16.67 msec for 60-cycle power (20 msec with 50-cycle power).

Two basic assemblies comprise the DF32: the storage unit with read/write electronics, and computer interface logic. The storage unit contains a nickel-cobalt plated disc driven by a hysteresis synchronous motor. Data is recorded on a single disc surface by 16 read/write heads which are in a fixed position. A photo-reflective marker is used on the disc's outer perimeter to denote beginning and end of timing and address tracks.

Disk motor and shaft, read/write data heads, timing and address heads, and photocell assembly are mounted on a rack assembly which permits sliding the unit in and out of a standard Digital Equipment Corporation cabinet.

The disk is designed for rack mounting in a 19 inch relay rack.

Instructions

The command for the disk system are as follows:

DCMA Clear Disk Memory Address Register

Octal Code: 6601
Event Time: 1
Execution Time: 4.25 μsec
Operation: Clears Memory Address Register, parity error and completion flags. This instruction clears the disk memory request flag and interrupt flags.
Symbol: 0 = > completion flag
0 = > error flag

DMAR Load Disk Memory Address Register and Read

Octal code: 6603
Event time: 1, 2
Execution time: 4.25 μsec
Operation: The contents of the AC are loaded into the disk memory address register and the AC is cleared. Begin to read information from the disk into the specified core location. Clears parity error and completion flags. Clears interrupt flags.
Symbol: $AC_{0-11} \rightarrow DMA_{0-11}$
0 = > completion flag
0 = > error flag

DMAW Load Disk Memory Address Register and Write

Octal Code: 6605
Event Time: 1, 3
Execution Time: 4.25 μsec
Operation: The contents of the AC are loaded into the disk memory address register and the AC is cleared. Begin to write information into the disk from the specified core location. Clears parity error and completion flags. Clears interrupt flags. Data break must be allowed to occur within 66μsec after issuing this instruction.
Symbol: AC0-11DDMA0-11

$AC_{0-11} \rightarrow DMA_{0-11}$
0 = > completion flag
0 = > error flag

DCEA Clear Disk Extended Address Register

Octal Code: 6611
Event Time: 1
Execution Time: 4.25μsec
Operation: Clears the Disk Extended Address and memory address extension register.
Symbol: 0 = > Disk Extended Address Register
0 = > Memory Address Extenstion Register

DSAC Skip on Address Confirmed Flag

Octal Code: 6612
Event Time: 2
Execution Time: 4.4.25 μsec
Operation: Skips next instruction if address confirmed Flag is a 1. Flag is set for 16 μsec (AC is cleared)
Symbol: If address confirmed flag = 1, then
PC + 1 = > PC

DEAL Load Disk Extended Address

Octal code: 6615
Event time: 1, 3
Execution time: 4.25 μsec
Operation: The disk extended address and memory address extension registers are cleared and loaded with the track address data held in the AC.
Symbol: AC_{6-8} = > Core Memory Extension
AC_{1-5} = > Disk Address Extension 32K, 64K, 96K, 128K
$AC_{0\ 9-11}$ used in DEAC instruction
(See NOTE)

DEAC Read Disk Extended Address Register

Octal code: 6616
Event time: 2, 3
Execution time: 4.25 μsec

Operation: Clear the AC then load the contents of the disk extended address register into the AC to allow program evaluation. Skip next instruction if address confirmed flag is a 1.
Symbol: Disk Address Extension 32K, 64K, 96K, 128K = > AC_{1-5}
Core Memory Extention = > AC_{6-8}
Photo-cell sync mark = > AC_0 (Available 200 μsec)
Data Request Late flag = > AC_9
Non-existent or Write Lock switch on = Π AC_{10}
Parity Errors = > AC_{11}

NOTE

For the DEAL and DEAC Instructions, refer to the diagrams shown below.

BITS 1-5 (DEAL INST)	ACCUMULATOR (LOW ORDER 12 BITS) 0-11 of DMAW or DMAR	DISC ADDRESS (17 BIT)

FIELD BITS 6-8 (DEAL INST)	CELL 7751 (CURRENT ADDRESS)	CURRENT ADDRESS (MEMORY) ADDRESS (15 BIT)

DFSE Skip on Zero Error Flag

Octal code: 6621
Event time: 1
Execution time: 4.25 μsec
Operation: Skips next instruction if parity error, data request late, or write lock switch flag is a zero. Indicates no errors.
Symbol: If Parity Error flag = 1, then PC + 1 = > PC
If Data Request Late flag = 1, then PC + 1 = > PC
If Write Lock Switch flag = 1, then PC + 1 = > PC

DFSC Skip on Data Completion Flag

Octal code: 6622
Event time: 2

Execution time: 4.25 μsec
Operation: Skips next instruction if the completion flag is a 1. Indicates data transfer is complete.
Symbol: If Completion flag = 1, PC + 1 => PC

DMAC Read Disk Memory Address Register

Octal code: 6626
Event time: 2, 3
Execution time: 4.25 μsec
Operation: Clears the AC then loads contents of the Disk Memory Address register into the AC to allow program evaluation. During read, the final address will be the last one transferred.
Symbol: $DMA_{0-11} => AC_{0-11}$

The computer can handle 12 bits, therefore, the high order bits for Disk and Memory addresses are manipulated by the DEAL and DEAC instructions. Low order bits are manipulated in the accumulator (AC).

The Disk address is a 17 bit value. Bit 1 of the DEAL and DEAC instructions is the most significant bit. The Memory address is a 17 bit value. Bit 6 of the DEAL and DEAC instructions is the most significant bit.

Note that the Word Count 7750 is the 2's complement of the number of words to be transferred and that the Disk address is the desired starting address. The Memory or Current address (7751) is the desired address -1.

NOTE

Write Lock Switch status is true only when disk module contains write command. The non-existent disk condition will appear following the completion of a data transfer during read, where the address acknowledged was the last address of a disc and the next word to be addressed falls within a non-existent disk. The completion flag for the data transfer is set by the non-existent disk condition 15 microseconds following the completion flag for this data transfer is set by the non-existent disk condition 16 microseconds following the data transfer.

DF32 Software

DF32 Disk System, available with PDP-12, is a fast convenient keyboard oriented monitor which will enable the user to efficiently control the flow of programs through his PDP-12. This system is modular and open ended, allowing the user to build the software components required in his environment. The user may specify the system device (Disk or LINCtape), the amount of core, number of disks available and the number, name and size of his resident system programs.

5.4.10 Disk Memory System (Type RF08, RS08)

The RF08 control and RS08 disk combine as a fast, low-cost, random access bulk storage package for the PDP-8, PDP-8/I and PDP-12 computers. One RS08 and RF08 provide 262,144 13-bit words (12 bits plus parity) of storage. Up to four RS08 disks can be added to the RF08 control for a total of 1,048,576 words of storage.

Data transfer rate on 60 Hz power is 16.2 microseconds per word or 20 microseconds per word on 50 Hz. Data transfer is accomplished through the three-cycle break system of the PDP-8 or PDP-8/I.

Average access time with a 60 Hz disk is 16.67 milliseconds or 20 milliseconds at 50 Hz power. Worst case access time is 33 milliseconds on 60 Hz power or 40 milliseconds on 50 Hz power.

The RS08 disk unit contains a nickel-cobalt plated disk driven by a hysteresis synchronous motor. Data is recorded on a single disk surface by 128 fixed read/write heads.

The RF08 and RS08 are designed for rack mounting in a standard 19 inch DEC cabinet.

DF32 Programming Compatibility

The input-output transfer instructions for the RF/RS08 Disk Memory system are identical with the input-output transfer instructions for the Type DF32 Random Access Disk file with the following exceptions:

DEAL Load Disk Extended Address

Octal code: 6615
Execution time: 4.25 μsec
Operation: When used in the RF/RS08 Disk Memory system this IOT instruction does not transmit the extended disk address bits for addressing beyond 32K. Instead, accumulator bits AC_3, AC_4 and AC_5 are assigned to enable or disable conditions on the program interrupt line. The accumulator is cleared after this instruction has been executed.

DEAC Read Disk Extended Address Register

Octal code: 6616
Execution time: 4.25 μsec
Operation: When used in the RF08 Disk Memory system this IOT instruction does not read extended address bits AC_1 through AC_5 into the accumulator. These bits are assigned to examine the status of the interrupt enable. In addition accumulator bit AC_2 indicates the status of Write Lock and accumulator bit AC_{10} shows non-existent disk conditions only. Accumulator bit AC_1 shows the condition of Data Request Enable, which is used for maintenance purposes.

DFSE Skip Old Zero Error Flag

Octal code: 6621
Execution time: 4.25 μsec
Operation: When used in the RF08 and RS08 Disk Memory system, this IOT will cause a skip to be returned to the computer on error instead of no error. Non-existent disk has been included as an error-skip condition.

DISK Skip Error or Completion Flag

Octal code: 6623
Execution time: 4.25 μsec
Operation: This is a new skip instruction added to the RF08 and RS08 Disk Memory system. This IOT will return a skip to the computer if either the error or completion flags or both are raised.

The DF32 maintenance instruction IOT 663X is not assigned to the RF08 system.

DCIM

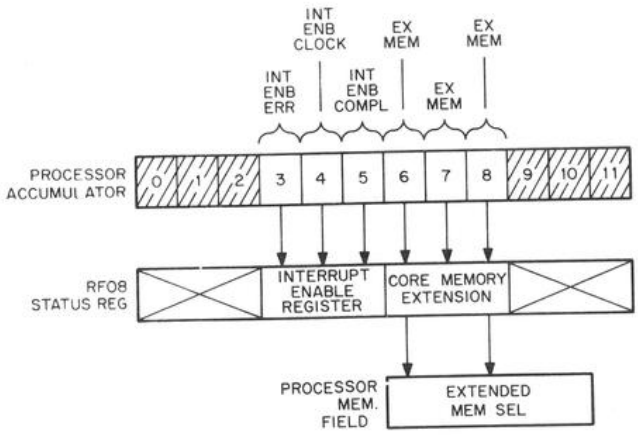
Octal code: 6611
Event time: 1
Execution time: 4.25 μsec
Operation: Clear the disk interrupt enable and core memory address extension registers.

DSAC

Octal code: 6612
Event time: 2
Execution time: 4.25 μsec
Operation: Maintenance Instruction Skip next instruction if the Address Confirmed flag is a 1. (AC is cleared.)

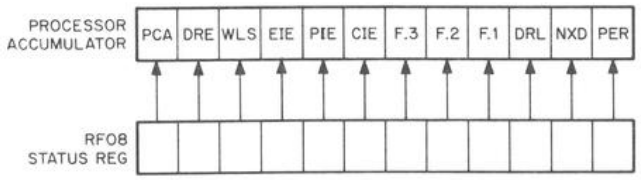
DIML

Octal code: 6615
Event time: 1, 3
Execution time: 4.25 μsec
Operation: Clear the interrupt enable, and memory address extension register. Then load the interrupt enable and memory address extension registers with data held in the accumulator. Then clear AC.



DIMA

Octal code: 6616
Event time: 2, 3
Execution time: 4.25 μsec
Operation: Clear the accumulator. Then load the contents of the status into the accumulator to allow program evaluation.



DFSE

Octal Code: 6621
Event time: 1
Execution time: 4.25 μsec
Operation: Skip next instruction if there is Parity Error, Data Request Late, Write Lock Status, or Non-Existent Disk flag set.

DFSC

Octal code: 6622
Event time: 2
Execution time: 4.25 μsec
Operation: Skip next instruction if the Completion flag is a 1 (data transfer is complete).

DISK

Octal code: 6623
Event time: 1, 2
Execution time: 4.25 μsec
Operation: Skip next instruction if either the Error or Completion flags or both are set.

DMAC

Octal code: 6626
Event time: 2, 3
Execution time: 4.25 μsec
Operation: Clear the accumulator. Then load the contents of the Disk Memory address register into the accumulator to allow program evaluation. This instruction must be issued when the completion flag is set.

DCMA

Octal code: 6601
Event time: 1
Execution time: 4.25 μsec
Operation: Clear the Disk Memory Address register, and all other disk and maintenance flags except interrupt enable.

DMAR

Octal code: 6603
Event time: 1, 2
Execution time: 4.25 μsec
Operation: Load the low order 12 bits of the Disk Memory Address with information (initial address) in the accumulator. Then clear the AC. Begin to read information from the disk into the specified core location. Clear parity error and completion flags. Clear interrupt flags. IOT 6605, see below.

During Read, the final address status is the last address transferred +1.

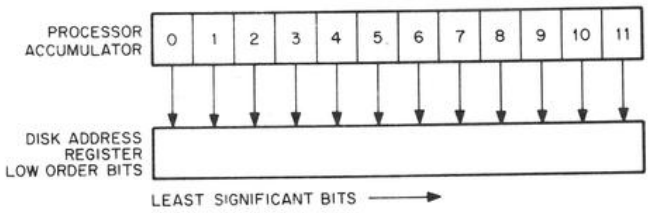
When reading the last address of the last available disk the non-existent flag is raised in coincidence with the completion flag.

DMAW

Octal code: 6605
Event time: 1, 3
Execution time: 4.25 μsec
Operation: Load the low order 12 bits of the Disk Memory Address register with information (initial address) in the accumulator (AC). Then clear the AC. Begin to Write information onto the disk from the specified core location. Clear parity error and completion flags. Clear interrupt flags.

During Write, the final address status is the last address transferred.

Write Lock Switch status is true only when disk module contains a Write Command.

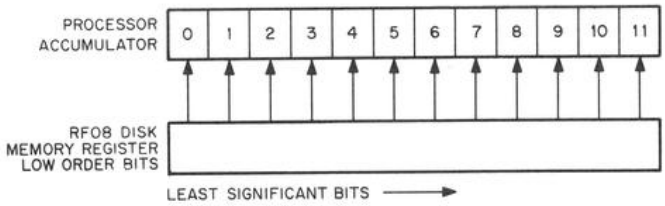


DMAR

Octal code: 6603
Event time: 1.2
Execution time: 4.25 μsec
Operation: If given during photocell time the read or write operation can begin at Disk Memory Address 0000 without latency due to logic-disc sync. time as is the case if given at any other time. However, in all cases, if the read or write operation is to be on an odd track 1, 2, 5, etc. bit 0 of the AC must be in the one state at the time the DMAR or DMAW is given, as bit 0 of the Disk Memory Address is the first bit of the track address.

DMAW

Octal code: 6605
Event time: 1, 4,
Execution time: 4.25 μsec
Operation: If given during photocell time the read or write operation can begin at Disk Memory Address 0000 without latency due to logic-disk sync. time as is the case if given at any other time. However, in all cases, if the read or write operation is to be on an odd track 1, 2, 5, etc. bit 0 of the AC must be in the one state at the time the DMAR or DMAW is given, as bit 0 of the Disk Memory Address is the first bit of the track address.



DMMT

Octal code: 6646
Execution time: 4.25 μsec
Operation: For maintenance purposes only with the appropriate maintenance cable connections and the disk disconnected from the RS08 logic the following standard signals may be generated by IOT 646 and associated AC bits. AC is cleared. The maintenance register is initiated by issuing an IOT 601 command.

AC_{11} (1)	Track *A* Pulse
AC_{10} (1)	Track *B* Pulse
AC_{9} (1)	Track *C* Pulse
AC_{7} (1)	DATA PULSE (DATA HEAD #0)
AC_{6} (1)	+1 Photocell
AC_{0} (1)	+1 DBR

Setting DBR to a 1 causes Data Break Request in computer.

NOTE

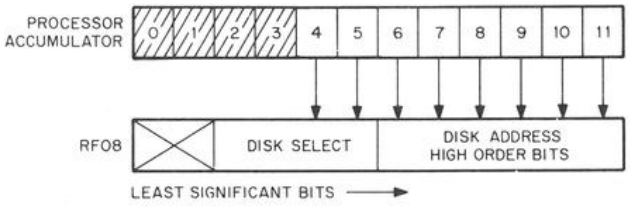
TAP must be generated to strobe Track *B* signal into address comparison network.

DCXA

Octal code: 6641
Execution time: 4.25 μsec
Operation: Clear the high order 8 bit disk address register.

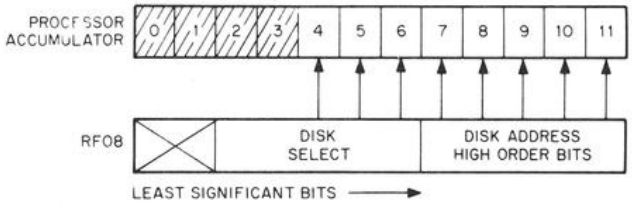
DXAL

Octal code: 6643
Execution time: 4.25 μsec
Operation: Clear the High Order 8 bits of the Address register. Then load the Disk Address register from data held in the accumulator. Then clear the AC.

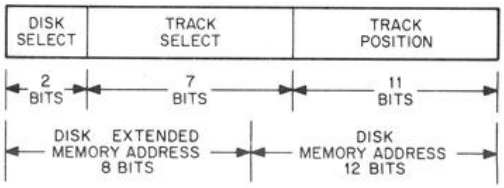


DXAC

Octal code: 6645
Execution time: 4.25 μsec
Operation: Clear the accumulator then load the contents of the High Order 8 bit Disk Address register into the PDP accumulator.



Disk Address Register - 20 bits



Programming Example

A programming example that writes a block of data onto the disk is shown below. For simplicity, the example assumes that all data and instructions are within the same page, but in actual practice, this may not be true.

```
                /CALLING SEQUENCE
SUB             JMS WRT             /JUMP TO WRITE SUBROUTINE
                0                   /CONTAINS WORD COUNT
                0                   /CONTAINS INITIAL CORE MEMORY ADDRESS
                0                   /CONTAINS TRACK AND UNIT NUMBER
                0                   /CONTAINS TRACK ADDRESS
                XXX                 /CONTINUE WITH MAIN PROGRAM

                /WRITE SUBROUTINE
WRT,            0                   /ENTER WIRTE SUBROUTINE
                TAD I WRT           /FETCH WORD COUNT
                DCA WC              /DEPOSIT IN WORD COUNT REGISTER
                ISZ WRT             /INCREMENT POINTER
                TAD I WRT           /FETCH INITIAL CORE MEMORY ADDRESS
                DCA CA              /DEPOSIT INTO CURRENT ADDRESS REGISTER
                ISZ WRT             /INCREMENT POINTER
                TAD I WRT           /FETCH TRACK AND UNIT NUMBER
                DXAL                /DEPOSIT INTO REGISTER IN RF08 CONTROL
                ISZ WRT             /INCREMENT POINTER
                TAD I WRT           /FETCH TRACK ADDRESS
                DMAW                /TRACK ADDRESS IO DMA IN DISK;
                                    /START
                                    /WRITE OPERATION
                DFSC                /JOB DONE?
                JMP .-1             /NO, WAIT
                DFSE                /ANY ERRORS?
                JMP .+2             /NO, SKIP OVER ERROR EXIT
                JMP ERR             /YES, TO ERROR SUBROUTINE
                ISZ WRT             /INCREMENT POINTER TO EXIT ADDRESS
                JMP I WRT           /EXIT PROGRAM
```

The calling subroutine must be set up so that the subsequent locations to SUB (SUB+1, SUB+2, etc.) contain the parameters as shown in the comments column.

The JMS WRT instruction causes a subroutine jump to location WRT with the contents of the PC-1 (which contains symbolic address SUB-1) deposited into location WRT. Since location WRT now contains SUB-1, the first instruction of the subroutine (TAD IWRT) loads the AC with the contents of SUB-1 which is the word count. The word count is then deposited into the WC (Memory Address 7750) register. Similarly, the initial address is deposited into the CA (Memory Address 7751) register. The program then proceeds to set up the EMA and DMA registers and starts the write operation. After the DMAW instruction is issued, the data transfer operation begins and continues independently of the program; it operated under the control of the data break facility to transfer data. When the transfer is complete, the DCF (Data Complete Flag) comes up and, when sensed by the DFSC control, passes to the DFSE instruction. DFSE then senses for errors, and if any, control jumps to an error or diagnostic (not shown) routine. If no errors, control exits from the subroutine back to the main program to resume main processing.

It should be noted that since the data transfer operates independently of the program, the subroutine could be exited following the DMAW instruction. An interrupt subroutine could handle the post data transfer processing since the DCF and ERROR FLAGS generated an interrupt.

An identical program could handle data transfers for a read operation except that the DMAW instruction is replaced by the DMAR instruction.

Specifications:

Storage Capacity	Each RS08 stores 262,144 13-bit words (12 plus one event parity bit)	
Disks	Four RS08's may be controlled by one RF08 for 1,048,576 words.	
Data Transfer Path	3-Cycle Break	Address Locations 7750 Word Count 7751 Memory Address
Data Transfer Rate	60 Hz Power 16.2 μsec per word	50 Hz Power 20 μsec per word
Minimum Access Time	258 μsec	320 μsec
Average Access Time	16.9 msec	20.3 msec
Maximum Access Time	33.6 msec	40.3 msec
Program Interrupt	33 Milliseconds Clock Flag Data Transmission Complete Flag Error Flag	
Write Lock Switches	Eight switches per disk capable of locking out any combination of eight 16,384 word blocks in addresses 0 to 131,071.	
Data Tracks	128	
Words Per Track	2048	
Recording Method	NRZI	
Density	1100 BPI Maximum	
Timing Tracks	3 plus 3 spare	
Operating Environment	Recommended temperature 65° to 90°F. Relative humidity 20% to 80%. No condensation, storage or operating.	
Vibration/Shock	Good isolation is provided. Vibrating, shaking or rocking of the cabinet with large, low frequency displacements can cause data errors. For example, hand fork lift trucks operating on wooden floors cause excessive vertical displacements which could cause errors. The RS08 is not designed for aircraft of shipboard mounting.	
Heat Dissipation:	RF08: 150 watts RS08: 500 watts	
A.C. Power Requirements:	115 ±10 VAC, single phase, 50 ±2 or 60 ±2Hz	

Specifications (cont)

RS08:	Motor start 5.5 amps for 20 ±3 sec. Motor run 4.0 amps continuous.
Line Frequency Stability:	Maximum line frequency drift 0.1 Hz/sec. A constant frequency motor-generator set or static AC/AC inverter should be provided for installations with unstable power sources.
Reliability:	Six recoverable errors and one non-recoverable error in 2×10^9 bits transferred. A recoverable error is defined as an error that occurs only once in four successive reads. All other errors are non-recoverable. On-off cycling of the RS08 is not recommended. The RS08 motor control operates independently of the computer power control, thus eliminating on-off cycling except for power failures.
Cabinet:	A dedicated H950 cabinet is designed to accommodate one RF08, up to two RS08's and power supplies. Two additional RS08's can be mounted in a second H950. Other equipment should not be mounted in disk cabinets.

5.4.11 Automatic Magnetic Tape Control, Type TC58

Functional Description

The Type TC58 will control the operation of a maximum of eight digital magnetic tape transports, Types TU20 and TU20A. The Type TC58 interfaces to and uses the PDP-12 3-cycle data break facility for data transfer directly to or from system core memory and magnetic tape. The tape transports offer industry-compatible (or IBM-compatible) in both 7 and 9 channel tape transports with the following characteristics:

Transport	*Tape Speed (ips)*	*Densities (bpi)*
TU20 (7-channel)	45	200/556/800
TU20A (9-channel)	45	800

Transfers are governed by the in-memory word count (WC) and current address (CA) register associated with the assigned data channel (memory locations 32_8 and 33_8). Since the CA is incremented before each data transfer, its initial contents should be set to the desired initial address minus one. The WC is also incremented before each transfer and must be set to the 2's complement of the desired number of data words to be transferred. In this way, the word transfer which causes the word count to overflow (WC becomes zero) is the last transfer to take place. The number of IOT instructions required for the TYPE TC58 is minimized by transferring all necessary control data (i. e., unit number, function, mode, direction, etc.) from the PDP-12 accumulator (AC) to the control using IOT instructions. Similarly, all status information (i, e., status bits, error flags, etc.) can be read into the AC from the control unit by IOT instructions.

During normal data reading, the control assembles 12-bit computer words from successive frames read from the information channels of the tape. During normal data writing, the control disassembles 12-bit words and distributes the bits so they are recorded on successive frames of the information channels.

Instructions

The commands for the Magnetic Tape Control System are as follows:

MTSF Skip on Error Flag or Magnetic Tape Flag

Octal code: 6701
Event time: 1
Execution time: 4.25 μsec
Operation: The status of the error flag (EF) and the magnetic tape flag (MTF) are sampled. If either or both are set to 1, the content of the PC is incremented by one to skip the next sequential instruction.
Symbol: If MTF or EF = 1, PC + 1 = > PC

MTCR Skip on Tape Control Ready

Octal code: 6711
Event time: 1

Skip on Tape Control Ready (cont)

Execution time: 4.25 μsec
Operation: If the tape control is ready to receive a command, the PC is incremented by one to skip the next sequential instruction.
Symbol: If Tape Control Ready, PC + 1 = > PC

MTTR Skip on Tape Transport Ready

Octal code: 6721
Event time: 1
Execution time: 4.25 μsec
Operation: The next sequential instruction is skipped if the tape transport is ready.
Symbol: If tape unit ready, PC + 1 = > PC

MTAF Clear Registers, Error Flag and Magnetic Tape Flag

Octal code: 6712
Event time: 2
Execution time: 4.25 μsec
Operation: Clears the status and command registers, and the EF and MTF if tape control is ready. If tape control not ready, clears MTF and EF flags only.
Symbol: If tape control is ready, 0 = > MTF, 0 = > EF,
0 = > command register
If tape control not ready, 0 = > MTF, 0 = > EF

Inclusive OR Contents of Command Register

Octal code: 6724
Event time: 3
Execution time: 4.25 μsec
Operation: Inclusively OR the contents of the command register into bits 0-11 of the AC.
Symbol: AC V command register = > AC

MTCM Inclusive OR Contents of Accumulator

Octal code: 6714
Event time: 3
Execution time: 4.25 μsec
Operation: Inclusively OR the contents of AC bits 0-5, 9-11 into the command register; JAM transfer bits 6, 7, 8 (command function).
Symbol: AC_{0-5}, AC_{9-11} V command register = > command register
AC_{6-8} = > command register bits 6-8

MTLC Load Command Register

Octal code: 6716
Event time: 2, 3
Execution time: 4.25 μsec
Operation: Load the contents of AC bits 0-11 into the command register.
Symbol: AC_{0-11} = > command register

Inclusive OR Contents of Status Register

Octal code:	6704
Event time:	3
Execution time:	4.25 μsec
Operation:	Inclusively OR the contents of the status register into bits 0-11 of the AC.
Symbol:	Status Register V AC = > AC_{0-11}

MTRS Read Status Register

Octal code:	6706
Event time:	2, 3
Execution time:	4.25 μsec
Operation:	Read the contents of the status register into bits 0-11 of the AC.
Symbol:	Status Register = > AC_{0-11}

MTGO Mag Tape GO

Octal code:	6722
Event time:	2
Execution time:	4.25 μsec
Operation:	Set *GO* bit to execute command in the command register if command is legal.
Symbol:	None

Clear the AC

Octal code:	6702
Event time:	2
Execution time:	4.25 μsec
Operation:	Clear the accumulator.
Symbol:	0 = > AC

Although any number of tapes may be simultaneously rewinding, data transfer may take place to or from only one transport at any given time. In this context, data transfer includes these functions: read or write data, write EOF (end of file), read/compare, and space. When any of these functions are in process, the tape control is in the *not ready* condition. A transport is said to be *not ready* when tape is in motion, when transport power is off, or when it is off line.

Data transmission may take place in either parity mode, odd-binary or even-BCD. When reading a record in which the number of characters is not a multiple of the number of characters per word, the final characters come into memory left-justified.

Ten bits in the magnetic tape status register retain error and tape status information. Some error types are combinations, such as lateral and longitudinal parity errors (parity checks occur after both reading and writing of data), or have a combined meaning, such as illegal command, to allow for the maximum use of the available bits.

The magnetic tape status register reflects the state of the currently selected tape unit. Interrupts may occur only for the selected unit. Therefore, other units which may be rewinding, for example, will not interrupt when done.

A special feature of this control is the Write Extended Inter-Record Gap capability. This occurs on a write operation when Command Register bit 5 is set. The effect is to cause a 3-inch inter-record gap to be produced before the record is written. The bit is automatically cleared when the writing begins.

This is very useful for creating a 3-inch gap of blank tape over areas where tape performance is marginal.

Magnetic Tape Functions

For all functions listed below, upon completion of the data operation (after the end-of-record character passes the read head), the MTF (magnetic tape flag) is set, an interrupt occurs (if enabled), and errors are checked.

No Operation - A NO OP command defines no function in the command register. A MTGO instruction with NO OP will cause an illegal command error (set EF).

Space - There are two commands for spacing records, SPACE FORWARD and SPACE REVERSE. The number of records to be spaced (2's complement) is loaded into the WC. CA need not be set. MTF (magnetic tape flag) is set, and an interrupt occurs at WC overflow, EOF (end of file), or EOT (end of tape), whichever occurs first. When issuing a space command, both the density and parity bits must be set to the density and parity in which the records were originally written.

Load Point or Beginning of Tape (BOT) detection during a backspace terminates the function with the BOT bit set. If a SPACE REVERSE command is given when a transport is set at BOT, the command is ignored, the illegal command error and BOT bits are set, and an interrupt occurs.

Read Data - Records may be read into memory only in the forward mode. Both CA and WC must be set: CA, to the initial core address minus one; WC, to the 2's complement of the number of words to be read. Both density and parity bits must be set.

If WC is set to less than the actual record length, only the desired number of words are transferred into memory. If WC is greater than or equal to the actual record length, the entire record is read into memory. In either case, both parity checks are performed, the MTF is set, and an interrupt occurs when the end-of-record mark passes the read head. If either lateral or longitudinal parity errors or bad tape have been detected, or an incorrect record length error occurs (WC not equal to the number of words in the record) the appropriate status bits are set. An interrupt occurs only when the MTF is set.

To continue reading without stopping tape motion, MTAF (clear MTF) and MTGO instructions must be executed. If the MTGO command is not given before the shut down delay terminates, the transport will stop.

Write Data - Data may be written on magnetic tape in the FORWARD DIRECTION ONLY. For the WRITE DATA function, the CA and WC registers and density and parity bits must be set. WRITE DATA is controlled by the WC, such that when the WC overflows, data transfer stops, and the EOR (end of record) character and IRG (inter-record gap) are written. The MTF is set after the EOR has passed the read head. To continue writing, a MTGO command must be issued before the shut down delay terminates. If any errors occur, the EF will be set when the MTF is set.

Write EOF - The WRITE EOF command transfers a single character (17_8) record to magnetic tape and follows it with EOR character. CA and WC are ignored for WRITE EOF. The density bits must be set, and the command register parity bit should be set to even (BCD) parity. If it is set to odd parity, the control will automatically change it to even.

When the EOF marker is written, the MTF is set and an interrupt occurs. The tape transport stops, and the EOF status bit is set, confirming the writing of EOF. If odd parity is required after a WRITE EOF, it must be specifically requested through the MTLC command.

Read/Compare - The READ/COMPARE function compares tape data with core memory data. It can be useful for searching and positioning a magnetic tape to a specific record, such as a label or leader, whose content is known in core memory, or to check a record just written. READ/COMPARE occurs in the forward direction only; CA and WC must be set. If there is a comparison failure, incrementing of the CA ceases, and the READ/COMPARE error bit is set in the status register. Tape motion continues to the end of the record; the MTF is then set and an interrupt occurs. If there has been a READ/COMPARE error, examination of the CA reveals the word that failed to compare.

Rewind - The high speed REWIND command does not require setting of the CA or WC. Density and parity settings are also ignored. The REWIND command rewinds the tape to loadpoint (BOT) and stops. Another unit may be selected after the command is issued and the rewind is in process. MTF is set, and an interrupt occurs (if the unit is selected) when the unit is ready to accept a new command. The selected unit's status can be read to determine or verify that REWIND is in progress.

Continued Operation

1. To continue operating in the same mode, the MTGO instruction is given before tape motion stops. The order of commands required for continued operation are as follows:

   a. MTLC, if the command is to be changed.
   b. MTAF, will only clear MTF and EF flags since tape control will be in a Not Ready state.
   c. MTGO, if MTLC requested an illegal condition, the EF will be set at this time.

2. To change modes of operation, either in the same or opposite direction, the MTLC command is given to change the mode and a MTGO command is given to request the continued operation of the drive. If a change in direction is ordered, the transport will stop, pause, and automatically start up again.

3. If the WRITE function is being performed, the only forward change in command that can be given is WRITE EOF.

4. If no MTGO instruction is given, the transport will shut down in the inter-record gap.

NOTE

No flags will be set when the control becomes ready or the transport becomes ready, except if the REWIND command is present in the command register and the selected drive reaches BOT and is ready for a new command.

5. If a WRITE (odd parity) command is changed to WRITE EOF, the parity is automatically changed to even.

NOTE

Even parity will remain in the command register unless changed by a new command instruction, MTLC, which clears and loads the entire command register.

Status or Error Conditions

Twelve bits in the magnetic tape status register indicate status or error conditions. They are set by the control and cleared by the program.

The magnetic tape status register bits are:

*Bit**	*Function (When Set)*
0	Error flag (EF)
1	Tape rewinding
2	Beginning of tape (BOT)
3	Illegal command
4	Parity error (Lateral or Longitudinal)
5	End of file (EOF)
6	End of tape (EOT)
7	Read/compare error
8	Record length incorrect WC = 0 (long) WC ≠ 0 (short)
9	Data request late
10	Bad tape
11	Magnetic tape flag (MTF) or job done

MTF (SR11) - The MTF flag is set under the following conditions:

1. Whenever the tape control has completed an operation (after the EOR mark passes the read head).

2. When the selected transport becomes ready following a normal REWIND function.

These functions will also set the EF if any errors are present.

EOF (SR5) - End-of-file (EOF) is sensed and may be encountered for those functions which come under the heading of READ STATUS FUNCTION, i. e., SPACE, READ DATA, or READ/COMPARE and WRITE EOF. When EOF is encountered, the tape control sets EOF = 1. MTF is also set; hence, an interrupt** occurs and the EOF status bit may be checked.

EOT (SR6) and BOT (SR2) - End-of-tape (EOT) detection occurs during any forward command when the EOT reflective strip is sensed. When EOT is sensed, the EOT bit is set, but the function continues to completion. At this time the MTF is set (and EF is set), and an interrupt occurs.

Beginning-of-tape (BOT) detection status bit occurs only when the beginning-of-tape reflective strip is read on the transport that is selected.

When BOT detection occurs, and the unit is in reverse, the function terminates. If a tape unit is at load point when a REVERSE command is given, an illegal command error bit is set, causing an EF with BOT set. An interrupt then occurs.

*The register bits are equivalent in position to the AC bits (i. e., $SR_0 = AC_0$, etc.).

**All references to interrupts assume the tape flags have been enabled to the interrupt (command register bit 9 = 1) and that the unit is selected.

Illegal Command Error (SR3) - The illegal command error bit is set under the following conditions:

1. A command is issued to the tape control with the control not ready.

2. A MTGO command is issued to a tape unit which is not ready, and the tape control is ready.

3. Any command which the tape control, although ready, cannot perform; e. g.:

    a. WRITE with WRITE LOCK condition
    b. 9-channel tape and incorrect density
    c. BOT and SPACE REVERSE

Parity (SR4) - Longitudinal and lateral parity checks will occur in both reading and writing. The parity bit is set for either lateral or longitudinal parity failure. A function is not interrupted, however, until MTF is set. Maintenance panel indicators are available to determine which type of parity error occurred.

Read Compare Error (SR7) - When READ/COMPARE function is underway, SR7 is set to 1 for a READ/COMPARE ERROR (see earlier section on READ/COMPARE for further details).

Bad Tape (SR10) - A BAD TAPE ERROR indicates detection of a bad spot on the tape. Bad tape is defined as three or more consecutive missing characters followed by data, within the period defined by the READ SHUTDOWN DELAY. The error bit is set by the tape control when this occurs. MTF and interrupt do not occur until the end of the record in which the error was detected.

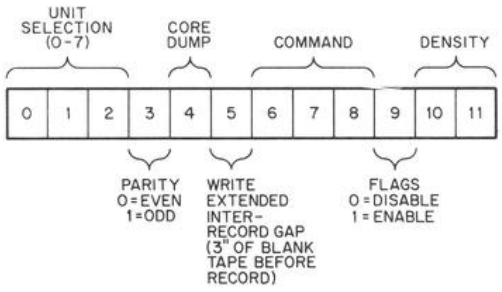
Tape Rewinding (SR1) - When a REWIND command has been issued to a tape unit and the function is underway, the tape rewinding bit is set in the control. This is a transport status bit, and any selected transport which is in a high speed rewind will cause this bit to be set.

Record Length Incorrect (SR8) - During a read or read/compare, this bit is set when the WC overflow differs from the number of words in the record. The EF flag is set.

Data Request Late (SR9) - This bit can be set whenever data transmission is in progress. When the DATA FLAG causes a break cycle, the data must be transmitted before a write pulse or a read pulse occurs. If it does not, this error will occur, and data transmission will cease. The EF flag and bit 9 of the status register are set when the MTF is set.

Error Flag (SR0) - The ERROR FLAG (EF) is set whenever any error status bit is present at the time that MTF is set. However, when an ILLEGAL COMMAND is given, the EF is set and the MTF is not set.

Command Register Contents



Unit Selection

	Bits		
Unit	0	1	2
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

Density Selection

Density	Bits 10-11	
200 BPI	0	0
556 BPI	0	1
800 BPI	1	0
800 BPI 9 channel	1	1

Command Selection

Command	Bits 6	7	8
NO OP	0	0	0
Rewind	0	0	1
Read	0	1	0
Read/Compare	0	1	1
Write	1	0	0
Write EOF	1	0	1
Space Forward	1	1	1
Space Reverse	1	1	1

Magnetic Tape Function Summary

LEGEND:

CA	=	Current Address Register = 33_8
WC	=	Word Count Register = 32_8
F	=	Forward
R	=	Reverse
DS	=	Density Setting
PR	=	Parity Setting
EN	=	Enable Interrupt

Function	*Characteristics*	*Status or Error Types*
NO-OP	CA: Ignored WC: Ignored DS: Ignored PR: Ignored EN: Ignored	Illegal BOT Tape Rewinding
SPACE FORWARD	CA: Ignored WC: 2's comp. of number of records to skip DS: Must be set PR: Must be set EN: Must be set	Illegal EOF Parity Bad Tape MTF, BOT, EOT
SPACE REVERSE	Same as SPACE FORWARD	Illegal EOF Parity Bad Tape BOT MTF
READ DATA	CA: Core Address–1 WC: 2's comp. of number of words to be transferred DS: Must be set PR: Must be set EN: Must be set	Illegal EOF Parity Bad Tape MTF EOT Data Request Late Record Length Incorrect
WRITE DATA	Same as READ DATA	Illegal EOT Parity MTF Bad Tape Data Request Late
WRITE EOF	CA: Ignored WC: Ignored DS: Must be set PR: Must be set EN: Must be set	Same as WRITE DATA plus EOF
READ/COMPARE	Same as READ DATA	Illegal EOF Read/Compare Error Bad Tape

Function	*Characteristics*	*Status of Error Types*
READ/COMPARE (cont.)		MTF EOT Data Late Record Length Incorrect
REWIND	CA: Ignored WC: Ignored DS: Ignored PR: Ignored EN: Must be set	Illegal Tape Rewinding MTF BOT

Magnetic Tape Transport, Type TU20 (7-Channel)

The Type TU20 is a digital magnetic tape transport designed to be compatible with the Type TC58 Magnetic Tape Control. The transport operates at a speed of 45 inches per second and has three selectable densities: 200, 556, and 800 bpi. The maximum transfer rate is 36,000 six-bit characters per second. Standard seven-channel IBM-compatible tape format is used. The specifications for the unit are as follows:

Format: NRZI. Six data bits plus one parity bit. End and loadpoint sensing compatible with IBM 729 I-VI.

Tape: Width of 0.5 inch length of 2400 ft. (1.5 mil.). Reels are 10.5 inches, IBM-compatible, with file protect (WRITE LOCK) ring.

Head: Write-read gap of 0.300 inch Dynamic and static skew is less than 14 μsec.

Tape Specifications: 45 IPS speed. Start time is less than 5 msec. Start distance is 0.080 in. (+0.035, -0.025 in.). Stop time is less than 1.5 msec. Stop distance is 0.045 in. ($\pm$0.05 in.).

Density: 200, 556, and 800 BPI. Maximum transfer rate is 36 kHz.

Transport Mechanism: Pinch roller drive; vacuum column tension.

Controls: ON/OFF, ON LINE, OFF LINE, FORWARD, REVERSE, REWIND, LOAD, RESET.

Physical Specifications: Width of 22 1/4 in., depth of 27 1/6 in., height of 69 1/8. Weight–600 lbs.

Read (Read/Compare) Shutdown Delay: 3.6 milliseconds.

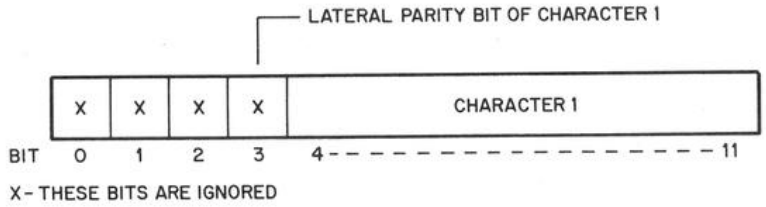
Write Shutdown Delay: Approximately 4.5 milliseconds.

9-Track Operation

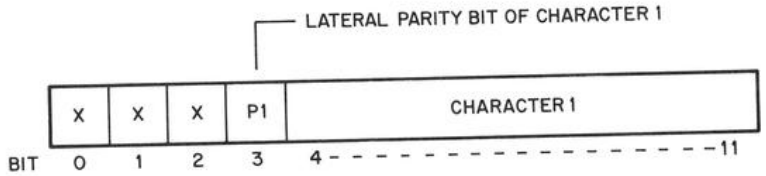
9- and 7-track transports may be intermixed on the Type TC58 control. When a transport is selected, it automatically sets the control for proper operation with its number of tracks.

Control of 9-track operation is identical to 7-track, except as noted below:

Write - A word in memory is written on tape with format shown below:



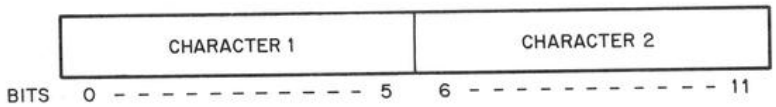
Read - A word is read into memory from tape with the format shown below:



Read/Compare - A direct comparison of the characters on tape is made with those in memory. The parity bit is ignored, as are bits 0-3 in each memory word.

Core Dump Mode - This mode is used only with 9-track transports. It is entered by setting bit 4 of the command register.

Core dump mode permits the dumping of complete memory words in the form of two six-bit characters. The format is:



This is accomplished by only utilizing 7 of the 9 tracks on the tape.

Tape written in CORE DUMP MODE, must be READ (READ/COMPARE) in the same mode. These operations are the same as for a 7-track transport.

Magnetic Tape Transport, Type TU20A (9-Channel)

The Type TU20A is a digital magnetic tape transport designed to be compatible with the Type TC58 Magnetic Tape Control. The transport operates at a speed of 45 inches per second and a density of 800 bpi. The maximum transfer rate is 36,000 eight-bit characters per second. Standard nine-channel IBM-compatible tape format is used. The specifications for the unit are as follows:

Format: NRZI. Eight data bits plus on parity bit. End and loadpoint sensing compatible with IBM.

Tape: Width of 0.5 inch length of 2400 ft. (1.5 mil.). Reels are 10.5 in., IBM-compatible, with file protect (WRITE LOCK) ring.

Heads: Write-read gap of 0.150 in. Dynamic and static skew is less than 14 μsec.

Tape Specifications: 45 IPS speed. Rewind time is less than 5 msec. Start distance is 0.080 in. (+0.035, -0.025 in.). Stop time is less than 1.5 msec. Stop distance is 0.045 in. (±0.015 in.).

Density: 800 BPI. Maximum transfer rate is 36 kHz.

Transport Mechanism: Pinch roller drive; vacuum column tension.

Controls: ON/OFF, ON LINE, OFF LINE, FORWARD, REVERSE, REWIND, LOAD, RESET.

Physical Specifications: Width of 22 1/4 in., depth of 27 1/6 in., height of 69 1/8 in. Weight–600 lbs.

Read (READ/COMPARE) Shutdown Delay: 3.6 milliseconds.

Write Shutdown Delay: Approximately 4.5 milliseconds.

5.4.12 General Purpose Multiplexed Analog-to-Digital Converter System (Type AF01A)

The Type AF01A General-Purpose Multiplexed Analog-to-Digital Converter combines a versatile, multi-purpose converter with a multiplexer to provide a fast, automatic, multichannel scanning and conversion capability. It is intended for use in systems in which computers sample and process analog data from sensors or other external signal sources at high rates. The AF01A is used when greater accuracy than provided by the standard AD12 A-D converter is needed. The Type AF01A option is used with the PDP-12 to multiplex up to 64-analog signals and to convert the signals to binary numbers. Analog data on each of 64 channels can be accepted and converted into 12-bit digital numbers 420 times per second.*

Switching point accuracy in this Instance is 99.975 per cent, with an additional quantization error of half the digital value of the LSB.

*Conversion rate = $[(35 + 2)\ 10\text{-}^{6})\ (64)]^{-1}$ = 420 cycles/sec.
= $[(9 + 2)\ (10\text{-}^{6})\ (64)]^{-1}$ = 1420 cycles/sec.

A/D Converter Specifications

The Type AF01A has a successive approximation converter that measures a 0 to −10 volt analog input signal and provides a binary output indication of the amplitude of the input signal. The characteristics of the A/D converter are as follows:

Accuracy and Conversion Times:	See Table 5-4 (includes all linearity and temperature errors)
Converter Recovery Time:	Zero
Input and Input Impedance:	0 to -10v at 10 megohms standard. Input scaling may be specified using the amplifier or sample and hold options (see Table 1)
Input Loading:	±1 μA and 125 pf for 0 to -10v input signal.
Output:	Binary number of 6 to 12 bits, with negative numbers represented in 2's complement notation. A 0v input gives a 4000_8; a -5v input a 0000_8 and a -10v (minus 1 LSB*) input gives 3777_8 number. *LSB, least significant bit

Provision is made for using the Type A400 Sample and Hold Amplifier (AH02 option) between the multiplexer output and A/D converter input to reduce the effective aperture to less than 150 nsec. The Type A400 may also be used to scale the signal input to accept ±10v, ±5v, or 0 to +10v. The Type A200 amplifier (AH03 option) may be substituted for the Type A400 to accomplish the same signal scaling without reducing the effective aperture. Both the AH02 and AH03 options may be used to obtain high input impedance and small aperture. (See Table 5-3.)

Multiplexer Specifications

The multiplexer can include from 1 to 16 Type A121 Switch Modules. Each module contains four single-pole, high speed, insulated gate FET switches with appropriate gating. The Type A121 Switches are arranged as a 64-channel group of series-switch single-pole switches with a separate continuous ground wire for each signal input. The switched signal input wire and the continuous ground for each channel are run as twisted pairs to the input connectors mounted on the rear panel. The continuous grounds for all channels are terminated at the high quality ground of the AF01A System. Specifications (measured at input connector) are as follows:

Input Operating Signal Voltages:	+ 10v to -10v
Current:	1 mA
On Resistance	450 ohm (max)
Voltage Offset	0
"Off Leakage"	10 nA (max)
Capacitance	10 pf (max)
Speed 10% Input to within 1 LSB* of output	2 μs

Operate Time — The time required to switch from one channel to another is 2 μs to within 1 LSB of the final voltage. This time is preset within the control and starts when a set or index command is received.

Operation

The Type AF01 System may be operated in either the random or sequential address modes. In the random address mode, the control routes the analog signal from any selected channel to the A/D converter input. In the sequential address mode, the multiplexer control advances its channel address by one each time an index command is received. After indexing through the maximum number of channels implemented, the address is returned to 0. When using sequential operation, the conditioning levels for random addressing are ignored.

The multiplexer switch settling time is preset within the control to initiate the conversion process automatically after a channel has been selected in either the random or sequential address mode. Two separate A/D Convert I/O Transfer Commands may also initiate one or more conversions on a currently selected channel.

A/D conversion times are increased by 2 μsec when multiplexer channels are switched to allow for settling time of the analog signal at the multiplexer output. Conversion times are increased an additional 3 μsec when AH03 is used. These times are added to the conversion times shown in Table 5-4 under selected channel conversion time which is the only time required for each successive conversion on a selected channel.

When the Type AH02 Sample and Hold option is required, the multiplexer switch settling time and the sample and hold acquisition time are overlapped. The total conversion and switching time is increased by 10 μsec. (See A400 specifications).

A/D CONVERTER/MULTIPLEXER CONTROLS

DESIGNATION	FUNCTION
WORD LENGTH:	Rotary switch used to select digital word length or conversion accuracy. Refer to Table 5-4 for corresponding conversion times.
POWER ON/OFF:	Applies 117 Vac power to internal power supplies.
CLR	Clear multiplexer channel-address registers; i. e., selects analog channel 0 for conversion.

INDEX:	Advances multiplexer channel-address register by one each time it is depressed, enabling manual addressing of channels (up to 64) in sequential mode. Returns address to zero when maximum value is reached.
ADC:	Starts conversion of the analog voltage on the selected channel to a binary number when depressed.
A/D CONVERTER:	Indicates binary contents of A/D converter register.
MULTIPLEXER	Indicates binary contents of multiplexer channel-address register.
POWER:	Indicates ON/OFF status.

TABLE 5-3. INPUT SIGNAL SCALING

CONFIGURATION	GAIN	INPUT SIGNAL	INPUT IMPEDANCE	BINARY OUTPUT	OPTION DESIGNATION
Standard		0	10 meg.	4000_8	
		−5	10 meg.	0000_8	STD
		−10	10 meg.	3777_8	
Sample &	−1	+5	10K	3777_8	
Hold	−1	0	10K	0000_8	AHO2
	−1	−5	10K	4000_8	
Sample &	−½	+10	10K	3777_8	
Hold	−½	0	10K	0000_8	AHO2
	−½	−10	10K	4000_8	
	+1	+5	>100 meg.	4000_8	
Amplifier	+1	0	>100 meg.	0000_8	AHO3
	+1	−5	>100 meg.	3777_8	
	+½	+10	>100 meg.	4000_8	
Amplifier	+½	0	>100 meg.	0000_8	AHO3
	+½	−10	>100 meg.	3777_8	
Amplifier	−1	+5 +10	>100 meg.	3777_8	AHO3
and Sample	or	0 or 0	>100 meg.	0000_8	&
& Hold	−½	−5 −10	>100 meg.	4000_8	AHO2

Note: Unipolar signals (0 to +5, or 0 to +10v) may also be specified with either the AHO3 or AH02 option.

TABLE 5-4. SYSTEM CONVERSION CHARACTERISTICS**

		Selected Channel (A/D)	Random or Sequential (MPX & A/D)	AHO3 MPX A/D	AHO2 MPX A/D	AHO2 AHO3 MPX &A/D
Word Length (No. of Bits)	Max Switching Point Error*	Conversion Time (μsec)	Conversion Time (μsec)**	Conversion Time (μsec)**	Conversion Time (μsec)**	Conversion Time (μsec)**
6	±1.6%	9.0	11.0 (9.5)	14.0 (11.0)	19.0 (14.0)	21.0 (18.0)
7	±0.8%	10.5	12.5 (11.0)	15.5 (12.5)	20.5 (15.5)	22.5 (19.5)
8	±0.4%	12.0	14.0 (12.5)	17.0 (14.0)	22.0 (17.0)	24.0 (21.0)
9	±0.2%	13.5	15.5 (14.0)	18.5 (15.5)	23.5 (18.5)	25.5 (22.5)
10	±0.1%	18.0	20.0 (18.5)	23.0 (20.0)	28.0 (23.0)	30.0 (27.0)
11	±0.05%	25.0	27.0	30.0	35.0	37.0
12	±0.025%	35.0	37.0	40.0	45.0	47.0

*±½ LSB for quantizing error.
**If system is to operate at less than 10 bits continuously, conversion times may be reduced to times shown in parentheses.

Programming

Programmed control of the converter/multiplexer by the PDP-12 is accomplished with the IOT instructions listed below. PDP-12 selects the converter/multiplexer with two device selection codes, depending upon whether conversion of multiplexing functions are being selected; 53_8 and 54_8. The converter/multiplexer interprets the device selection code to enable execution of the IOP command pulse generated by the IOT instruction.

ADSF Skip on A-D Flag

Octal code: 6531
Event time: 1
Execution time: 4.25 μsec
Operation: The A-D converter flag is sensed, and if it contains a binary 1 (indicating that the conversion is complete) the content of the PC is incremented by one so that the next instruction is skipped.
Symbol: If A-D Flag = 1, then PC + 1 = > PC

ADCV Convert Analog Voltage to Digital Value

Octal code: 6532
Event time: 2
Execution time: This time is a function of the accuracy and word length switch setting as listed in Table 2.
Operation: The A-D converter flag is cleared, the analog input voltage is converted to a digital value, and then the A-D converter flag is set to 1. The number of binary bits in the digital-value word and the accuracy of the word is determined by the preset switch position.
Symbol: 0 = > A-D Flag at start of conversion, then
1 = > A-D Flag when conversion is done.

ADRB Read A-D Converter Buffer

Octal code: 6534
Event time: 3
Execution time: 4.25 μsec
Operation: The converted number contained in the converter buffer (ADCB) is transferred into the AC left justified; unused bits of the AC are left in a clear state, and the A-D converter flag is cleared. This command must be preceded by a CLA instruction.
Symbol: ADCB = > AC
0 = > A-D Converter Flag

ADCC Clear Multiplexer Channel

Octal code: 6541
Event time: 1
Execution time: 4.25 μsec
Operation: The channel address register (CAR) of the multiplexer is cleared in preparation for setting of a new channel.
Symbol: 0 = > CAR

ADSC Set Multiplexer Channel

Octal code: 6542
Event time: 2
Execution time: 4.25 μsec
Operation: The channel address register of the multiplexer is set to the channel specified by bits 6 through 11 of the AC. A maximum of 64 single-ended input channels can be used.

ADIC Increment Multiplexer Channel

Octal code: 6544
Event time: 3
Execution time: 4.25 μsec
Operation: The content of the channel address register of the multiplex is incremented by one. If the maximum address is contained in the register when this command is given, the minimum address (00) is selected.
Symbol: CAR + 1 = > CAR

The converter/multiplexer may be operated by the computer program in either the random or sequential addressing mode. In the random addressing mode, the analog channel is selected arbitrarily by the program for digitizing and the resultant binary word is read into the accumulator. A sample program for the random addressing mode is as follows:

```
TAD ADDR        /YES–GET CHANNEL ADDRESS
ADSC            /AND SEND TO MULTIPLEXER
ADCV            /CONVERT A TO D
ADSF            /SKIP ON A/D DONE FLAG
JMP. –1         /WAIT FOR FLAG
ADRB            /AND READ INTO AC
```

In the sequential address mode, the program advances the multiplexer channel-address register to the next channel each time an analog value is converted and read into the accumulator.

Should the converter/multiplexer be operated in the interrupt mode, the computer will be signaled each time that a binary word is ready, enabling the system to use processor time more efficiently.

Amplifier, Sample and Hold Options for AF01A

The AHO3 consists of a DEC amplifier (part # 1505379) mounted on an A990 Amplifier Board with appropriate scaling networks and gain trim and balance potentiometers.

Open loop gain	2×10^6
Rated output voltage	(@ 10 ma) ±11v
Frequency response	
Unity Gain, small signal	10 mc
Full output voltage	300 kc
Slewing rate	30v/μsec
Overload recovery	200 μsec
Input voltage offset	Adjustable to 0
Avg vs temp	20 μv/°C
Vs supply voltage	15 μv/%
Vs time	10 μv/day
Input current offset	±2 na
Avg vs temp	0.4 na/°C
Vs supply voltage	0.15 na/%
Input impedance	
Between inputs	6 meg
Common mode	500 meg
Input voltage	±15v
Max common mode	±10v
Common mode rejection	20,000
Power	
Voltage	±15 to 16v
Current at rated load	35 ma

A400 (standard gain options)

Acquisition time to 0.01% (full-scale step)	<12 μsec
Aperture time	<150 nsec
Hold inaccuracy (droop)	<1 mv/msec
Temperature coefficient	0.1 mv/msec/°C
Gain (negative)	1.0 0.5
Input range (volts)	±5.0 ±10.0
Impedance	10K 10K
Output voltage	0 to −10v
Impedance	<1.0 ohm

5.4.13 Guarded Scanning Digital Voltmeter (Type AF04A)

Description

The Type AFO4A is a guarded scanning digital voltmeter system, with wide dynamic range and high common-mode rejection, and fully capable of expansion to 1000 channels. The Type AFO4A is used with a PDP-12 computer to multiplex up to 1000 3-wire analog channels into a 6-decimal-digit (BCD) integrating digital voltmeter. Full scale ranges are from ±10 mv to ±300v, with automatic ranging, 300 percent over ranging, and a usable 5 μv resolution. Guarded input construction and active integration assist in attaining an effective common-mode rejection of greater than 140 db at all frequencies. (Normal-mode rejection is infinite at multiples of power line frequency.)

This system is ideally suited for data acquisition or process monitoring where a wide range of signals requires large dynamic range. The 10-mv range has 0.001 percent resolution and, coupled with excellent noise rejection, allows accurate direct measurement of thermocouples, strain gauges, load cells, and other low-level transducers without additional amplification.

The AFO4A Voltmeter, operated under program control, is capable of either random channel selection or sequential channel selection. The computer selectts either program controlled ranging (for fastest speed) or autoranging, as well as the integration time of the integrating digital voltmeter (IDVM). The digitized data, as well as the currentt channel address, is read by the computer in either two or three bytes.

A decimal display of the digitized value, including sign and decimal location, is continuously displayed on the front panel. The current channel number is also displayed. Front-panel controls on the digital voltmeter allow manual settinng of all the programmed functions. A front-panel control allows continuous display of the internal secondary standard, which can be prewired to a particular channel for reference checking during normal operation. The AFO4A Voltmeter System may be manually controlled, completely independent of the computer.

SPECIFICATIONS

Full scale ±	10mv, 100mv, 1v, 10v, 100v, 300v, and automatic ranging
Over ranging	300% on all but highest range
Resolution	5 μv (usable), 0.1 μv (LSB)
Accuracy (overall worst case with daily calibration at calibration temperature)	±0.004% of reading ±0.01% of full scale±5μv
Stability (RMS full scale and zero drift)	±0.006%/day
Temperature coefficient	±0.003% of reading/°C
Full scale	±0.002% of full scale/°C
Zero	(±0.006% of full scale/°C on 10 mv and 1v range)
Line voltage stability	±0.0005%/10% change
Maximum common-mode voltage	±300v from power line ground
Common-mode rejection (166.6 msec integration period and 1000 ohm-source unbalance)	> 140 db at all frequencies

Normal-mode rejection	Infinite at multiples of line frequency
Input impedance	
10, 100, 1000 mv ranges	1000 meg/v
Internal secondary standard	
Value	±1.000v
Accuracy	±0.002% traceable to N.B.S.
Stability	±0.005%/month
Temperature coefficient	negligible

SELECTED RESOLUTION

	0.001%		0.01%		0.1%	
DC Voltage Range	Maximum Reading	Resolution	Maximum Reading	Resolution	Maximum Reading	Resolution
10mv	30.0000 mv	0.1μ v	030.000 mv	1 μ v	0030.00 mv	10 μ v
100mv	300.00 mv	1μ v	0300.00 mv	10μ v	00300.0 mv	100μ v
1000 mv	3000.00 mv	10μ v	03000.0 mv	100μ v	003000. mv	1 mv
10v	30.0000v	100μ v	030.000v	1mv	0030.00v	10 mv
100v	300.000v	1mv	0300.00v	10 mv	00300.0v	100 mv
1000v*	0300.00v	10 mv	00300.0v	100 mv	000300.v	1v

*1000v range is scanner-limited to 300v peak maximum

SCANNING SPEED

(Programmed Range)

Resolution	Integration Time	Total Time	Scanning Speed
0.1%	1.6	20 msec	50 ch/sec.
0.01%	16.6 msec	40 msec	25 ch/sec.
0.001%	166.6 msec	188 msec	5 ch/sec.

Scanning Speed (Auto Range)–Add 6-36 msec depending on per-channel voltage span.

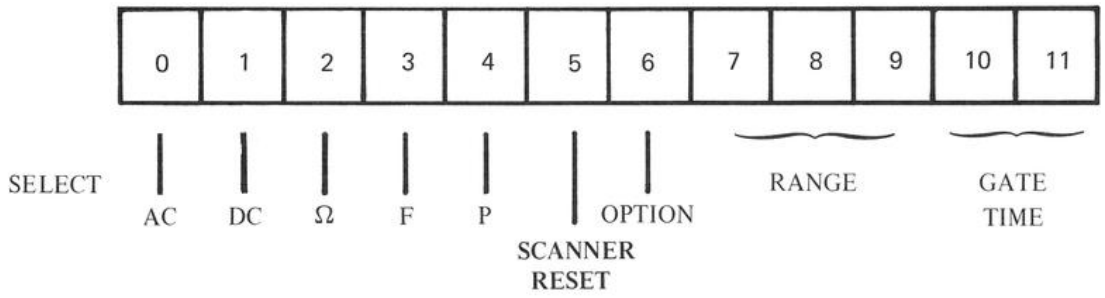
Instructions

The I/O transfer (IOT) commands associated with the scanning digital coltmeter system are designed to minimize the computer overhead associated with this option while retaining maximum program controlled flexibility. The IOT instructions are:

VSEL Select Range and Gate

Octal code: 6542
Event time: 2
Execution time: 4.25 μsec
Operation: The contents of the accumulator are transferred to the AD04A control register as shown below.
Symbol: C(AC) = > C(VCR)

CONTROL WORD 1 (FROM PDP-12)

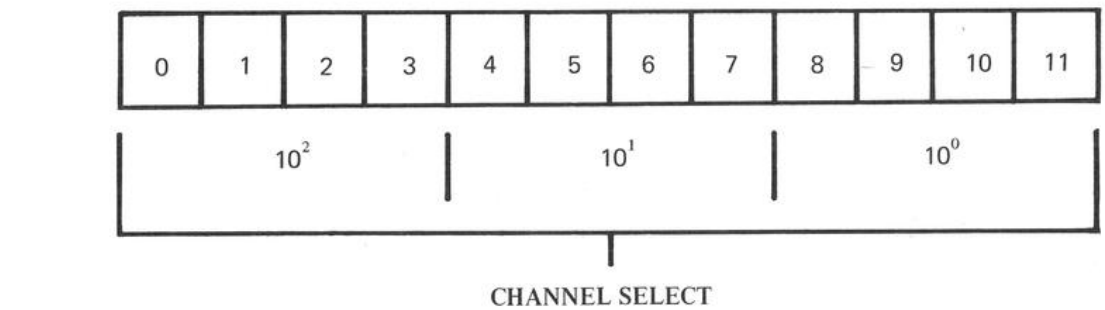


Control Word 1 only used if a range change is required

VCNV Select Channel and Convert

Octal code: 6541
Event time: 1
Execution time: 4.25 μsec
Operation: The contents of the accumulator are transferred to the AF04A channel address register as shown below. The analog signal on the selected channel is automatically digitized.
Symbol: C(AC) = > C(VAR)

CONTROL WORD 2 (FROM PDP-12)



VINX Index Channel and Convert

Octal code: 6544
Event time: 3
Execution time: 4.25 μsec
Operation: The last channel address is incremented by one and the analog signal on the selected channel is automatically digitized. The contents of the control register is unchanged.
Symbol: VAR + 1 = > VAR

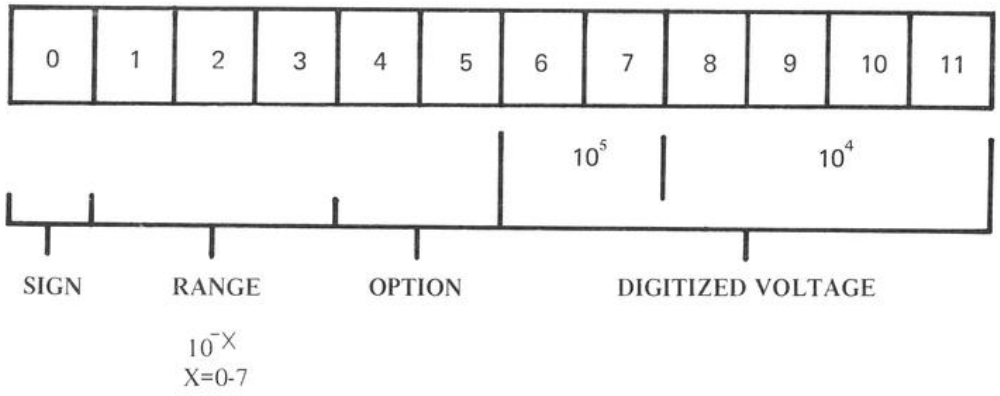
VSDR Skip on Data Ready

Octal code: 6531
Event time: 1
Execution time: 4.25 μsec
Operation: When the scanning voltmeter has selected a channel and digitized the analog signal, a data ready flag is set. This instruction is used to test for the data ready flag.
Symbol: If Flag = 1, the PC + 1 = > PC

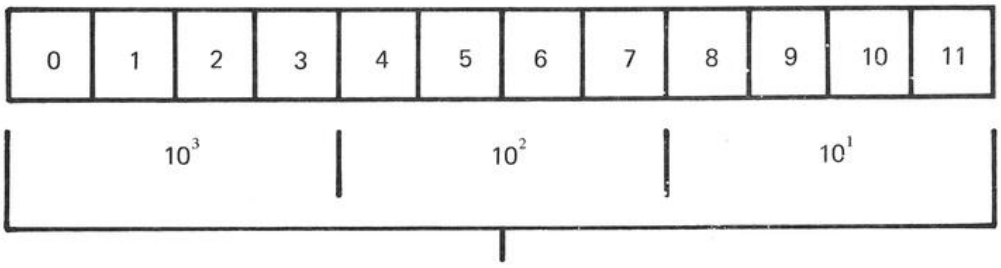
VRD Read Data and Clear Flag

Octal code: 6532
Event time: 2
Execution time: 4.25 μsec
Operation: The contents of the selected byte of the voltmeter output word is transferred to the accumulator and the data ready flag is cleared. The first data flag after the flag is set, is always byte 1 (see below). Subsequent bytes are program selected using the byte advance command.
Symbol: C(VOR) = > C(AC)

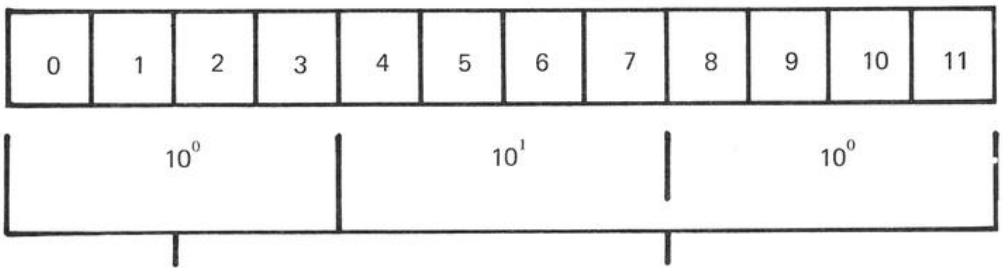
DATA WORD (TO PDP-12) BYTE 1



DATA WORD (TO PDP-12) BYTE 2



DATA WORD (TO PDP-12) BYTE 3



Data word 3 seldom required, all address and digitized data are in 8-4-2-1 BCD format.

VBA Byte Advance

Octal code: 6534
Event time: 3
Execution time: 4.25 μsec
Operation: The total data word from the AF04A is 36-bits long. The first data word after the flag is set, is always the twelve most significant bits. The BYTE ADVANCE command requests the next twelve most significant bits. When the data is available, the data ready flag is set again. To select the twelve least significant bits, a second BYTE ADVANCE command is required. When the data is available, the data ready flag is set again.
Symbol: $C(VOR_{0-12}) => C(VOR_{13-23})$
or $C(VOR_{13-23}) => C(VOR_{24-35})$

VSCC Sample Current Channel

Octal code: 6571
Event time: 1
Execution time: 4.25 μsec
Operation: The analog signal on the current channel is digitized. This command is not required except when multiple samples are required on any channel. (Using this command on a preselected channel saves up to 10 milliseconds per sample.)
Symbol: None

5.4.14 Frequency and Period Measurement Options for AF04A

A separate input permits the IDVM to be used as a frequency counter capable of counting to 2mHz with selectable gate times of 1, 10, and 100 milliseconds, providing measurement resolution of 10Hz. Increased accuracy at low frequencies (to 10kHz with automatic 250% overranging) is accomplished with the period-measurement mode. This mode counts an internal frequency source for 1, 10, or 100 periods of the frequency being measured, thereby providing increased full-scale accuracy. Period readout is in milliseconds.

Frequency and voltage measurements may be made within one scanning cycle by grouping all frequency inputs in one master or slave scanner and all voltage inputs in another master or slave scanner. The output of the scanner may then be connected to the frequency-input connector of the IDVM and the output of the other scanner to the voltage input. One of the optional control word bits is used to program the IDVM for frequency or period measurements.

SPECIFICATIONS

Frequency Measurements

Range:	10Hz to 2mHz
Sensitivity:	100mv rms or −1v pulses, at least 0.3 μsec wide at 50% points. 100 v rms maximum working voltage.
Input Impedance:	22k ohms shunted by less than 1000 pf, including internal cabling.
Accuracy:	±1 count + time base accuracy
Time Base:	100 KHz crystal oscillator with initial accuracy of ±0.0005%, long-term stability ±0.0001%/wk; temp. coefficient ±0.0002%/°C.

Period Measurements

Range:	1, 10, and 100 period average. Input frequency from 10Hz to 25kHz sine wave or 0.1 pps. to 25,000 pps.
Sensitivity:	100 mv rms or −1v pulses, at least 0.3 μsec wide at 50% points. 100v rms maximum working voltage.
Input Impedance:	22k ohms shunted by less than 1000pf, including internal cabling.
Accuracy:	±1 count + time base accuracy + trigger error. Trigger error $<$ ± 0.03% for 100mv rms sine wave with 40db signal-to-noise ratio.
Time Base:	100kHz crystal oscillator with initial accuracy of ±0.0005% long-term stability ±0.0001%/wk; temp. coefficient ±0.0002%/°C.

Selected Resolution	0.001%		0.01%		0.1%	
Function	Maximum Reading	Resolution	Maximum Reading	Resolution	Maximum Reading	Resolution
Frequency	2000.00kHz	10Hz	02000.0kHz	100Hz	002000kHz	1kHz
Period	99.9999msec	0.1μs	999.999msec	1.0μs	9999.99msec	10μs

Additional AF04A Options

Information on the following options may be had from your nearest DIGITAL EQUIPMENT CORPORATION Office:

Frequency (period) measurements.
AC/ohms/DC Converter
Time-of-day clock.
Thumb-wheel data entry panel.
Thermocouple reference junctions.
Extended scanner for more than 1000 channels.
Special cabinet with roll-out drawer chassis accessibility.

CHAPTER 6
PROGRAM LIBRARY

Because of the dual nature of the PDP-12 virtually all PDP-8, classic LINC and LINC-8 programs will run on the PDP-12 if equipped with the correct peripherals. The following programs are normally supplied with the PDP-12 and most will run on just the PDP-12A. There are other programs available from both Digital Equipment Corporation and from DECUS. These have not been included in the Standard Library for various reasons such as: duplicate function, special peripherals, incomplete documentation, or limited applicability. This chapter is divided into four sections; PDP-12 Programs, PDP-8 Programs, Decus Programs, and Diagnostic Programs.

6.1 PDP-12 PROGRAMS

These programs include those written especially for the PDP-12 as well as some programs previously available on the LINC-8.

DIAL Diaplay Interactive Assembly Language

DIAL is the basic operating system of the PDP-12. This sophisticated operating system assumes the use of two tape transports and a display as integral parts of the system. The operating system is composed of an editor, an assembler, a monitor, and data-handling routines for performing the typical functions required in program development.

The operating system is normally in the editor mode, awaiting input information that may be entered via the Teletype, magnetic tape units, or disks. The command structure of the operating system allows simple and straightforward commands to perform operations such as manipulation (addressing and deleting) of manuscripts, filing of information, copying of manuscripts and data, editing, listing, searching, and assembling.

The editor allows editing on a line-by-line or on a character basis by the use of a controllable cursor on the display. The editor allows random access to any character or line being edited in the text, without resorting to any scheme using paging techniques. The assembler is a two-pass program that is called directly via the monitor. The assembler utilizes additional core and/or disk storage for handling symbols that it cannot contain within the initial 4K of memory. The concept is to maximize the speed at which the assembly can be made. The assembler will handle both PDP-8 and LINC instructions. Six character symbols and tags are used throughout.

Q & A Subroutine

Q & A - allows the user to compose Question and Answer dialogues between the operator via the key- board and the computer via the display.

DATA-12

This program retrieves, displays, and stores individual data blocks from magnetic tape and provides the user with a repertoire of mathematical operations for manipulating this data. These operations include high and low pass filtration, differentiation and integration, attenuation and amplification, inversion, addition of a constant, and plotting of a bar graph. The data or resulting waveforms are continuously displayed.

GRAPH

This program allows data to be retrieved from magnetic tape and displayed, as well as a graph to be composed for this data with appropriate lettering and axes. The graph is assembled on the display and the finished product may be photographed, plotted on an incremental plotter, or saved on magnetic tape for future reference.

FRQN 12

This program performs a frequency analysis of 512 points of data, and resolves the resulting spectrum into 64 components. The sine, cosine, and rms spectra are subsequently displayed and can be scaled. A resynthesis from the spectra can then be performed to provide a comparative display of the original data and the resynthesized waveform.

A-D TO TAPE

This program uses the KW12 Real Time Clock and TC12 Buffered Tape control to collect and save continuous records of analog data.

MAGSPY

This program provides a moving window for scanning data stored on digital magnetic tape. The data is displayed on the scope and can be scanned at a rate determined by a potentiometer setting. The data can be interpreted either as binary numbers or packed characters.

COMPAR

This program compares contents of specified tape blocks.

SEARCH

This program performs a search of blocks of tape for a specific word. The user may specify the word to be searched and a mask of bits for comparison.

CONVERT

This program is used to convert symbols from LAP-4, LAP-6 and PAL III manuscripts for compatibility with the DIAL Programming system and Assembler.

MARK12

This program is used to format tapes to be used by the PDP-12. Several format options are available and using the subroutines within MARK12 the user can generate a tape of arbitrary format.

TRAP PROCESSOR

This is a collection of routines to be used in response to Instruction Traps. These Trap Processor programs are useful for simulating LINC-8 operations no longer applicable on the PDP-12. The Trap feature is particularly useful in developing "device independent software".

6.2 PDP-8 PROGRAMS

The PDP-12 is delivered to the user complete with an extensive selection of system programs and routines making the full data processing capability of the new computer immediately available to each user, eliminating many commonly experienced initial programming delays.

The programs described in these abstracts come from two sources, past programming efforts on the PDP-5, 8, 8/S, 8/I, PDP-8/L, and present and continuing programming effort on these same machines plus the PDP-12. Thus the programming system takes advantage of the many man-years of program development and field testing by Digital computer users. There are over 3500 Family-of-8 systems in the field already.

Although in many cases PDP-12 programs originated from previous Family-of-8 computers, all utility and functional program documentation is issued anew, recursive format introduced with the Family-of-8 computers. Programs, written by users of Family-of-8 computers and submitted to the DECUS library (DECUS - Digital Equipment Corporation User's Society) are immediately available to Family-of-8 users. Consequently, users of all Family-of-8 computers can take advantage of continuing program developments.

6.2.1 System Programs

DEC-08-AJAB-D FOCAL

FOCAL (for *FO*rmula *CAL*culator) is an on-line, conversational, service program for the PDP-8-family of computers, designed to help scientists, engineers, and students solve numerical problems. The language consists of short imperative English statements which are relatively easy to learn. Mathematical expressions are typed, for the most part, in standard notation. No previous programming experience is needed either to understand this manual or to use FOCAL at the Teletype console. However, the best way to learn the FOCAL language is to sit at the Teletype and try the commands, starting with the examples given in the manual.

Disk/LINC tape Software

DEC-D8-SDAA-D Disk Monitor System (PDP-8 Mode)

This system consists of a keyboard-oriented Monitor, which enables the user to efficiently control the flow of programs through his PDP-12, and a comprehensive software package, which includes a FORTRAN Compiler, Program Assembly Language (PAL-D), Edit program (EDITOR), Peripheral Interchange Program (PIP) and Dynamic Debugging Technique (DDT-D) program. Also provided is a program (BUILDER) for generating a customized monitor according to the user's particular machine configuration (amount of core, number of discs or LINC tapes, etc.).

The system is modular and open ended, permitting the user to construct the software required in his environment, and allows the user full access to his disc (or LINC tape) - referred to as the system device - for storage and retrieval of his programs. By typing appropriate commands to the Monitor, the user can load a program (construct it from one or more units of binary coding previously punched out on paper tape or written on the disc by the Assembler, and assign it core), save it (write it out, with an assigned starting address, on the system device), and later call it (read it back into core from the system device) for execution.

In order to have a complete DISC/LINC tape package, the user may order the following in addition to DEC-D8-SDAA-D above:

1. Disc System Builder
2. Disc Editor
3. PIP
4. Disc DDT
5. Disc DDT Driver (ASCII)
6. Disc/LINC tape FORTRAN
7. PAL-D Assembler

DEC-D8-ASAA-D PAL-D Disk Assembler

PAL-D is the symbolic assembly program designed primarily for the 4K PDP-8 family of computers with disc or LINC tape. The PAL-D Assembler makes machine language programming easier, faster, and more efficient. Basically, the Assembler processes the programmer's source program statements by translating mnemonic operation codes needed in machine instructions, relating symbols to numeric values, assigning absolute core addresses for program instructions and data, and preparing an output listing of the program, which includes notification of any errors detected during the assembly process (for PDP-8 mode only).

DEC-08-ESAA-D Symbolic Editor

The Symbolic Editor program is used to generate, edit, correct, and update symbolic program tapes using the tape teleprinter. With the Editor in memory, the user reads in portions of his symbolic tape, removes, changes, or adds instructions and operands, and gets back a new, complete, symbolic tape with errors removed. He can work through the program instruction by instruction, spot check it, or concentrate on new selections. The tape can contain either symbolic machine language, FORTRAN source statement, data, or text information. This program is available for use with either the 33ASR reader/punch or the high speed reader/punch.

DEC-08-AFAB-D FORTRAN II Manual

One-pass FORTRAN compiler and operating system compiles FORTRAN source language statements into an object program tape. The operating system executes the program. This system contains the interpreter, arithmetic function subroutines, and input-output packages.

DEC-08-ASAA-D PAL III (Program Assembler Language)

Symbolic machine language assembler. Converts programs code in symbolic machine language to binary machine language. The basic process performed by the Assembler is the substitution of numeric values for symbols, according to associations defined in the symbol table. In addition, the user may request that the Assembler itself assign values to the user's own symbols at assembly time. These symbols are normally used to name memory locations, which may then be referenced by name. An assembly listing may be produced.

Digital-8-4-S DDT

Dynamic Debugging Tape provides a means for on-line program debugging at the symbolic or mnemonic level. By typing commands on the console teleprinter, memory locations can be examined and changed, program tapes can be inserted, selected portions of the program can be run, and the updated program can be punched.

Digital-8-5-S Floating-Point System

A Basic System
B Interpreter, I/O, I/O Controller
C Interpreter, I/O Functions
D Interpreter, I/O, I/O Controller, Functions

Includes Floating-Point Interpreter and I/O subsystems. Allows the programmer to code his problem in floating-point machine language.

Floating-point operations automatically align the binary points of operands, retaining the maximum precision available by discarding leading zeros. In addition to increasing accuracy, floating-point operations relieve the programmer of the scaling problems common in fixed-point operations. This system includes elementary function subroutines programmed in floating-point. These subroutines are sine, cosine, square root, logarithm, arctan, and exponential functions. Data being processed in floating-point is maintained in three words of memory (12-bit exponent, 24-bit mantissa). An accuracy of seven decimal places is maintained.

DEC-08-AFAZ-PB FORTRAN Symbol Print

Loaded over the FORTRAN Compiler, this program lists the variables used and where they will be located in core. It also indicates the section of core not used by the compiled program and data.

Digital-8-10-S CALCULATOR

CALCULATOR is an equation evaluation routine. It differs from FORTRAN in that the function to be evaluated is entered via keyboard and calculated immediately upon termination of entry. Format control is provided so that computer results may be conveniently tabulated. Expressions causing the calling of common function subroutines are included.

DEC-08-COAA-D(L) ODT-11

ODT-11 (Octal Debugging Tape) acts in debugging a program by facilitating communication with the program being run via the ASR 33 Teletypewriter. ODT-11 features include register examinations and modification, control transfer, word searching, octal dumping, and instruction traps.

6.2.2 Elementary Function Routines

The following routines are described in the Program Library Math Routines Manual (DEC-08-FFAA-D).

Square Root Subroutine-Single Precision

Forms the square root of a single-precision number. An attempt to take the square root of a negative number will give 0 for a result.

Signed Multiply Subroutine-Single Precision

Forms a 22-bit signed product from 11-bit signed multiplier and multiplicand.

Signed Divide Subroutine-Single Precision

This routine divides a signed 11-bit divisor into a signed 23-bit dividend giving a signed 11-bit quotient and a remainder of 11 bits with the sign of the dividend.

Double-Precision Multiply Subroutine-Signed

This subroutine multiplies a 23-bit signed multiplicand by a 23-bit signed multiplier and returns with a 46-bit signed product.

Double-Precision Divide Subroutine-Signed

This routine divides a 23-bit signed divisor into a 47-bit signed dividend and returns with a 23-bit signed quotient and a remainder of 23 bits with the sign of the dividend.

Sine Routine-Double Precision

The Double-Precision sine subroutine evaluates the function Sin (X) for $-4<4$ (X is in radians). The argument is a double-precision word, two bits representing the integer part and 21 bits representing the fractional part. The result is a 23-bit signed fraction $-1<\text{Sin}(X)<1$.

Cosine Routine-Double Precision

This subroutine forms the cosine of a double-precision argument (in radians). The input range is $-4<X<4$.

Four-Word Floating-Point Package

This is a basic floating-point package that carries data as three words of mantissa and one word of exponent. Common arithmetic operations are included as well as basic input/output control. No functions are included.

Logical Subroutines

Subroutines for performing the logical operations of inclusive and exclusive OR are presented as a package.

Shift Right, Shift Left Subroutines (Single and Double Precision)

Four basic subroutines, shift right and shift left, each at both single and double precision, are presented as a package.

Logical Shift Routines

Two basic subroutines, shift right at both single and double precision, are presented as a package. The shifts are logical in nature.

Digital-8-21-FSigned Multiply (Uses EAE) Single Precision

This subroutine forms a 22-bit signed product from an 11-bit signed multiplier and multiplicand using the Extended Arithmetic Element. It occupies less storage and takes less time to execute than its non-EAE counterpart, and it has the same calling sequence.

Digital-8-22-F Signed Divide (Uses EAE) Single Precision

This subroutine divides a double-precision signed 22-bit dividend by a signed 11-bit divisor, producing a signed 11-bit quotient and a remainder of 11 bits having the sign of the dividend.

It makes use of the Extended Arithmetic Element instruction set and occupies less storage and takes less time to execute than its non-EAE counterpart. It has the same calling sequence except that the subroutine name is changed from DIVIDE to SPDIV.

Digital-8-23-F Signed Multiply (Uses EAE) Double Precision

This subroutine multiplies a 23-bit, signed 2's complement binary number by a 23-bit signed 2's complement binary number, giving a 46-bit product with two signs on the higher order end. It makes use of the Extended Arithmetic Element instruction set and, because of this, occupies less storage and takes less time to execute than its non-EAE counterpart. Its calling sequence is comparable with the non-EAE version.

Digital-8-25-F EAE Floating-Point Package

These packages perform the same tasks as the Floating-Point Packages (Digital-8-5-S A, B, C, D) except that certain routines have been speeded up by the use of the Extended Arithmetic Element.

For a detailed description of a floating-point arithmetic and the interpretive Floating-Point Packages, the reader is referred to Digital-8-5-S.

6.2.3 Utility Programs

Digital-8-0 Format for Program Documentation

With the advent of the PDP-8, Digital Equipment Corporation introduced a new, recursive format for program documentation. This format is used for routines and subroutines, such as utility and functional, but not necessarily for system programs.

This format and its use are described in this document.

Digital-8-1-U Read-In-Mode Loader

The RIM Loader is a minimum-sized routine for reading and storing the information in Read-In-Mode coded tapes via the ASR 33 Perforated Tape Reader.

Digital-8-2-U Binary Loader (33ASR, PR12, MC12 Memory Extension)

The Binary Loader is a short routine for reading and storing the information in binary-coded tapes via the ASR 33 Perforated Tape Reader or by means of the Type PR12 High-Speed Perforated Tape Reader.

DEC-08-PMPA-D RIM Punch

This program provides a means of punching out the information as selected blocks of core memory as RIM-coded tape via the ASR 33 Perforated Tape Reader.

Digital-8-5-U-SYM Binary Punch 33/PP-8/I

This program provides a means of punching out the information in selected blocks of core memory as binary-coded tape via the ASR 33 Perforated Tape Punch or via the High-Speed Punch PP-8/I.

Digital-8-6-U-SYM Octal Memory Dump

This routine reads the console switches to obtain the upper and lower limits of an area of memory, then types on the Teletype an absolute address plus the octal contents of the first four words specified and repeats this until the block is exhausted, at which time the user may repeat the operation.

Digital-8-10-U Binary-Coded-Decimal to Binary Conversion Subroutine

This basic subroutine converts unsigned binary-coded-decimal numbers to their equivalent binary values.

Digital-8-11-U Double Precision BCD-to-Binary by Radix Deflation

This subroutine converts a 6-digit BCD number to its equivalent binary value contained in two computer words.

Digital-8-12-U Incremental Plotter Subroutine

This subroutine moves the pen of an incremental plotter to a new position along the best straight line. The pen may be raised or lowered during the motion.

Digital-8-14-U Binary to Binary-Coded-Decimal Conversion

This subroutine provides the basic means of converting binary data to binary-coded-decimal (BCD) data for typeout, magnetic tape recording, etc.

Digital-8-15-U-SYM Binary-to-Binary-Coded-Decimal Conversion (Four Digit)

This subroutine extends the method used in Digital-8-14-U so that binary integers from 0 to 4095 in a single computer word may be converted to four binary-coded-decimal characters packed in two computer words.

Digital-8-17-U EAE Instruction Set Simulator

This routine permits the automatic multiply-divide hardware option to be simulated on a basic PDP-8/I.

Digital-8-18-U-SYM Alphanumeric Message Typeout

This is a basic subroutine to type messages packed in computer words. Two 6-bit characters are packed internally in a single word. All ASR 33 codes from 301 to 337 and from 240 to 277 (excepting 243 and 245) can be typed. The typing of line feed (code 212) and carriage return (code 215) are made possible by arbitrarily assigning internal codes of 43 and 45, respectively, to represent these characters, thus preventing the output of ASCII codes 243 (#) and 245 (%).

Digital-8-19-U-SYM Teletype Output Subroutine

A group of subroutines useful in controlling ASR 33 output is presented as a package. Provision is made for simulation of tabulation stops. The distance *tabbed* may be controlled by the user. Characters whose ASR 33 codes are in the groups 241 through 277, inclusive, and 300 through 337, inclusive, are legal. Space, carriage return then line feed, and tabulation are provided via subroutines.

Digital-8-20-U-SYM Character String Typeout Subroutine

This basic subroutine types messages stored internally as a *string* of coded characters. All ASR 33 characters are legal.

Digital-8-21-U-SYM Symbolic Tape Format Generator

The Format generator allows the user to create PDP-8 symbolic tapes with Formatting. It may be used to condense tapes with spaces by inserting tabs, or merely to align tabs, instructions, and comments.

Digital-8-22-U-SYM Unsigned Decimal Print

This subroutine permits the typeout of the contents of a computer word as a four-digit, positive, decimal integer.

Digital-8-23-U-SYM Signed Decimal Print, Single Precision

This subroutine permits the typeout of the contents of a computer word as a signed 2's complement number. If bit 0 of the computer word is a 1, the remaining bits represent a negative integer in 2's complement form; if bit 0 equals 0, the remaining bits represent a positive integer. If the number is negative, a minus sign is printed; if positive, a space.

Digital-8-24-U-SYM Unsigned Decimal Print, Double Precision

This subroutine permits the typeout of a double-precision integer stored in the usual convention for double-precision numbers, (see DEC-08-FFAA-D). The one exception is that all 24 bits are interpreted as magnitude bits (i. e., the bit 0 of the high-order word is not a sign bit). The typeout is in the form of a seven-digit, positive, decimal integer.

Digital-8-25-U-SYM Signed Decimal Print, Double Precision

This subroutine permits the typeout of the contents of two consecutive computer words as one signed double-precision, 2's complement number. If bit 0 of the high-order word is a 1, the remaining 23 bits represent a negative integer in 2's complement form; if bit 0 equals 0, the remaining bits represent a positive integer. If the number is negative, a minus sign is printed; if positive, space.

Digital-8-28-U-SYM Single Precision DBC & Input (ASR-33) Signed or Unsigned

This routine accepts a string of up to four decimal digits (single precision for the PDP-12) from the Teletype keyboard and converts it to the corresponding 2's complement binary number.

The string may contain as legal characters a sign (+, –, or space) and the digits from 0-9. If the first legal character is not a sign, the conversion is unsigned. A back arrow (←) at any point in the string erases the current string and allows the operator to reenter the correct value. Any character after the first, other than another digit or back arrow causes the conversion to terminate and is found in location SISAVE within the subroutine.

Digital-8-29-U-SYM Double Precision DBC & Input (ASR-33) Signed or Unsigned

This routine accepts a string of up to eight decimal digits (double-precision for the PDP-12) from the Teletype keyboard and converts it to the corresponding 2's complement binary number.

The string may contain as legal characters a sign (+, –, or space) and the digits 0-9. If the first legal character is not a sign, the conversion is unsigned. A back-arrow (←) at any point in the string erases the current string and allows the operator to reenter the value. Any character after the first, other than another digit or back-arrow, causes the conversion to terminate and is found in location DIDSAV within the subroutine.

6.3 DECUS PROGRAMS

DECUS is the Digital Equipment Corporation User's Society which collects, files and distributes user written programs for all DEC computers. Below is a partial list of PDP-8 programs available through DECUS. More detailed lists are available in the periodic DECUS program catalog.

DECUS NO. 5/8-1.1 BPAK – A BINARY INPUT-OUTPUT PACKAGE

A revision of the binary package originally written by A. D. Hause of Bell Telephone Laboratories. With BPAK the user can read in binary tapes via the photoreader and punch them out via the Teletype punch. It may be used with any in-out device, but is presently written for the photoreader and Teletype punch. A simple modification converts BPAK so that it reads from the Teletype reader if the photoreader is disabled. In its present form it occupies locations 7600-7777.

DECUS NO. 5.2.1 OPAK – AN ON-LINE DEBUGGING PROGRAM FOR THE PDP-5

A utility program which enables the user to load, examine, and modify computer programs by means of the Teletype. This program is a revision of the program written by A. D. Hause of Bell Telephone Laboratories. Extensive use of the program has suggested many refinements and revisions of the original program, the most significant additions being the word search and the break point. The standard version of OPAK is stored in 6200 to 7577 and also 0006. An abbreviated version is available (7000 to 7577, 0006) which is identical to the other except that it has no provision for symbolic dump. Both programs are easily relocated. Control is via Teletype with mnemonic codes, (e.g., B for inserting breakpoint, P for proceed, etc).

DECUS NO. 5-3 BRL – A BINARY RELOCATABLE LOADER WITH TRANSFER VECTOR OPTIONS FOR THE PDP-5 COMPUTER

A binary loader program occupying 4640_8 to 6177_8 registers, also 160 to 177. It has two main functions:

1. It allows a PDP-5 operator to read a suitable prepared binary program into any page location in memory except the registers occupied by BRL.

2. It greatly simplifies the calling of programmed subroutines by allowing the programmer to use an arbitrary subroutine calling sequence when writing his program, instead of having to remember the location of the subroutines.

DECUS NO. 5-4, OCTAL TYPEOUT OF MEMORY AREA WITH FORMAT OPTION

(Write-up and Listing Only)

DECUS NO. 5-5, EXPANDED ADDING MACHINE

Expanded Adding Machine is a minimum-space version of Expensive Adding Machine (DEC-5-43-D) using a table lookup method including an error space facility.

This is a basic version to which additional control functions can easily be added. Optional vertical or horizontal format, optional storage of intermediate result without reentry, fixed-point output of results within reason, and other features that can be had in little additional space under switch register control. (Write-up and Listing Only)

DECUS NO. 5-6, BCD TO BINARY CONVERSION OF 3-DIGIT NUMBERS

This program is based on DEC-5-4 and is intended to illustrate the use of alternative models in program construction.

While not the fastest possible, this program has one or two interesting features. It converts any 3-digit BCD-coded decimal number, $D_1 D_2 D_3$ into binary in the invariant time of 372 microseconds. Efficient use is made of BCD positional logic to work the conversion formula $[10(10D_1 + D_2) 10 + D_3]$ by right shifts in the accumulator. In special situations, it could be profitable to insert and initial test/exit on zero, adding 12 microseconds to the time for non-zero numbers.

(Write-up and Listing Only)

DECUS NO. 5/8-7, DECIMAL TO BINARY CONVERSION BY RADIX DEFLATION ON PDP-8

(Write-up and Listing Only)

DECUS NO. 5-8, PDP-5 FLOATING POINT ROUTINES

Consists of the following routine:

1. Square Root – Binary Tape and Symbolic Listing
2. Sine-Cosine – Binary Tape only
3. Exponential – Binary Tape only

DECUS NO. 5/8-9, ANALYSIS OF VARIANCE PDP-5/8

An analysis of variance program for the standard PDP-5/8 configuration.

The output consists of:

A. For each sample:

   1. sample number
   2. sample size
   3. sample mean
   4. sample variance
   5. sample standard deviation

B. The grand mean

C. Analysis of Variance Table:

1. The grand mean.
2. The weighted sum of squares of class means about the grand mean.
3. The degrees of freedom between samples.
4. The variance between samples.
5. The pooled sum of squares of individual values about the means of their respective classes.
6. The degrees of freedom within samples.
7. The variance within samples.
8. The total sum of squares of deviations from the grand mean.
9. The degrees of freedom.
10. The total variance.
11. The ratio of the variance between samples to the variance with samples.

This is the standard analysis of variance table that can be used with the F test to determine the significance, if any, of the differences between sample means. The output is also useful as a first description of the data.

All arithmetic calculations are carried out by the Floating Point Interpretive Package (Digital-8-5-S).

DECUS NO. 5-10, PAPER TAPE READER TEST

A test tape can be produced and will be continuously read as an endless tape. Five kinds of errors will be detected and printed out. The Read routine is in 6033-6040. Specifications: Binary with Parity Format – Length: registers in locations (octal): 10, 11, 4 through 67 (save 63, 64), and 6000-7777.

DECUS NO. 5-11, PDP-5 DEBUG SYSTEM

Purpose of this program is to provide a system capable of:

1. Octal dump 1 word per line.
2. Octal dump 10_8 words per line.
3. Modifying memory using the typewriter keyboard.
4. Clearing to zero parts of memory.
5. Setting to HALT codes part of memory.
6. Entering breakpoints into a program.
7. Initiating jumps to any part of memory.
8. Punching leader on tape.
9. Punching memory on tape in RIM format.
10. Punching memory on tape in PARITY format.
11. Load memory from tape in PARITY format.

DECUS NO' 5-12, PACK-PUNCH PROCESSOR AND READER FOR THE PDP-5

The processor converts a standard binary-format tape into a more compressed format, with two twelve-bit words contained on every three lines of tape. Check sums are punched at frequent intervals, with each origin setting or at least every 200 words.

The reader, which occupies locations 7421 to 7577 in the memory will load a program which is punched in the compressed format. A test for checksum error is made for each group of 200 or less and the program will halt on error detection. Only the most recent group of words need be reloaded. Read-in time is about ten per cent less than for conventional binary format, but the principle advantage is that little time is lost when a checksum error is detected, no matter how long the tape.

DECUS NO. 5/8-14, DICE GAME FOR THE PDP-5/8

ENABLES A USER TO PLAY THE GAME, DICE, on either the PDP-5 or PDP-8.

DECUS NO. 5-15, ATEPO (AUTO TEST IN ELEMENTARY PROGRAMMING AND OPERATION OF A PDP-5 COMPTUER)

The program will type questions or instructions to be performed by the operator of a 4K PDP-5. The program will check to see if the operator has answered the questions correctly. If this is the case, it will type the next question or instruction.

DECUS NO. 5-16, PAPER TAPE DUPLICATOR FOR PDP-5

The tape duplicator for the PDP-5 is a single buffered read and punch program utilizing the program interrupt. It computes a character count and checksum for each tape and compares with checks at the end of the tape. Checks are also computed and compared during punching.

DECUS NO. 5/8-17, TYPE 250 DRUM TRANSFER ROUTINE FOR USE ON PDP-5/8

Transfers data from drum to core (Read) or core to drum (Write) via ASR-33 Keyboard Control.

DECUS NO. 5/8-18a, BINARY TAPE DISASSEMBLY PROGRAM

Disassembles a PDP-5 or 8 program, which is on tape in BIN format. It prints the margin setting, address, octal contents, mnemonic interpretation (PAL) of the octal contents. A normal program or a program which uses Floating Point may be disassembled.

DECUS NO. 8-19a, DDT-UP OCTAL-SYMBOLIC DEBUGGING PROGRAM

DDT-UP is an octal-symbolic debugging program for the PDP-8 which occupies locations 5600 through 7677. It is able to read a symbol table punched by PDPSYM and stores symbols, four locations per symbol, from 5577 down towards 0000. The mnemonics for the eight basic instructions and standard OPR and IOT group instructions are initially defined and the highest available location for the user is initially 5363.

From the Teletype, the user can symbolically examine and modify the contents of any memory location. DDT-UP allows the user to punch a corrected program in CBL format.

DDT-UP has a breakpoint facility to help the user run sections of his program. When this facility is used, the debugger also uses location 0005.

DECUS NO. 5/8-20, REMOTE OPERATOR FORTRAN SYSTEM

Program modification and instructions to make the FORTRAN OTS version dated 2/12/65 operate from remote stations.

DECUS NO. 5/8-21, TRIPLE PRECISION ARITHMETIC PACKAGE FOR THE PDP-5 AND THE PDP-8

An arithmetic package to operate on 36-bit signed integers. The operations are add, subtract, multiply, divide, input conversion, and output conversion. The largest integer which may be represented is $2^{35} - 1$ or 10 decimal digits. The routines simulate a 36-bit (3 word) accumulator in core locations 40, 41, and 42 and a 36-bit multiplier quotient register in core locations 43, 44, and 45. Aside from the few locations in page 0, the routines use less core storage space than the equivalent double-precision routines.

DECUS NO. 5/8-23, PDP-5/8 OSCILLOSCOPE SYMBOL GENERATOR

The subroutine may be called to write a string of characters, a pair of characters, or a single character on an oscilloscope. Seventy (octal) symbols in ASCII Trimmed Code and four special *format* commands are acceptable to this routine. The program is operated in a fashion similar to the DEC Teletype Output Package.

DECUS NO. 5-24, VECTOR INPUT/EDIT

Accepts input to a PDP-5 and allows both time-of-entry and post time-of-entry corrections.

DECUS NO. 5-25, A PSEUDO RANDOM NUMBER GENERATOR

The random number generator subroutine, when called repeatedly, will return a sequence of 12-bit numbers which, though deterministic, appears to be drawn from a random sequence uniform over the interval 0000_8 to 7777_8. Successive numbers will be found to be statistically uncorrelated. The sequence will not repeat itself until it has been called over 4 billion times.

DECUS NO. 8-26a, COMPRESSED BINARY LOADER (CBL)

The CBL (Compressed Binary Loader) format in contrast to BIN format utilizes all eight information channels of the tape, thus achieving nearly 25% in time savings.

Whereas BIN tapes include only one checksum at the end of the tape, CBL tapes are divided into many independent blocks, each of which includes its own checksum. Each block has an initial loading address for the block and a word count of the number of words to be loaded.

The CBL loader occupies locations 7700 through 7777.

DECUS NO. 8-26b, CBC (BIN TO CBL) AND CONV (CBL TO BIN)

Two conversion programs which use the PDP-8 on-line Teletype to read a binary tape in one format and punch a binary tape ii the other format. The conversion programs both ignore memory field characters so that the output is a tape for memory field 0.

DECUS NO. 8-26c, XCBL – EXTENDED MEMORY CBL LOADER

XCBL is used to load binary tapes punched in CBL format into a PDP-8 with more than standard 4K memory. This loader occupies locations 7670 through 7777 of any memory field.

DECUS NO. 8-26d, XCBL PUNCH PROGRAM

This program permits a user to prepare an XCBL tape of portions of a PDP-8 extended memory through the control of the keyboard of the on-line Teletype.

The program is loaded by the SCBL Loader.

There are two versions of the program so that any section of memory may be punched:

LOW XCBL occupies 00000-00377 and its starting address is 00000.

HIGH XCBL occupies 17200-17577 and its starting address is 17200.

The program may be restarted at the starting address at any time.

One option is provided according to the setting of bit 0 of the Switch Register. If bit 0 is a ONE, the operation of XCBL PUNCH is similar to that of DDT-UP (DECUS no. 8-19a).

DECUS NO. 5/8-27 AND 5/8-27a, BOOTSTRAP LOADER AND ABSOLUTE MEMORY CLEAR

Bootstrap Loader inserts a bootstrap loading program in page 0 from a minimum of toggled instructions.

Absolute Memory Clear leaves the machine in an absolutely clear state and, therefore, cycling around memory obeying an AND instruction with location zero. Should not be used unless one plans to reinsert the loader program.

DECUS NO. 5/8-28a, PAL III MODIFICATIONS-PHOENIX ASSEMBLER

This modification of the PAL III Assembler speeds up assembly on the ASR-33/35 and operates only with this I/O device. Operation is essentially the same as PAL III, except that an additional pass has been added, Pass 0. This pass, started in the usual manner but with the switches set to zero, reads the symbolic tape into a core buffer area. Subsequent passes then read the tape image from storage instead of from the Teletype.

DECUS NO. 5/8-29, BCD TO BINARY CONVERSION SUBROUTINES

These two subroutines improve upon the DEC supplied conversion routine. Comparison cannot be made to the DECUS-supplied fixed-time conversions. DECUS No. 5-6, because it is specified only for the PDP-5. One routine is designed for minimal storage, the other for minimal time. Both are fixed-time conversions; time specified is for a 1.5-μ sec machine.

Minimal time routine: 73.6 μ sec/32 locations

Minimal storage routine: 85 μ sec/29 locations

DEC Routine: 64-237 μsec/37 locations

DECUS NO. 5-30, GENPLOT–GENERAL PLOTTING SUBROUTINE

This self-contained subroutine is for the PDP-5 with a 4K memory and a CALCOMP incremental plotter. The subroutine can move (with the pen in the up position) to location (x,y), make an *x* at this location, draw a line from this present position to location (x,y) and initialize the program location counters.

DECUS NO. 5-31, FORLOT–FORTRAN PLOTTING PROGRAM FOR PDP-5

FORPLOT is a general-purpose plotting program for the PDP-5 computer in conjunction with the CALCOMP 560 Plotter. It is self-contained and occupies memory locations 0000_8 to 4177_8. FORPLOT accepts decimal data inputted on paper tape in either fixed or floating point formats. Formats can be mixed at will. PDP-5 FORTRAN output tapes are acceptable directly and any comment on these are filtered out.

DECUS NO. 5/8-32a, PROGRAM TO RELOCATE AND PACK PROGRAM IN BINARY FORMAT

Provides a means to shuffle machine language programs around in memory to make the most efficient use of computer store.

DECUS NO. 5/8-33, TAPE TO MEMORY COMPARATOR

Tape to Memory Comparator is a debugging program which allows comparison of the computer memory with a binary tape. It is particularly useful for detecting reader problems, or during stages of debugging a new program. Presently, uses high-speed reader, but may be modified for TTY reader.

DECUS NO. 5-34, MEMORY HALT–A PDP-5 PROGRAM TO STORE HALT IN MOST OF MEMORY

With Memory Halt and Opak, (DECUS No. 5-2.1), in memory, it is possible to store halt (7420) in the following memory locations:

0001 to 0005
0007 to 6177
7402 to 7403

DECUS NO. 5/8-35, BCD TO BINARY CONVERSION SUBROUTINE AND BINARY TO BCD SUBROUTINE (DOUBLE PRECISION)

This program consists of a pair of relatively simple and straightforward double-precision conversions.

DECUS NO. 5-36, OCTAL MEMORY DUMP REVISED

The Octal Memory Dump on Teletype is a DEC routine (DEC-5-8-U) which dumps memory by reading the switch register twice; once for a lower limit and again for an upper limit. It then types an address, the contents of the program and the next three locations, issues a CR/LF, then repeats the process for the next four locations. This leaves the right two-thirds of the Teletype page unused. The 78_{10} instructions occupy two pages.

This revised routine uses the complete width of the Teletype page and occupies only one memory page, using less paper and two less instructions. Now an address and the contents of 15 locations are typed out before a carriage return.

Octal Memory Dump Revised has proved its value as a subroutine and/or a self-contained dump program when it is necessary to dump large sections of DECtape, magnetic tape (IBM compatible) or a binary formatted paper tape.

DECUS NO. 5-37, TRANSFER II

For users who have more than one memory bank attached to the PDP-5/8, Transfer II may prove valuable in moving information from one field to another. When debugging, Transfer II enables a programmer to make a few changes in a new program and test it without reading in the original program again. Transfer II enables more extensive use of memory banks.

DECUS NO. 5/8-38, FTYPE–FRACTIONAL SIGNED DECIMAL TYPE-IN

Enables a user to type fractions of the form: .582. –.73, etc., which will be interpreted as sign plus 11 bits (e.g., $0.5 \mu 2000_8$). Subroutine reads into 300-3177 and is easily relocated, as it will work on any page without modifications.

DECUS NO. 5/8-39, DSDPRINT, DDTYPE–DOUBLE-PRECISION SIGNED DECIMAL INPUT-OUTPUT PACKAGE

DSDPRINT, when given a signed 24-bit integer, types a space or minus sign, and then a 7-digit decimal number in the range –8388608 to +8388607. DDYTPE enables user to type in a signed decimal number in either single or double precision. These routines are already separately available, but the present subroutine package occupies only one memory page and allows for more efficient memory allocation. Located in 3000-3177, but will work on any page.

DECUS NO. 5-40, ICS DECTAPE ROUTINES (ONE-PAGE)

The routines will read or write from the specified DECtape unit and delay the program until all I/O is completed. The last block read will overflow the specified region and destroy one core location. Only standard 129 word DECtape blocks will be read or written. The routines will halt if an error occurs with the status bits in the AC.

DECUS NO. 5-41, BREAKPOINT

This debugging routine has been reduced to a minimum operation. It is a mobile routine which can operate around any program that leaves an extra 30 cells of memory space.

Its function is to insert break points in any given location of the program being debugged, and to hold the contents of AC and Link. The programmer may examine any locations desired and then continue to the next breakpoint. It is presently located in 140_8-170_8 but may be easily relocated.

DECUS NO. 5-42 ALPHANUMERIC INPUT

With the Alphanumeric Input Package, any character may be read into the PDP-5 through either the Teletype or the high-speed reader. The characters are packed two/cell and stored in the address indicated in the switch register.

DECUS NO. 5/8-43, UNSIGNED OCTAL-DECIMAL FRACTION CONVERSION

This routine accepts a four-digit octal fraction in the accumulator and prints it out as an N-digit decimal fraction where N = 12 unless otherwise specified. After N digits, the fraction is truncated. Programs are included for use on the PDP-5 with Type 153 Automatic Multiply-Divide and the PDP-8 with Type 182 Extended Arithmetic Element.

Storage requirements: 55 Octal locations for the PDP-5. 47 Octal locations for the PDP-8.

DECUS NO. 8-44, MODIFICATIONS TO THE FIXED POINT OUTPUT IN THE PDP-8 FLOATING POINT PACKAGE (Digital 8-5-S)

The Floating Point Package (Digital 8-5-S) includes an Output Controller which allows output in fixed point as well as floating point format. This Output Controller takes the form of a certain number of patches to the "Floating Output E Format" routine, plus an additional page of coding.

Using the Calculator program (Digital 8-10-S), which includes the Floating Point Package, certain deficiencies were noted in the fixed-point output format, particularly the lack of any automatic rounding off. For example, the number 9, if outputted as a single digit, appears as 8. Modification attempts to provide automatic rounding off resulted in the Output Controller being completely rewritten with minor changes in the format.

This new version of the Output Controller is also in the form of patches to the Floating Output with an additional page of coding, thereby not increasing the size of the Floating Point Package.

The following summarizes this new version:

1. The number output is automatically rounded off to the last digit printed, or the sixth significant digit, whichever is reached first. Floating point output is rounded off to six figures since the seventh is usually meaningless.

2. A number less than one is printed with a zero preceding the decimal point (e.g., +0.5 instead of + .5).

3. A zero result, after rounding off, is printed as +0 instead of +.

4. The basic Floating Point Package includes the facility to specify a carriage return/line feed after the number using location 55 as a flag for this purpose. The patches for the Output Controller caused this facility to be lost. This version restores this facility.

DECUS NO' 5/8, PDP-5/8 REMOTE & TIME–SHARED SYSTEMS

A time-shared programming system which allows remote stations immediate access to the computer and a wide selection of programs.

DECUS NO. 5/8-46, PDP-5/8 UTILITY PROGRAMS

Consists of four programs (listed below) each of which may be selected via the teletypewriter. When the program is started, either by a self-starting binary loader or by manually starting the computer in address $200_{(8)}$ it is in its executive mode. In this mode, it will respond only to five keys and perform the following functions:

B–go to BIN to QK Converter Program
E–go to Editor Program
L–type a section of leader and stay in executive
P–go to Page Format Program
Q–go to QK to BIN Converter Program

DECUS NO. 8-47, ALBIN–A PDP-8 LOADER FOR RELOCATABLE BINARY PROGRAMS

ALBIN is a simple method for constructing relocatable binary formatted programs, using the PAL III Assembler. Allocation of these programs can be varied in units of one memory page (128_{10} registers.). When loading an ALBIN program, the actual absolute addresses of indicated program elements (e.g., the keypoint of subroutines) are noted down in fixed program-specified location on page zero. In order to make a DEC symbolic program suitable for translation into its relocatable binary equivalent, minor changes are required, which, however, do not influence the length of the program. Due to its similarity to the standard DEC BIN loader, the ALBIN loader is also able to read-in normal DEC binary tapes. ALBIN requires 122_{10} locations, RIM loader included. Piling-up in core memory of ALBIN programs stored on conventional or DECtape can be achieved using the same method with some modifications.

DECUS NO. 5/8-48, MODIFIED BINARY LOADER MKIV

The Mark IV loader was developed to accomplish four objectives:

1. Incorporate the self-starting format described in DECUS 5/8-27, ERC Boot.
2. Select the reader in use, automatically, without switch register settings.
3. Enable a newly-prepared binary tape to be checked prior to loading by calculating the checksum.
4. Reduce the storage requirements for the loader so that a special program would fit on the last page of memory with it.

DECUS NO. 8-49, RELATIVISTIC DYNAMICS

Prints tables for relativistic particle collisions and decay in the same format as the Oxford Kinematic Tables. It can be used in two ways:

1. Two-particle Collisions–Given the masses of incident, target, and emitted particles, the incident energy and centre-of-mass angles, the program calculates angles and energies of the emitted particles in the Lab frame. If the process is forbidden energetically, program outputs E allowing the threshold energy to be found.

2. Single Particle Decays–By specifying M2 = 0 (target), the problem will be treated as a decay, and similar tables to the above will be printed.

DECUS NO. 5/8-50, ADDITIONS TO SYMBOLIC TAPE FORMAT GENERATOR (DEC-8-21-U)

Performs further useful functions by the addition of a few octal patches. By making the appropriate octal patches via the toggles, the Format Generator can also format FORTRAN tapes, shorten tape by converting space to tabs, and convert the type of tape.

A short binary tape may be made and added on to the end of 8-21-U to edit an original tape that was punched off-line.

The rubout character will cause successive deletion of the previous characters until the last C. R. is reached but not removed. The use of ← will cause the current line to be restarted. Thus an input tape may be prepared off-line without attention to format spacing, mistakes corrected as they occur, and finally passed through the Format Generator to create a correctly formatted, edited, and line-fed on either rolled or fanfold paper tape.

DECUS NO. 5/8-51, CHARACTER PACKING AND UNPACKING ROUTINES

ASCII characters may be packed two to a word and recovered. Control characters are also packable but are preceded by a 37 before being packed into the buffer. The two programs total 663_{10} words.

DECUS NO. 8-52, TINY TAPE EDITOR

This Tiny Tape Character Editor fits in core at the same time as the PAL III or MACRO-8 assemblers. A tape may be duplicated at three speeds and stopped at any character for insertion or deletion. The toggle switches control the speed and the functions desired.

The program occupies 72_{10} registers.

DECUS NO. 5/8-54, TIC-TAC-TOE LEARNING PROGRAM–T^3

This program plays Tic-tac-toe basing its moves on stored descriptions of previously lost games. The main program is written in FORTRAN. There is a short subroutine written in PAL II used to print out the Tic-tac-toe board. The program comes already educated with about 32 lost games stored. Requires FORTRAN Object Time System.

DECUS NO. 5/8-55, PALEX–AN ON-LINE DEBUGGING PROGRAM FOR PDP-5 AND PDP-8

One problem with programs written in Program Assembly Language (PAL) for operation on a PDP-5/8 computer is the danger of an untested program being self-destructive, running wild, destroying other programs residing in memory such as loading programs. PALEX prevents any of the above unwanted operations from occurring while it gives the operator-programmer valuable debugging information and enables him to make changes in his program and try out the modified program. Once running, PALEX cannot be destroyed by any program or instruction in memory, the operator need not touch any manual console controls, and all required information is printed in easy-to-read format on the Teletype console.

DECUS NO. 5/8-56, FIXED POINT TRACE NO. 1

A minimum size monitor program which executes the users program one instruction at a time and reports the contents of the program counter, the octal instruction, the contents of the accumulator and link and the contents of the effective address by means of the ASR-33 Teletype. Storage Requirements: two pages.

DECUS NO. 5/8-57, FIXED POINT TRACE NO. 2

Similar to Fixed Point No. 1 except that the symbolic tape provided has a single origin setting instruction of (6000). Any four consecutive memory pages can be used, with the exception of page zero, by changing this one instruction.

DECUS NO. 8-58, ONE-PAGE DECTAPE ROUTINE (522 CONTROL)

A general-purpose program for reading, writing, and searching of magnetic tape. This program was written for the Type 552 Control. It has many advantages over both the standard DEC routines and also over the DECUS No. 5-46. The routines are one-page long and can be operated with the interrupt on or off. The DEC program delays the calling program while waiting for the unit and movement delays to time-out. This routine returns control to the calling program. This saves ½second every time the tape searches forward and half that time when it reverses. In addition, it will read and write block O,. This program is an advantage over the previous one-page routines in that it allows interrupt operations, doesn't overflow by one location, interrupts the end zone correctly and not as an error, and provides a calling sequence identical to the DEC program.

DECUS NO. 8-59, PALDT–PAL MODIFIED FOR DECTAPE (552 CONTROL)

When assembling programs, PALDT requires that the symbolic tape be read in only once. The program writes on the library tape itself after finding the next available block from the directory. During pass 0 the tape is read in using the entire user's symbol table. During passes 1, 2, 3, as much of the symbol table is used as possible. This means the fewest tape passes as possible. As an added advantage pass 0 ignores blank tape, leader-trailer, line feeds, form feeds, and rub outs; saving space. The whole program decreases the users symbol table by only three pages: one for the DECtape program above, one for pass 0, and one for the minimal length read in buffer.

DECUS No. 8-60, SQUARE ROOT FUNCTION BY SUBTRACTION REDUCTION

A single precision square routine using EAE. This routine is usally faster than the DEC routine and can easily be modified for double precision calculation at only twice the computation time.

DECUS No. 8-61, IMPROVEMENT TO DIGITAL 8-9 F SQUARE ROOT

An improved version of the DEC Single Precision Square Root Routine (without EAE). Saves a few words of storage and execution is speeded up 12 per cent.

DECUS NO. 8-62, HIGH-SPEED READER OPTION FOR FORTRAN COMPILER

Program modification that allows the PDP-8 FORTRAN Compiler to read source tapes through the high-speed reader, and punch on the ASR-33. The program is loaded in over the compiler. It can be punched on an extension of the compiler tape so that by depressing the CONTINUE key, it can be read in immediately following the compiler.

DECUS NO. 5-63, SBUG-4

SBUG-4 allows the PDP-5 to execute one instruction of any given program at a time, returning to SBUG-4 following each instruction and printing out the contents of various registers. This permits following the path of a program which has gone astray or examining some defective operation.

DECUS NO. 8-65, A PROGRAMMED ASSOCIATIVE MULTICHANNEL ANALYSER

The program describes the use of a small computer as an associate analyser with special reference to the PDP-8. The advantages and limitations of the method are discussed in the write-up, and general program algorithms are presented.

DECUS NO. 8-68, ALP PROGRAM

The ALP Program punches labels for paper tapes. When a key is stuck on the on-line Teletype keyboard, no echo is performed, but the PDP-8 outputs a few characters to the Teletype punch which form the outline of the character associated with the key.

The character outlines have a fixed width or 5 lines of tape, followed by 3 blank lines for separation between characters; all 8 columns of the paper tape are used to provide the maximum height of character outlines.

DECUS NO. 5/8-69, LESQ29 AND LESQ11

The purpose of the program is to fit the best sequences of parabolas to a given 400 point data curve in order to remove extraneous noise; rather than rely on a single 400 point parabola least squares fit to approximate a given data curve. Approximately 400 individual parabolas are computed as follows.

LESQ29

Data values 1 through 29 are subjected to a second order Least Squares fit. The median point of the resulting parabola (point $\propto$ 15) is then substituted for the original data value # 15.*

A second parabola is then computed using data values 2 through 30. The median point of this parabola (point # 16) is then substituted for point # 16 of the original data curve.

This procedure is repeated until all data values have been replaced (except for the first and last 14 points which are excluded by the mechanics of the operation).

LESQ11

Process identical to LESQ29 except that an 11 rather than a 29 point smooth interval is used. First point replaced is point # the first and last 5 points are excluded from smoothing.

LESQ11 will preserve higher frequency data than LESQ29 for a given data curve with constant time between data points.

Minimum Hardware: 4K Memory PDP-5 or PDP-8, Teletypewriter (plotter, DECtape optional)

Other Programs Needed: Floating Point Package and appropriate data handling routines.

Storage Requirements: (LESQ11: 400-564; 700-716)
(LESQ29: 400-564; 700-751)

*See B. J. Power, R. N. Hagen, S. O. Johnson, "SPORT, A System for Processing Reactor Transient Data on the IBM-7040 Computer," pp. 4-8, AEC Research and Development Report (IDO-17078), Available from: The Clearinghouse for Federal Scientific and Technical Information, National Bureau of Standards, U. S. Department of Commerce, Springfield, Virginia.

Execution Time: (PDP-5) LESQ11: 1 minute.
LESQ29: 2.5 minutes.

Restrictions: Positive integer data $<3777_8$; time between data points constant.

DECUS NO. 8-70, EAE ROUTINES FOR FORTRAN OPERATING SYSTEM (DEC-08-CFA3)

These are two binary patches to the FORTRAN Operating System which utilizes the Type 182 EAE hardware for single precision multiplication and normalization, replacing the software routines in FOSSIL (the operating system). The binary tape is loaded by the BIN Loader after FOSSIL has been loaded. Execution time of a Gauss-Jordan matrix inversion is reduced by approximately 30%.

Minimum Hardware: PDP-8 with Type 182 EAE

Other Programs Needed: FORTRAN Operating System (DEC-08-CFA3-PB) dated March 2, 1967.

DECUS NO. 8-71, PERPETUAL CALENDAR

The program is designed as a computer demonstration. When a valid date is fed into the computer, the corresponding day of the week is typed out. If an invalid date is given, *YOU GOOFED' TRY AGAIN* is typed out. The program is based on the Gregorian Calendar and is, therefore, limited to years between 1500 and 4095. The upper limit being due to the computer's capacity.

Storage: 20-1333

Minimum Hardware: 4K storage, ASR-33 Teletype

DECUS NO. 8-72, MATRIX INVERSION–REAL NUMBERS

The program inverts a matrix, up to size 12 x 12, of real numbers. The algorithm used is the Gauss-Jordan method. A unit vector of appropriate size is generated internally at each stage. Following the Gauss sweep-out, the matrix is shifted in storage, another unit vector is generated, and the calculation proceeds.

Other Programs Needed: FORTRAN Compiler and FORTRAN Operating System.

Storage: This program uses essentially all core not used by the FORTRAN Operating System.

Execution Time: Actual computation takes less than 10 seconds. Data read-in and read-out may take up to five minutes.

DECUS NO. 8-73, MATRIX INVERSION–COMPLEX NUMBERS

The program inverts a matrix, up to size 6 x 6, of complex numbers. The algorithm used is the Gauss-Jordan method, programmed to carry out complex number calculations. A unit vector of appropriate size is generated internally. Following the Gauss sweep-out, the matrix is shifted, another unit vector is generated, and the calculation proceeds. The print-out of the matrices uses the symbol J to designate the imaginary part, e.g. A = a + jb.

Other Programs Needed: FORTRAN Compiler and FORTRAN Operating System.

Storage: This program uses essentially all core not used by the FORTRAN Operating System.

Execution Time: Actual computation takes less than 10 seconds. Data read-in and read-out may take up to five minutes.

DECUS NO. 8-74, SOLUTION OF SYSTEM OF LINEAR EQUATIONS: AX = B, BY MATRIX INVERSION AND VECTOR MULTIPLICATIONS

This program solves the set of linear algebraic equations AX = B by inverting matrix A using a Gauss-Jordan method. When the inverse matrix has been calculated, it is printed out. At that point, the program requests the

B-vector entries. After read-in of the B-vector, the product is computed and printed out. The program then loops back to request another B-vector, allowing the system to solve many sets of B-vectors without the need to invert matrix A again. Maximum size is 8 x 8.

Other Programs Needed: FORTRAN Compiler and FORTRAN Operating System.

Storage: This program uses essentially all core not used by the FORTRAN Operating System.

Execution Time: Actual computation is less than 10 seconds. Data read-in and read-out may take up to five minutes.

DECUS NO. 8-75, MATRIX MULTIPLICATION–INCLUDING CONFORMING RECTANGULAR MATRICES

This program multiplies two matrices, not necessarily square but which conform for multiplication.

Other Programs Needed: FORTRAN Compiler and FORTRAN Operating System.

Storage: This program uses essentially all core not used by the FORTRAN Operating System.

Execution Time: Actual computation takes less than 10 seconds. Data read-in and read-out may take up to five minutes.

Author's comments regarding the four matrix routines: DECUS Nos. 8-72, -73, -74, -75.

"Each program has been written in FORTRAN for use on the basic PDP-8. The printed output of each program has been organized to efficiently and effectively use the Teletype, yet maintain a readable and meaningful output format. This has been done at the expense of optimizing storage requirements. Thus, each program may be changed to handle slightly larger cases by cutting down on the print-out information. Since the source programs are in FORTRAN, a user may readily change the program to suit his own requirements.

"Another common feature of each program is a print-out of the input data from core. This has been found desirable for checking that data was read in properly, and for purposes of having an accurate record of a given calculation. Also, since each program requires a large amount of the data input, it is suggested that a data tape be made prior to running the program.

"The matrix inversion scheme used is straightforward and gives good results on run-of-the-mill matrices. However, error build-up is quite rapid on an ill-conditioned matrix. This is partly due to errors in the fifth and sixth decimal place caused by the floating point conversion at read-in time, and also to the limited mantissa carried in the PDP-8 floating point word."

DECUS NO. 8/8S-76, PDP NAVIG 2/2

This program utilizes the output of the U. S. Navy's AN/SRN-9 satellite navigation receiver to obtain fixes on a PDP-8 or PDP-8/S. This program, except for some details of input and output, follows very closely NAVIG 2 written for the IBM 1620 which in turn is derived from the TRIDON program written at the Applied Physics Laboratory of Johns Hopkins University for the IBM 7090.

PDP NAVIG 2/2 is written in PAL III for a 4096 core machine using the ASR-33. Floating point numbers using two 12-bit words as mantissa and one 12-bit word as exponent are employed. The accuracy is slightly less than that using 7 decimal digits per word.

DECUS NO. 8/8S-77, PDP-8 DUAL PROCESS SYSTEM

The purpose of this system is to expedite the programming of multiprocessing problems on the PDP-8 and PDP-8/S. It maximizes both the input speed and the portion of real time actually used for calculations by allowing the program to run during the intervals between issuing I/O commands and the raising of the device flag to signal completion of the command. The technique also allows queuing of input data or commands so that the user need not wait while his last line is being processed, and so that each line of input may be processed as fast as possible regardless of its length. The system uses the interrupt facilities and has less than a 3% overhead on the PDP-8/S (about .1% on the PDP-8).

This method is especially useful for a slower machine where the problem may easily be calculation limited but would, without such a system, beocme I/O Bound.

The program may also be easily extended to handle input from an A/D converter. Here, the input would be buffered by groups of readings terminated either arbitrarily in groups of readings terminated either arbitrarily in groups of N or by zero crossings.

The system requires 600_8 registers for two TTY's plus buffer space. Several device configurations are possible.

This program can increase the I/O to computation efficiency of some programs by 100%. It can do this even for a single Teletype. Each user will probably want to tailor the program to his individual needs.

DECUS NO. 8-78, DIAGNOSE: A VERSATILE TRACE ROUTINE FOR THE PDP-8 COMPUTER WITH EAE

This trace routine will track down logical errors in a program (the *sick* program). Starting at any convenient location in the *sick* program, instructions are executed, one at a time, and a record of all operations is printed out via the Teletype. To avaid tracing proven subroutines, an option is provided to omit subroutine tracing. The present routine is significantly more versatile than two other trace routines in the DECUS library (DECUS Nos. 8-56 and 8-57–Biavati) for the PDP-8 in that it is able to trace *sick* programs containing floating-point, extended arithmetical, and a variety of input-output instructions. Diagnose is, however, at a disadvantage compared with Biavati's first routine (DECUS No. 8-56) in requiring more memory space (five pages as opposed to two); and compared with his second routine (DECUS No. 8-57) in not possessing the trace-suppression features of the latter. The mode of operation of Diagnose is quite different from that of the trace routines of Biavati.

Minimum Hardware: PDP-8 with EAE

Other Programs Needed: Floating Point Package needed for floating point tracing.

Storage: 5(4) pages of memory.

Miscellaneous: Program is relocatable.

DECUS NO. 8-799, TIC-TAC-TOE (TRINITY COLLEGE VERSION)

This TIC-TAC-Toe game is programmed, using internal logic, so that the computer will either win or stalemate, but not lose a game. Either the player or the computer may choose to go first. At the termination of a game, the program restarts for the next game by typing anew the grid code to be followed.

DECUS NO. 8-80, DETERMINATION OF REAL EIGENVALUES OF A REAL MATRIX

This is a two-part program for determining the real eigenvalues of a real-valued matrix. The matrix does not have to be symmetric. Part I uses the power method of iterating on aneigenvector to determine the largest eigenvalue of the matrix. Part II then deflates the matrix using the results of Part I so as to produce a matrix of order one less than that solved for in Part I. Part I can then be relaoded, and the next eigenvalue in line may be calculated. In this, all the real eigenvalues may be computed in order.

DECUS NO. 8-81, A BIN OR RIM FORMAT DATA OR PROGRAM TAPE GENERATOR

This program enables a PDP-8 operator to generate tapes under Teletype control in RIM or PAL BIN format without formal assembly, assuming the operator knows the octal codes corresponding to each instruction. This is particularly useful when one is dealing with small programs for testing interface equipment or when making small modifications to large programs when one does not wish to spend time reassembling the whole program. Often during program debugging, changes are repeatedly toggled into core manually, which leaves no permanent record of the changes made and is prone to error. Tapes generated using this program can be appended to existing BIN or RIM tapes and can then be loaded with the original tape into core with the appropriate loader. Another use of this program is in the preparation of data tapes in RIM or BIN format so that data can be loaded straight into PDP-8 core via the usual leaders. The program also generates leader/trailer code and a checksum under program control.

Storage: Program occupies locations 6000-6077.

DECUS NO. 8-82, LIBRARY SYSTEM FOR 580 MAGNETIC TAPE (PRELIMINARY VERSION)

The system provides for storing program files (or other files) on the 580 Magnetic Tape with PDP-8, and recalling them at will without altering the state of the rest of the computer. In general principle, it is similar to the DECtape Library System, and the only effective storage requirement is the last page of memory.

At present, the system consists of the three programs known as BOOTSTRAP 1, BOOTSTRAP 2, and the LIBRARY Routines.

Bootstrap 1 is a minimal loader program which resides in the last page of memory. Its function is to rewind the tape and load Bootstrap into the last page, automatically transferring control to it. Bootstrap saves the area of core to be used by the system as a record on the magnetic tape, loads the Library Routines into core, and transfers control to them.

The Library Routines comprise a Directory of the files on tape, an Input-Output package, enabling communication with the Teletype, and four system programs:

LIst: Types out the names of files in the Directory

CAll: Transfers a file into core and exits

DUmp: Writes a file on tape, rewrites the Directory, and exits

EXit: Restores the computer to its original state, with Bootstrap 1 and BIN on the last page.

The magnetic tape subroutines and some control functions are included in Bootstrap 2. Each entry in the directory consists of the three words: the name of the file, its first location in core, and the number of words it occupies. The capacity of the directory is 22_{10} entries.

DECUS NO. 8/8S-83A AND B, OCTAL DEBUGGING PACKAGE (WITH AND WITHOUT FLOATING POINT)

This program is an on-line debugger which will communicate with the operator through the ASR-33 Teletype. It allows register examination and modification, octal dumping, binary punching, multiple and simultaneous breakpoints, starting a program, and running at a particular location with preset AC and link. ODP is completely relocatable at the beginning of all pages except page zero, and is compatible with the PDP-5, the PDP-8, and the PDP-8/S.

Requirements: The high version of ODP requires locations 7000-7577. The low version requires locations 0200-0777. All versions will require three pages. Also, location 0002 is used for a breakpoint pointer to ODP.

Equipment: The standard PDP-8 with ASR-33 Teletype is required. A highspeed punch is optional.

DECUS NO. 8-84, ONE–PASS PAL III

This is a modification to Digital 8-3L-S. It is for use on an 8K PDP-8 with ASR 33. The principle of the modification is to store the incoming characters during Pass I into the memory extension and to take them from there during Pass 2 and 3. Source programs must be limited to 4095 characters. This modification can save about 40% of assembly time.

Operation of the program is the same as for PAL III except that the reading of the source program for Pass 2 and 3 need not be repeated. For these passes, one simply presses CONTINUE after setting the correct switches.

Restriction: The program does not work with high-speed reader and punch.

DECUS NO. 5/8-85, SET MEMORY EQUAL TO ANYTHING

This program will preset all locations to any desired settings. Thus, combining a memory clear, set memory equal to HALT, etc. into a single program. The program is loaded via the switch registers into core.

DECUS NO. 8-86, HIGH-SPEED READER OPTION FOR PDP-8 FORTRAN COMPILER FOR USE WITH DECTAPE-STORED COMPILER

This program is for DECtape installations and utilizes the High-Speed Reader Routine in the DECtape Binary Loader. It is a modification to DECUS 8-82, High-Speed Reader Option. DECUS 8-82 is not usable by installations equipped with DECtape because it utilizes part of the last page reserved for the DECtape loader.

The binary tape of this modification is loaded in over the FORTRAN Compiler, then the entire load put onto DECtape using the routine, *UPDATE*.

DECUS NO. 6/8-122, PDP-8 ASSEMBLER FOR PDP-6

Assembles PDP-8 programs written in PAL on a PDP-6 using any I/O devices.

6.4 DIAGNOSTIC PROGRAMS

PDP-12 PROCESSOR TEST

This diagnostic tests LINC mode processor and memory control instructions.

PDP-12 TC12 TAPE CONTROL TEST

This diagnostic makes use of the TC12 maintenance instructions to thoroughly test the tape control. After all static tests are performed, data transfers and dynamic characteristics are tested.

PDP-12 DISPLAY TEST (VC12)

The diagnostic tests both the VC12 display control and the VR12 display oscilloscope.

PDP-12 CLOCK TEST (KW12)

This diagnostic thoroughly tests the KW12 clock.

PDP-12 A-D TEST (AD12)

This test verifies the logic of the AD12 and allows checking and aalignment all 32 channels of the A-D converter.

PDP-12 CONSOLE TEST

This diagnostic is used to verify the proper functioning of the LEFT SWITCHES, RIGHT SWITCHES, E STOP function, F STOP function, and AUTO RESTART'

Maindec-08-D03A-D Basic JMS and JMP Test

This is a diagnostic program for testing the JMP and JMS instructions of the PDP-8/I.

Maindec-08-D04A-D Random JMP Test

This program tests the JMP instruction of the PDP-12 . Most of memory is used as a JUMP field with a random number generator selecting each JUMP FROM and JUMP TO location.

Maindec-08-D05A-D Random JMP-JMS Test

This is a diagnostic program to test the JMS instruction of the PDP-12 . Random *FROM* and *TO* addresses are selected for each test. The JMP instruction is tested in that each test requires a JMP to reach the JMS.

Maindec-08-D07A-D Random ISZ Test

This program is written to test the ISZ instruction of the PDP-12. An ISZ instruction is placed in a FROM location, and a TO location contains the OPERAND. Part 1 of the program selects FROM, TO, and OPERAND from a random number generator, with the option of holding any or all constant. Part 2 uses a fixed set of FROM, TO, and OPERAND numbers.

Maindec-08-D0AA-D Instruction Test (EAE) Part 3A

This program is a test of the Extended Arithmetic Element Type KE12. The following instructions are tested: MQL, MQA, SHL, LSR, ASR, NMI, SCA. An attempt is made to detect and isolate errors to their most basic faults and to the minimum number of logic cards. Multiply and divide are tested by Maindec-08-D0BA-D.

Maindec-08-D0BA-D Instruction Test (EAE) Part 3B

Divide overflow detection hardware and divide and multiply hardware are tested by using a pseudo random-number generator to produce the parameters for each test. A software simulated divided and multiply are used to test the results of the hardware divide and multiply.

Maindec-08-D1AA-D Memory Power On/Off Test

This program is a Memory Data Validity Test to be used after a simulated power failure.

Maindec-08-D1BA-D Memory Checkboard Test High/Low

Tests memory for core failure on half-selected lines under the worst possible conditions for reading and writing. It is used primarily for testing the operation of memory at marginal voltages.

Maindec-08-D1CA-D Extended Memory Control Part 1

This program exercises and tests Extended Memory instructions CDF, CIF, RDF, RIF, RMF, and RIB, for proper operation. Basically, this program tests the control section of the memory. Data is tested by tests Maindec-08-D1BA-D and Maindec-08-D1DA-D.

Maindec-08-D1DA-D Extended Memory Checkerboard Part 2

This program is a preliminary test for core memory failures on half-selected lines under worst-case conditions of reading and writing. It is used to test memory module X while running the program in memory module Y.

Maindec-08-D2AA-D Teletype Reader Test

Tests performance of the Teletype Model 33 Perforated Tape Reader using the reader to scan a closed-top test tape punched with alternating groups of character codes 000 to 377.

Each character is tested for bits dropped or gained while reading; each group of characters is checked for characters missed entirely or read more than once.

Maindec-08-D2BA-D Exerciser for the Teletype Paper Tape Reader

This is an exerciser program for the Teletype Paper Tape Reader. Test tapes are read in a random start-stop fashion and errors reported on the Teletype printer.

Maindec-08-D2CA-D Teletype Punch Test

Punches a test tape in a predetermined pattern. The tape passes directly from the Teletype punch to the Teletype reader, which checks the pattern for accuracy.

Maindec-08-D2DA-D Teleprinter Test

The Teleprinter Test tests performance of the Teletype 33 Keyboard Printer. There are two parts to the test, selectable by the operator. The first part tests keyboard input by immediately causing the character typed to be printed for comparison. The second part tests continuous operation of the teleprinter by causing a line consisting of the ASCII character set to be repeatedly printed. The latter also tests for correct functioning of the interrupt after a character has been printed.

Maindec-08-D2EA-D High Speed Reader Test

This program tests performance of the Type 750 High Speed Perforated Tape Reader and control by scanning a closed-loop test tape for transmission accuracy. The reader control is tested for correct operation with the PDP-8 interrupt system.

Maindec-08-D2FA-D High Speed Reader Test (PR12)

This is a diagnostic program for the Digitronics 2500, the PR12 and the PR8/I High Speed Paper Tape Readers. The program is divided into three parts, the first of which is a test tape generator that punches test tapes for parts two and three on the high speed punch. Part two is a series of specific tests with module isolation provided for error situations. Part three reads a preselected tape pattern with the choice of random or fixed block lengths and stalls between blocks.

Maindec-08-D2GA-D High Speed Punch Test

This program consists of two seperate tests. The first causes the High Speed Punch Type PP12 and PP8/I to produce a tape containing a sequence of pseudo-random character codes. This tape is checked for accuracy using either the high-speed reader of the Teletype reader.

In the second test, the character code represented by the setting SR_{4-11} is punched repeatedly. The switch setting may be changed while the test is running.

Maindec-08-D6CA-D Calcomp Plotter Test

This program tests the CALCOMP or HOUSTON Plotters and their control. All control and plotting functions are tested.

APPENDIX A
LINC MODE INSTRUCTIONS

The following table lists all LINC mode instructions, their mnemonics octal codes and execution times. Full instruction descriptions will be found in Chapter 3.

Mnemonic	Function	Octal	Time (μsec)
	ADD		
ADD	Add memory to AC (full address)	2000	3.2*
ADA	Add memory to AC (index class)	1100	3.2*
ADM	Add AC to memory (sum also in AC)	1140	3.2*
LAM	Add link and AC to memory (sum also in A)	1200	3.2*
	MULTIPLY		
MUL	Signed multiply	1240	9*
	LOAD		
LDA	Load AC, full register	1000	3.2*
LDH	Load AC, half register	1300	3.2*
	STORE		(sum also in AC)
STC	Store and clear AC (full address)	4000	3.2
STA	Store AC (index class)	1040	3.2*
STH	Store half AC	1340	3.2*
	SHIFT/ROTATE		
ROL N	Rotate left N places	240	1.6-7.0
ROR N	Rotate right N places	300	1.6-7.0
SCR N	Scale right N places	340	1.6-7.0
	OPERATE		
HLT	Halt	0000	1.5
NOP	No operation	0016	1.5
CLR	Clear AC and LINC	0011	1.5
SET	Set register N to contents of register Y	0040	4.8*
JMP	Jump to register Y	6000	1.6*
DJR	Disable JMP return save	0006	1.6
ESF	Enable Special Function	0004	1.6
SFA	Special Functio~ to AC	0024	1.6
QAC	MQ transfer to AC	0005	1.6
PDP	switch to PDP8 mode	0002	1.6

* See Note at end of table.

** See Note at end of table.

Mnemonic	Function	Octal	Time (μsec)
	LOGICAL OPERATIONS		
BCL	Bit clear (any combination of 12-bits)	1540	3.2*
BSE	Bit set (any combination of 12-bits)	1600	3.2*
BCO	Bit complement (any combination of 12 bits)	1640	3.2*
COM	Complement AC	0017	1.6.
	SKIP		
	Skip next instruction if:		
SAE	AC equals memory register Y	1440	3.2*
SHD	Right half AC unequal to specified half of memory register Y	1400	3.2*
SNS N	SENSE switch N is set	0440+N	1.6
SKP	Unconditional skip	0446	1.6
AZE	AC equals 0000 or 7777	0450	1.6
APO	AC contains positive number	0451	1.6
LZE	Link bit equals 0	0452	1.6
IBZ	Between blocks on LINC tape	0453	1.6
FLO	Add overflow is set	0454	1.6
QLZ	MQ low-order bit zero	0455	1.6
SXL N	External level N is low	0400+N	1.6
KST	Keyboard has been struck	0415	1.6
SRO	Rotate memory register right one place; then if bit 0 of Y equals 0, skip next instruction	1500	3.2*
XSK	Index memory register Y, skip when contents of Y equal 1777	0200	3.2
	INPUT/OUTPUT		
ATR	AC to relay buffer	0014	1.6
RTA	Relay buffer to AC	0015	1.6
SAM N	Sample analog chan N	0100+N	3.2-19**
DIS	Display point on oscilloscope	0140	3.2-17**
DSC	Display character on oscilloscope (2 x 6 matrix)	1740	4.8-51**
RSW	RIGHT SWITCH register to AC	0516	1.6
LSW	LEFT SWITCH register to AC	0517	1.6
IOB	Execute input/output control through IO Bus	0500	5.9

Mnemonic		Octal	
	MEMORY		
LIF	Load instruction field buffer with N	0600	1.6
LDF	Load data field register with N	0640	1.6
IOB		0500	
RIF	Read instruction field	6224	5.9
IOB		0500	
RDF	Read data field	6214	5.9
IOB		0500	
RIB	Read interrupt buffer (Save Field)	6234	5.9
IOB		0500	
RMF	Restore memory fields	6244	5.9
	LINC TAPE		
RDE	Read one block into memory	0702	3.2**
RDC	Read and check one block	0700	3.2**
RCG	Read and check N consecutive	0701	3.2**
WRI	Write one block on tape	0706	3.2**
WRC	Write and check one block	0704	3.2**
WCG	Write and check N blocks	0705	3.2**
CHK	Check one block of tape	0707	3.2**
MTB	Move tape toward selected block	0703	3.2**
AXO	AC to Tape Extended Operations Buffer	0001	1.6
XOA	Extended Operations Buffer to AC	0021	1.6
TMA	AC to TMA setup register	0023	1.6
STD	Skip if tape done (CSKL16)	0416	1.6
TWC	Skip if tape word complete (CSKL17)	0417	1.6

*Indicates both direct and indirect addressing, add 1.6 μsec when indirect addressing is used.

**Indicates additional time dependent on I/O device such as tape, see Instruction Description, Chapter 3.

APPENDIX B
PDP-8 INSTRUCTIONS

TABLE B-1. PDP-8 MODE MEMORY REFERENCE INSTRUCTIONS

Mnemonic Symbol	Operation Code	Direct Addr.		Indirect Addr.		Operation
		States Entered	Execution Time (μsec)	States Entered	Execution Time (μsec)	
AND Y	0000	F,E	3.2	F,D,E	4.8	Logical AND. The AND operation is performed between the content of memory location Y and the content of the AC. The result is left in the AC, the original content of the AC is lost, and the content of Y is restored. Corresponding bits of the AC and Y are operated upon independently. $AC_j \wedge Y_j => AC_j$
TAD Y	1000	F,E	3.2	F,D,E	4.8	2's complement add. The content of memory location Y is added to the content of the AC in 2's complement arithmetic. The result of this addition is held in the AC, the original content of the AC is lost, and the content of Y is restored. If there is a carry from ACO, the link is complemented. AC + Y => AC
ISZ Y	2000	F,E	3.2	F,D,E	4.8	Increment and skip if zero. The content of memory location Y is incremented by one. If the resultant content of Y equals zero, the content of the PC is incremented and the next instruction is skipped. If the resultant content of Y does not equal zero, the program proceeds to the next instruction. The incremented content of Y is restored to memory. If resultant Y = 0, PC + 1 => PC.

TABLE B-1. PDP-8 MODE MEMORY REFERENCE INSTRUCTIONS (cont.)

Mnemonic Symbol	Operation Code	Direct Addr.		Indirect Addr.		Operation
		States Entered	Execution Time (μsec)	States Entered	Execution Time (μsec)	
DCA Y	3000	F,E	3.2	F,D,E	4.8	Deposit and clear AC. The content of the AC is deposited in core memory at address Y and the AC is cleared. The previous content of memory location Y is lost. AC => Y 0 => AC
JMS Y	4000	F,E	3.2	F,D,E	4.8	Jump to subroutine. The content of the PC is deposited in core memory location Y and the next instruction is taken from core memory location Y + 1. PC + 1 => Y Y + 1 => PC
JMP Y	5000	F	1.6	F,D	3.2	Jump to Y. Address Y is set into the PC so that the next instruction is taken from core memory address Y. The original content of the PC is lost. Y => PC

TABLE B-2. PDP-8 MODE GROUP 1 OPERATE MICROINSTRUCTIONS

Mnemonic Symbol	Octal Code	Sequence	Operation
NOP	7000	–	No operation. Causes a 1.6 μsec program delay.
IAC	7001	3	Increment AC. The content of the AC is incremented by one in 2's complement arithmetic.
RAL	7004	4	Rotate AC and L left. The content of the AC and the L are rotated left one place.
RTL	7006	4	Rotate two places to the left. Equivalent to two successive RAL operations.
RAR	7010	4	Rotate AC and L right. The content of the AC and L are rotated right one place.
RTR	7012	4	Rotate two places to the right. Equivalent to two successive RAR operations.
CML	7020	2	Complement L.
CMA	7040	2	Complement AC. The content of the AC is set to the 1's complement of its current content.
CIA	7041	2,3	Complement and increment accumulator. Used to form 2's complement.
CLL	7100	1	Clear L.
CLL RAL	7104	1,4	Shift positive number one left.
CLL RTL	7106	1,4	Clear link, rotate two left.
CLL RAR	7110	1,4	Shift positive number one right.
CLL RTR	7112	1,4	Clear link, rotate two right.
STL	7120	1,2	Set link, The L is set to contain a binary 1.
CLA	7200	1	Clear AC. To be used alone or in OPR 1 combinations.
CLA IAC	7201	1,3	Set AC = 1.
GLK	7204	1,4	Get link, Transfer L into AC 11.
CLA CLL	7300	1	Clear AC and L.
STA	7240	2	Set AC = –1. Each bit of the AC is set to contain a 1.

TABLE B-3. PDP-8 MODE GROUP 2 OPERATE MICROINSTRUCTIONS

Mnemonic Symbol	Octal Code	Sequence	Operation
HLT	7402	3	Halt. Stops the program after completion of the cycle in process. If this instruction is combined with others in the OPR 2 group the other operations are completed before the end of the cycle.
OSR	7404	2	OR with Right Switch register. The OR function is performed between the content of the RSW and the content of the AC, with the result left in the AC.
SKP	7410	1	Skip, unconditional. The next instruction is skipped.
SNL	7420	1	Skip if L $\neq$ 0.
SZL	7430	1	Skip if L = 0.
SZA	7440	1	Skip if AC = 0.
SNA	7450	1	Skip if AC $\neq$ 0.
SZA SNL	7460	1	Skip if AC = 0, or L = 1, or both.
SNA SZL	7470	1	Skip if AC $\neq$ 0 and L = 0.
SMA	7500	1	Skip on minus AC. If the content of the AC is a negative number, the next instruction is skipped.
SPA	7510	1	Skip on positive AC. If the content of the AC is a positive number, the next instruction is skipped.
SMA SNL	7520	1	Skip if AC $<$ 0, or L = 1, or both.
SPA SZL	7530	1	Skip if AC $\geqslant$ 0 and if L = 0.
SMA SZA	7540	1	Skip if AC $<$ 0.
SPA SNA	7550	1	Skip if AC $\geqslant$ 0.
CLA	7600	2	Clear AC. To be used alone or in OPR 2 combinations.
LAS	7604	1,2	Load AC with SR.
SZA CLA	7640	1,2	Skip if AC = 0, then clear AC.
SNA CLA	7650	1,2	Skip if AC $\neq$ 0, then clear AC.
SMA CLA	7700	1,2	Skip if AC $<$ 0, then clear AC.
SPA CLA	7710	1,2	Skip if AC $\geqslant$ 0, then clear AC.

TABLE B-4. PDP-8 MODE EXTENDED ARITHMETIC ELEMENT MICROINSTRUCTIONS

Mnemonic Symbol	Octal Code	Sequence	Operation
MUY	7405	3	Multiply. The number held in the MQ is multiplied by the number held in core memory location PC + 1 (or the next successive core memory location after the MUY Command). At the conclusion of this command the most significant 12 bits of the product are contained in the AC and the least significant 12 bits of the product are contained in the MQ. Y x MQ = > AC, MQ
DVI	7407	3	Divide. The 24-bit dividend held in the AC (most significant 12 bits) and the MQ (least significant 12 bits) is divided by the number held in core memory location PC + 1 (or the next successive core memory location following the DVI command). At the conclusion of this command the quotient is held in the MQ, the remainder is in the AC, and the L contains a 0. If the L contains a 1, divide overflow occured so the operation was concluded after the first cycle of the division. AC, MQ ÷ Y = > MQ.
NMI	7411	3	Normalize. This instruction is used as part of the conversion of a binary number to a fraction and an exponent for use in floating-point arithmetic. The combined content of the AC and the MQ is shifted left by this one command until the content of AC0 is not equal to the content of AC1, to form the fraction. Zeros are shifted into vacated MQ11 positions for each shift. At the conclusion of this operation, the step counter contains a number equal to the number of shifts performed. The content of L is lost. AC_j = > AC_j−1 AC0 = > L MQ 0 = > AC11 MQ_j = > MQ_j − 1 0 = > MQ11 until AC0 ≠ AC1
SHL	7413	3	Shift arithmetic left. This instruction shifts the combined content of the AC and MQ to the left one position more than the number of positions indicated by the content of core memory at address PC + 1 (or the next successive core memory location following the SHL command). During the shifting, zeros are shifted into vacated MQ11 positions. Shift Y + 1 positions as follows: AC_j = > AC_j − 1 AC0 = > L MQ0 = > AC11 MQ_j = > MQ_j − 1 0 = > MQ 11

TABLE B-4. PDP-8 MODE EXTENDED ARITHMETIC ELEMENT MICROINSTRUCTIONS (cont.)

Mnemonic Symbol	Octal Code	Sequence	Operation
ASR	7415	3	Arithmetic shift right. The combined content of the AC and the MQ is shifted right one position more than the number contained in memory location PC + 1 (or the next successive core memory location following the ASR command). The sign bit, contained in AC0, enters vacated positions, the sign bit is preserved, information shifted out of MQ11 is lost, and the L is undisturbed during this operation. Shift Y + 1 positions as follows: AC0 = > AC0 AC_j = > $AC_j + 1$ AC11 = > MQ0 MQ_j = > $MQ_j + 1$
LSR	7417	3	Logical shift right. The combined content of the AC and MQ is shifted left one position more than the number contained in memory location PC + 1 (or the next successive core memory location following the LSR command). This command is similar to the ASR command except that zeros enter vacated positions instead of the sign bit entering these locations. Information shifted out of MQ11 is lost and the L is undisturbed during this operation. Shift Y + 1 positions as follows: 0 = > AC0 AC_j = > $AC_j + 1$ AC11 = > MQ0 MQ_j = > $MQ_j + 1$
MQL	7421	2	Load multiplier quotient. This command clears the MQ, loads the content of the AC into the MQ, then clears the AC. 0 = > MQ AC = > MQ 0 = > AC
SCA	7441	2	Step counter load into accumulator. The content of the step counter is transferred into the AC. The AC should be cleared prior to issuing this command or the CLA command can be combined with the SCA to clear the AC, then effect the transfer. SC V AC = > AC
SCL	7403	3	Step counter load from memory. Loads complement of bits 7 through 11 of the word in memory following the instruction into the step counter. MB_{7-11} = > SC PC + 2 = > PC

TABLE B-4. PDP-8 MODE EXTENDED ARITHMETIC ELEMENT MICROINSTRUCTIONS (cont.)

Mnemonic Symbol	Octal Code	Sequence	Operation
MQA	7501	2	Multiplier quotient load into accumulator. The content of the MQ is transferred into the AC. This command is given to load the 12 least significant bits of the product into the AC following a multiplication or to load the quotient into the AC following a division. The AC should be cleared prior to issuing this command or the CLA command can be combined with the MQA to clear the AC then effect the transfer. MQ V AC = > AC
CLA	7∪01	1	Clear accumulator. The AC is cleared during sequence 1, allowing this command to be combined with the other EAE commands that load the AC during sequence 2 (such as SCA and MQA). 0 = > AC
CAM	7621	1,2	Clear accumulator and multiplier quotient. CAM = CLA LMQ.

APPENDIX C
I/O BUS INSTRUCTIONS

In PDP-8 mode these instructions are given directly while in LINC mode they are preceded by the instruction IOB (0500).

TABLE C-1. IOT INSTRUCTIONS

Mnemonic	Octal	Operation
Program Interrupt		
ION	6001	Turn interrupt on and enable the computer to respond to an interrupt request. When this instruction is given, the computer executes the next instruction, then enables the interrupt. The additional instruction allows exit from the interrupt subroutine before allowing another interrupt to occur.
IOF	6002	Turn interrupt off i.e., disable the interrupt.
High Speed Perforated Tape Reader and Control		
RSF	6011	Skip if reader flag is a 1.
RRB	6012	Read the content of the reader buffer and clear the reader flag. (This instruction does not clear the AC). RB V AC 4-11 = > AC 4-11.
RFC	6014	Clear reader flag and reader buffer, fetch one character from tape and load it into the reader buffer, and set the reader flag when done.
High Speed Perforated Tape Punch and Control		
PSF	6021	Skip if punch flag is a 1.
PCF	6022	Clear punch flag and punch buffer.
PPC	6024	Load the punch buffer from bits 4 through 11 of the AC and punch the character. (This instruction does not clear the punch flag or punch buffer). AC 4-11 V PB = > PB
PLS	6026	Clear the punch flag, clear the punch buffer, load the punch buffer from the content of bits 4 through 11 of the accumulator, punch the character, and set the punch flag to 1 when done.
Teletype Keyboard/Reader		
KSF	6031	Skip if keyboard flag is a 1.
KCC	6032	Clear AC and clear keyboard flag.
KRS	6034	Read keyboard buffer static. (This is a static command in that neither the AC nor the keyboard flag is cleared). TTI V AC 4-11 = > AC 4-11
KRB	6036	Clear AC, clear keyboard flag, and read the content of the keyboard buffer into the content of AC 4-11.

TABLE C-1. IOT INSTRUCTIONS (cont.)

Mnemonic	Octal	Operation
Teletype Teleprinter/Punch		
TSF	6041	Skip if teleprinter flag is a 1.
TCF	6042	Clear teleprinter flag.
TPC	6044	Load the TTO from the content of AC 4-11 and print and/or punch the character.
TLS	6046	Load the TTO from the content of AC 4-11, clear the teleprinter flag, and print and/or punch the character.
Automatic Restart Type KP12		
SPL	6102	Skip if power is low.
LINC	6141	Go to LINC mode
Memory Extension Control Type MC12		
CDF	62N1	Change to data field N. The data field register is loaded with the selected field number (0 to 7). All subsequent memory requests for operands are automatically switched to that data field until the data field number is changed by a new CDF command.
CIF	62N2	Prepare to change to instruction field N. The instruction buffer register is loaded with the selected field number (0 to 7). The next JMP or JMS instruction causes the new field to be entered.
RDF	6214	Read data field into AC 6-8. Bits 0-5 and 9-11 of the AC are not affected.
RIF	6224	Same as RDF except reads the instruction field.
RIB	6234	Read interrupt buffer. The instruction field and data field stored during an interrupt are read into AC 6-8 and 9-11 respectively.
RMF	6244	Restore memory field. Used to exit from a program interrupt.
Plotter and Control Type XY12		
PLSF	6501	Skip if plotter flag is a 1.
PLCF	6502	Clear plotter flag.
PLPU	6504	Plotter pen up. Raise pen off of paper.
PLPR	6511	Plotter pen right.
PLDU	6512	Plotter drum (paper) upward.
PLDD	6514	Plotter drum (paper) downward.
PLPL	6521	Plotter pen left.
PLUD	6522	Plotter drum (paper) upward. (Same as 6512.)
PLPD	6524	Plotter pen down. Lower pen on to paper.
Random Access Disc File (Type DF32)		
DCMA	6601	Clears memory address register, parity error and completion flags. This instruction clears the disc memory request flag and interrupt flags.

TABLE C-1. IOT INSTRUCTIONS (cont.)

Mnemonic	Octal	Operation
DMAR	6603	The contents of the AC are loaded into the disc memory address register and the AC is cleared. Begin to read information from the disc into the specified core location. Clears parity error and completion flags. Clears interrupt flags.
DMAW	6605	The contents of the AC are loaded into the disc memory address register and the AC is cleared. Begin to write information into the disc from the specified core location. Clears parity error and completion flags.
DCEA	6611	Clears the disc extended address and memory address extension register.
DSAC	6612	Skips next instruction if address confirmed flag is a 1. (AC is cleared.)
DEAL	6615	The disc extended address extension registers are cleared and loaded with the track data held in the AC.
DEAC	6616	Clear the AC then loads the contents of the disc extended address register into the AC to allow program evaluation. Skip next instruction if address confirmed flag is a 1.
DFSE	6621	Skips next instruction if parity error, data request late, or write lock switch flag is a zero. Indicates no errors.
DFSC	6622	Skip next instruction if the completion flag is a 1. Indicates data transfer is complete.
DMAC	6626	Clear the AC then loads contents of disc memory address register into the AC to allow program evaluation.
RF08/RS08	**Disk Memory**	
DCMA	6601	Clears memory address register, parity error and completion flags. This instruction clears the disc memory request flag and interrupt flags.
DMAR	6603	The contents of the AC are loaded into the disc memory address register and the AC is cleared. Begin to read information from the disc into the specified core location. Clears parity error and completion flags. Clears interrupt flags.
DMAW	6605	The contents of the AC are loaded into the disc memory address register and the AC is cleared. Begin to write information into the disc from the specified core location. Clears parity error and completion flags.
DCIM	6611	Clears the disc interrupt flag and memory address extension register.
DSAC	6612	Skips next instruction if address confirmed flag is a 1. (AC is cleared.)
DIML	6615	Accumulator to interrupt enables and memory extension register.
DIMA	6616	Status to AC
DFSE	6621	Skip on error condition
DFSC	6622	Skip next instruction if the completion flag is a 1. Indicates data transfer is complete.
DMAC	6626	Clear the AC then loads contents of disc memory address register into the AC to allow program evaluation.

TABLE C-1. IOT INSTRUCTIONS (cont.)

Mnemonic	Octal	Operation
CLSK	6131	Skip if Clock Interrupt condition exists
CLLR	6132	AC to Clock Control register.
CLAB	6133	AC to Clock Buffer-Preset register
CLEN	6134	AC to Clock Enable register
CLSA	6135	Clock Status to AC
CLBA	6136	Clock Buffer-Preset register to AC
CLCA	6137	Clock Counter to Buffer Preset register and AC
Automatic Line Printer and Control Type 645		
LSE	6651	Skip if line printer error flag is a 1.
LCF	6652	Clear line printer done and error flags.
LLB	6654	Load printing buffer from the content of AC 6-11 and clear the AC.
LSD	6661	Skip if the printer done flag is a 1.
LCB	6662	Clear both sections of the printing buffer.
LPR	6664	Clear the format register, load the format register from the content of AC 9-11, print the line contained in the section of the printer buffer loaded last, clear the AC, and advance the paper in accordance with the selected channel of the format tape if the content of AC 8 = 1. If the content of AC 8 = 0, the line is printed and paper advance is inhibited.
Card Reader and Control Type CR12		
RCSF	6631	Skip if card reader data ready flag is a 1.
RCRA	6632	The alphanumeric code for the column is read into AC6-11, and the data ready flag is cleared.
RCRB	6634	The binary data in a card column is transferred into AC0-11, and the data ready flag is cleared.
RCSP	6671	Skip if card reader card done flag is a 1.
RCSE	6672	Clear the card done flag, select the card reader and start card motion towards the read station, and skip if the reader-not-ready flag is a 1.
RCRD	6674	Clear card done flag.
Automatic Magnetic Tape Control Type TC58		
MTSF	6701	Skip on error flag or magnetic tape flag. The status of the error flag (EF) and the magnetic tape flag (MTF) are sampled. If either or both are set to 1, the content of the PC is incremented by one to skip the next sequential instruction.

Mnemonic	Octal	Operation
MTCR	6711	Skip on tape control ready (TCR). If the tape control is ready to receive a command, the PC is incremented by one to skip the next sequential instruction.
MTTR	6721	Skip on tape transport ready (TTR). The next sequential instruction is skipped if the tape transport is ready.
MTAF	6712	Clear the status and command registers, and the EF and MTF if tape control ready. If tape control not ready, clears MTF and EF flags only.
– –	6724	Inclusively OR the contents of the command register into bits 0-11 of the AC.
MTCM	6714	Inclusively OR the contents of AC bits 0-5, 9-11 into the command register; JAM transfer bits 6, 7, 8 (command function).
MTLC	6716	Load the contents of AC bits 0-11 into the command register.
– –	6704	Inclusively OR the contents of the status register into bits 0-11 of the AC.
MTRS	6706	Read the contents of the status register into bits 0-11 of the AC.
MTGO	6722	Set "to" bit to execute command in the command register if command is legal.
– –	6702	Clear the accumulator.
General Purpose Converter and Multiplexer Control Type AF01A		
ADSF	6531	Skip if A/D converter flag is a 1.
ADVC	6532	Clear A/D converter flag and convert input voltage to a digital number, flag will set to 1 at end of conversion. Number of bits in converted number determined by switch setting, 11 bits maximum.
ADRB	6534	Read A/D converter buffer into AC, left justified, and clear flag.
ADCC	6541	Clear multiplexer channel address register.
ADSC	6542	Set up multiplexer channel as per AC 6-11. Maximum of 64 single ended or 32 differential input channels.
ADIC	6544	Index multiplexer channel address (present address + 1). Upon reaching address limit, increment will cause channel 00 to be selected.
Guarded Scanning Digital Voltmeter Type AF04A		
VSEL	6542	The contents of the accumulator are transferred to the AF04A control register.
VCNV	6541	The contents of the accumulator are transferred to the AF04A channel address register. Analog signal on selected channel is automatically digitized.
VINX	6544	The last channel address is incremented by one and the analog signal on the selected channel is automatically digitized.
VSDR	6531	Skip if data ready flag is a 1.
VRD	6532	Selected byte of voltmeter is transferred to the accumulator and the data ready flag is cleared.
VBA	6534	BYTE ADVANCE command requests next twelve bits, data ready flag is set.
VSCC	6571	SAMPLE CURRENT CHANNEL when required to digitize analog signal on current channel repeatedly.

APPENDIX D
PDP-8 MODE PERFORATED – TAPE LOADER

READIN MODE LOADER

The readin mode (RIM) loader is a minimum length, basic, perforated-tape reader program for the 33 ASR. It is initially stored in memory by manual use of the operator console keys and switches. The loader is permanently stored in 18 locations of page 37.

A perforated tape to be read by the RIM loader must be in RIM format:

Tape Channel 8 7 6 5 4 S 3 2 1	Format
1 0 0 0 0 . 0 0 0	Leader-trailer code
0 1 A1 . A2 0 0 A3 . A4	Absolute address to contain next 4 digits
0 0 X1 . X2 0 0 X3 . X4	Content of previous 4-digit address
0 1 A1 . A2 0 0 A3 . A4	Address
0 0 X1 . X2 0 0 X3 . X4 (Etc.)	Content (Etc.)
1 0 0 0 0 . 0 0 0	Leader-trailer code

The RIM loader can only be used in conjunction with the 33 ASR reader (not the high-speed perforated-tape reader). Because a tape in RIM format is, in effect, twice as long as it need be, it is suggested that the RIM loader be used only to read the binary loader when using the 33ASR. (Note that some PDP-12 diagnostic program tapes are in RIM format).

The complete PDP-12 RIM loader (SA = 7756) is as follows:

Absolute Address	Octal Content	Tag	Instruction I Z	Comments
7756	6032	BEG,	KCC	/CLEAR AC AND FLAG
7757,	6031		KSF	/SKIP IF FLAG = 1
7760	5357		JMP .–1	/LOOKING FOR CHARACTER
7761,	6036		KRB	/READ BUFFER
7762,	7106		CLL RTL	
7763,	7006		RTL	/CHANNEL 8 IN ACO
7764,	7510		SPA	/CHECKING FOR LEADER
7765,	5357		JMP BEG +1	/FOUND LEADER

Absolute Address	Octal Content	Tag	Instruction I Z	Comments
7766,	7006		RTL	/OK, CHANNEL 7 IN LINK
7767	6031		KSF	
7770,	5367		JMP .–1	
7771,	6034		KRS	/READ, DO NOT CLEAR
7772,	7420		SNL	/CHECKING FOR ADDRESS
7773,	3776		DCA I TEMP	/STORE CONTENT
7774,	3376		DCA TEMP	/STORE ADDRESS
7775,	5356		JMP BEG	/NEXT WORD
7776,	0	TEMP,	0	/TEMP STORAGE
7777,	5XXX		JMP X	/JMP START OF BIN LOADER

Placing the RIM loader in core memory by way of the operator console keys and switches is accomplished as follows:

1. Set the starting address 7756 in the Left Switches.
2. Set the first instruction (6032) in the Right Switches.
3. Press the Fill Switch.
4. Set the next instruction (6031) in the Right Switches.
5. Press the FILL STEP Switch
6. Repeat steps 4 and 5 until all 16 instructions have been deposited.

To load a tape in RIM format, place the tape in the reader, set the Left Switches to the starting address 7756 of the RIM loader (not of the program being read), press the START LS key, and start the Teletype reader.

Refer to Digital Program Library document Digital-8-1-U for additional information on the Readin Mode Loader program.

BINARY LOADER (PDP-8 MODE)

The binary loader (BIN) is used to read machine language tapes (in binary format) produced by the program assembly language (PAL). A tape in binary format is about one half the length of the comparable RIM format tape. It can, therefore, be read about twice as fast as a RIM tape and is, for this reason, the more desirable format to use with the 10 cps 33 ASR reader or the Type PR12 High Speed Perforated Tape Reader.

The format of a binary tape is as follows:

LEADER: about 2 feet of leader-trailer codes.

BODY: characters representing the absolute, machine language program in easy-to-read binary (or octal) form. The section of tape may contain characters representing instructions (channels 8 and 7 not punched) or origin resettings (channel 8 not punched, channel 7 punched) and is concluded by 2 characters (channels 8 and 7 not punched) that represent a checksum for the entire section.
TRAILER: same as leader.

BODY: characters representing the absolute, machine language program in easy-to-read binary (or octal) form. The section of tape may contain characters representing instructions (channels 8 and 7 not not punched) or origin resettings (channel 8 not punched, channel 7 punched) and is concluded by 2 characters (channels 8 and 7 not punched) that represent a checksum for the entire section.
TRAILER: same as leader.

Example of the format of a binary tape:

Tape Channel Memory 8 7 6 5 4 S 3 2 1	Location	Content	Comments
1 0 0 0 0 . 0 0 0			leader-trailer code
0 1 0 0 0 . 0 1 0 0 0 0 0 0 . 0 0 0		0200	
0 0 1 1 1 . 0 1 0 0 0 0 0 0 . 0 0 0	0200	CLA	origin setting
0 0 0 0 1 . 0 1 0 0 0 1 1 1 . 1 1 1	0201	TAD 277	
0 0 0 1 1 . 0 1 0 0 0 1 1 1 . 1 1 0	0202	DCA 276	
0 0 1 1 1 . 1 0 0 0 0 0 0 0 . 0 1 0	0203	HLT	
0 1 0 0 0 . 0 1 0 0 0 1 1 1 . 1 1 1		0277	origin setting
0 0 0 0 0 . 0 0 0 0 0 1 0 1 . 0 1 1	0277	0053	
0 0 0 0 1 . 0 0 0 0 0 0 0 0 . 1 1 1		1007	sum check
1 0 0 0 0 . 0 0 0			leader-trailer code

After a BIN tape has been read in, one of the two following conditions exists:

a. No checksum error: halt with AC = 0

b. Checksum error: halt with AC = (computed checksum) – (tape checksum)

Operation of the BIN loader in no way depends upon or uses the RIM loader. To load a tape in BIN format place the tape in the reader, set the Left Switches to 7777 (the starting address of the BIN loader), set Right Switch register bit 00 up for loading via the Teletype unit or down for loading via the high speed reader, then press the START LS key, and start the tape reader.

Refer to Digital Program Library document Digital-8-2-U-RIM for additional information on the Binary Loader program.

APPENDIX E
TABLES OF CODES

TABLE E-1. MODEL 33 ASR/KSR TELETYPE CODE (ASCII) IN OCTAL FORM

Character	8-Bit Code (in octal)	Character	8-Bit Code (in octal)
A	301	!	241
B	302	"	242
C	303	#	243
D	304	$	244
E	305	%	245
F	306	&	246
G	307	'	247
H	310	(	250
I	311	)	251
J	312	*	252
K	313	+	253
L	314	,	254
M	315	-	255
N	316	.	256
O	317	/	257
P	320	:	272
Q	321	;	273
R	322	<	274
S	323	=	275
T	324	>	276
U	325	?	277
V	326	@	300
W	327	[	333
X	330	\	334
Y	331	]	335
Z	332	↑	336
		←	337
0	260		
1	261	Leader/Trailer	200
2	262	Line-Feed	212
3	263	Carriage-Return	215
4	264	Space	240
5	265	Rub-out	377
6	266	Blank	000
7	267	**act-mode**	**375**
8	270	**escape**	**233**
9	271		

TABLE E-2. MODEL 33 ASR/KSR TELETYPE CODE (ASCII) IN BINARY FORM

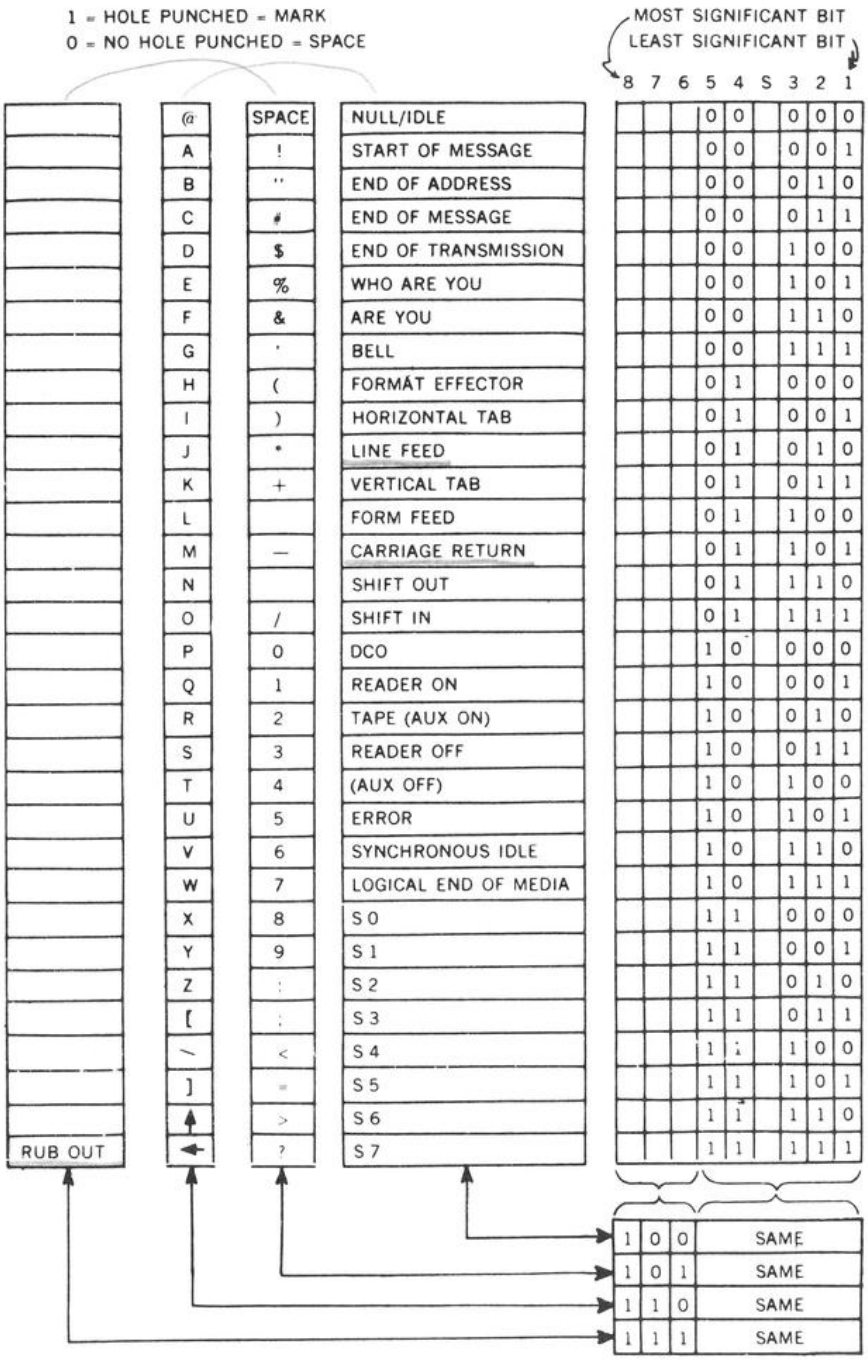


TABLE E-3. CARD READER CODE

Card Code		Internal Code	Character
Zone	Num.		
—	—	01 0000	Blank
12	8-3	11 1011	.
12	8-4	11 1100	)
12	8-5	11 1101	]
12	8-6	11 1110	<
12	8-7	11 1111	◄
12	—	11 0000	+
11	8-3	10 1011	$
11	8-4	10 1100	*
11	8-5	10 1101	[
11	8-6	10 1110	;
11	8-7	10 1111	&
11	—	10 0000	—
0	1	01 0001	/
0	8-3	01 1011	,
0	8-4	01 1100	(
0	8-5	01 1101	"
0	8-6	01 1110	=
0	8-7	01 1111	%
—	8-3	00 1011	#
—	8-4	00 1100	@
—	8-5	00 1101	▲
—	8-6	00 1110	'
—	8-7	00 1111	≥
12	0	11 1010	?
12	1	11 0001	A
12	2	11 0010	B
12	3	11 0011	C
12	4	11 0100	D
12	5	11 0101	E
12	6	11 0110	F
12	7	11 0111	G
12	8	11 1000	H
12	9	11 1001	I

Card Code		Internal Code	Character
Zone	Num.		
11	0	10 1010	▲
11	1	10 0001	J
11	2	10 0010	K
11	3	10 0011	L
11	4	10 0100	M
11	5	10 0101	N
11	6	10 0110	O
11	7	10 0111	P
11	8	10 1000	Q
11	9	10 1001	R
0	8-2	01 1010	!
0	2	01 0010	S
0	3	01 0011	T
0	4	01 0100	U
0	5	01 0101	V
0	6	01 0110	W
0	7	01 0111	X
0	8	01 1000	Y
0	9	01 1001	Z
—	0	00 1010	0
—	1	00 0001	1
—	2	00 0010	2
—	3	00 0011	3
—	4	00 0100	4
—	5	00 0101	5
—	6	00 0110	6
—	7	00 0111	7
—	8	00 1000	8
—	9	00 1001	9
All other codes		00 0000	◄

TABLE E-4. AUTOMATIC LINE PRINTER CODE

Character (ASCII)	6-Bit Code (in octal)	Character (ASCII)	6-Bit Code (in octal)
@	0		40
A	1	!	41
B	2	"	42
C	3	#	43
D	4	$	44
E	5	%	45
F	6	&	46
G	7	'	47
H	10	(	50
I	11	)	51
J	12	*	52
K	13	+	53
L	14	,	54
M	15	-	55
N	16	.	56
O	17	/	57
P	20	0	60
Q	21	1	61
R	22	2	62
S	23	3	63
T	24	4	64
U	25	5	65
V	26	6	66
W	27	7	67
X	30	8	70
Y	31	9	71
Z	32	:	72
[	33	;	73
\	34	<	74
]	35	=	75
↑	36	>	76
←	37	?	77

APPENDIX F
MATHEMATICAL DATA

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^{+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

$$0+1=1 \quad \begin{matrix} 0+0=0 \\ 1+0=1 \\ 1+1=10 \end{matrix}$$

$$0\times 1=0 \quad \begin{matrix} 0\times 0=0 \\ 1\times 0=0 \\ 1\times 1=1 \end{matrix}$$

Octal Scale

0	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

1	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
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

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	1024
to	to
2777	1535
(Octal)	(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	1536
to	to
3777	2047
(Octal)	(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	3238	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	DEC.	OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.
.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	DEC.	OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.
.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	DEC.	OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.
.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	.001010	.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	.0012	.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