Digital Equipment Corporation
Maynard, Massachusetts

digital

PDP-12 system reference manual

laboratory data products

DEC-12-SRZC-D

PDP-12 system reference manual

digital equipment corporation · maynard, massachusetts

1st Printing (Preliminary) June 1970
2nd Printing (Revised) July 1970
3rd Printing (Revised) August 1971
4th Printing October 1971
5th Printing (Revised) July 1972
6th Printing December 1972

CONTENTS

CONTENTS (cont)

CONTENTS (cont)

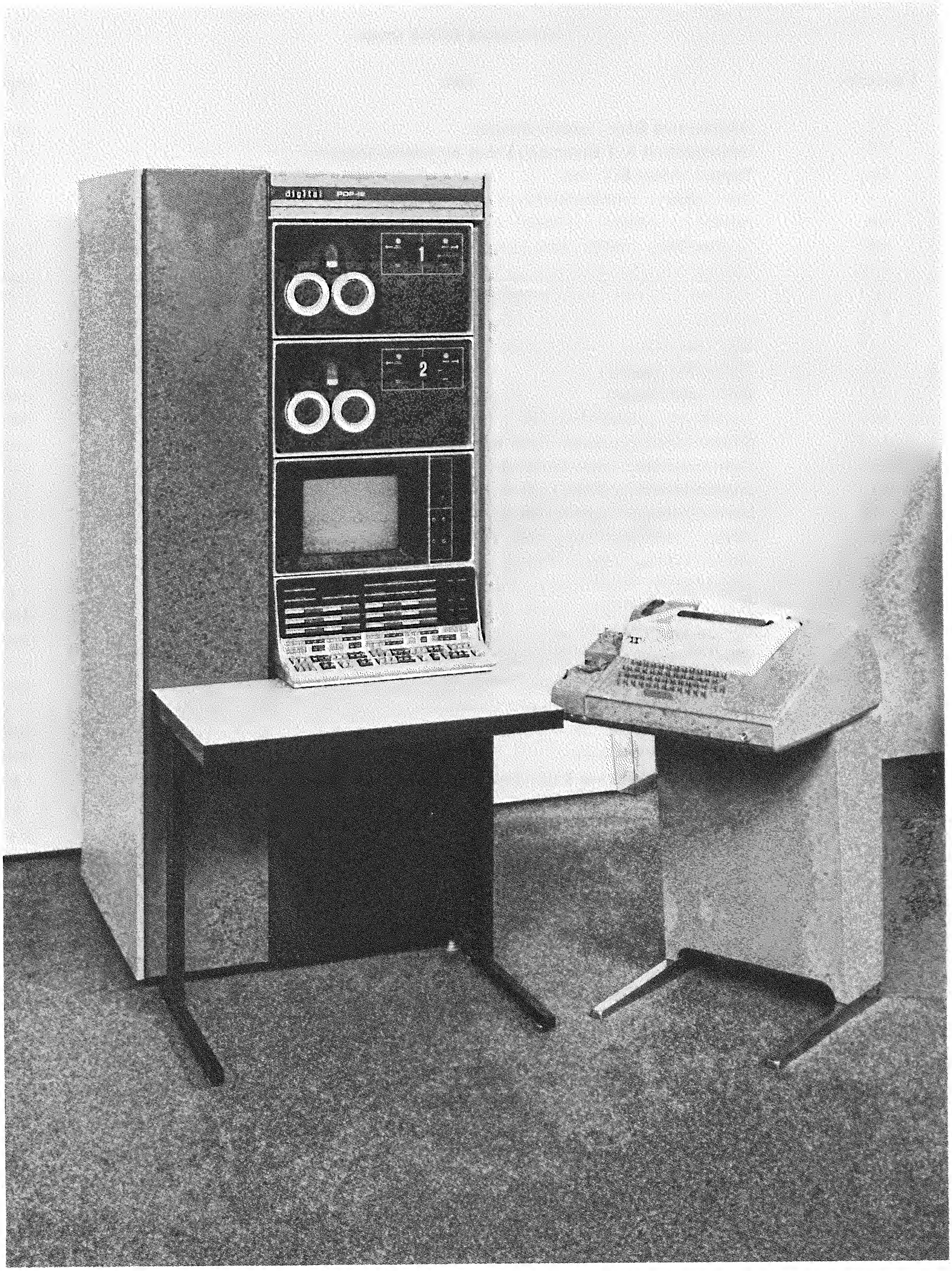
CONTENTS (cont)

CONTENTS (Cont)

TABLES

ILLUSTRATIONS

ILLUSTRATIONS (cont)



PDP-12 Programmed Data Processor

CHAPTER 1
GENERAL DESCRIPTION

1.1 DESCRIPTION

1.1.1 System

The PDP-12 (Programmed Data Processor-12) 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 processor cycle time is 1.6 microseconds ±20%; 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 (Laboratory Instrument Computer) 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 capability. The two smaller systems, B and C, are expandable into the A configuration. The system discussed in this handbook is the PDP-12A. The capacity of basic core memory storage in the PDP-12 is 4096 (4K) 12-bit words which can be expanded to 32,768 (32K) 12-bit words.

Figure 1-1 shows a system block diagram with many of the options and peripherals available, such as:

LINCtape –	Two TU55 tape transports or one TU56 dual drive transport controlled by a buffered subprocessor
CRT Display –	6" x 9" screen, two intensification channels
Analog Inputs –	Eight external inputs, eight variable potentiometers
Relay Buffer –	Six relays for control of external equipment

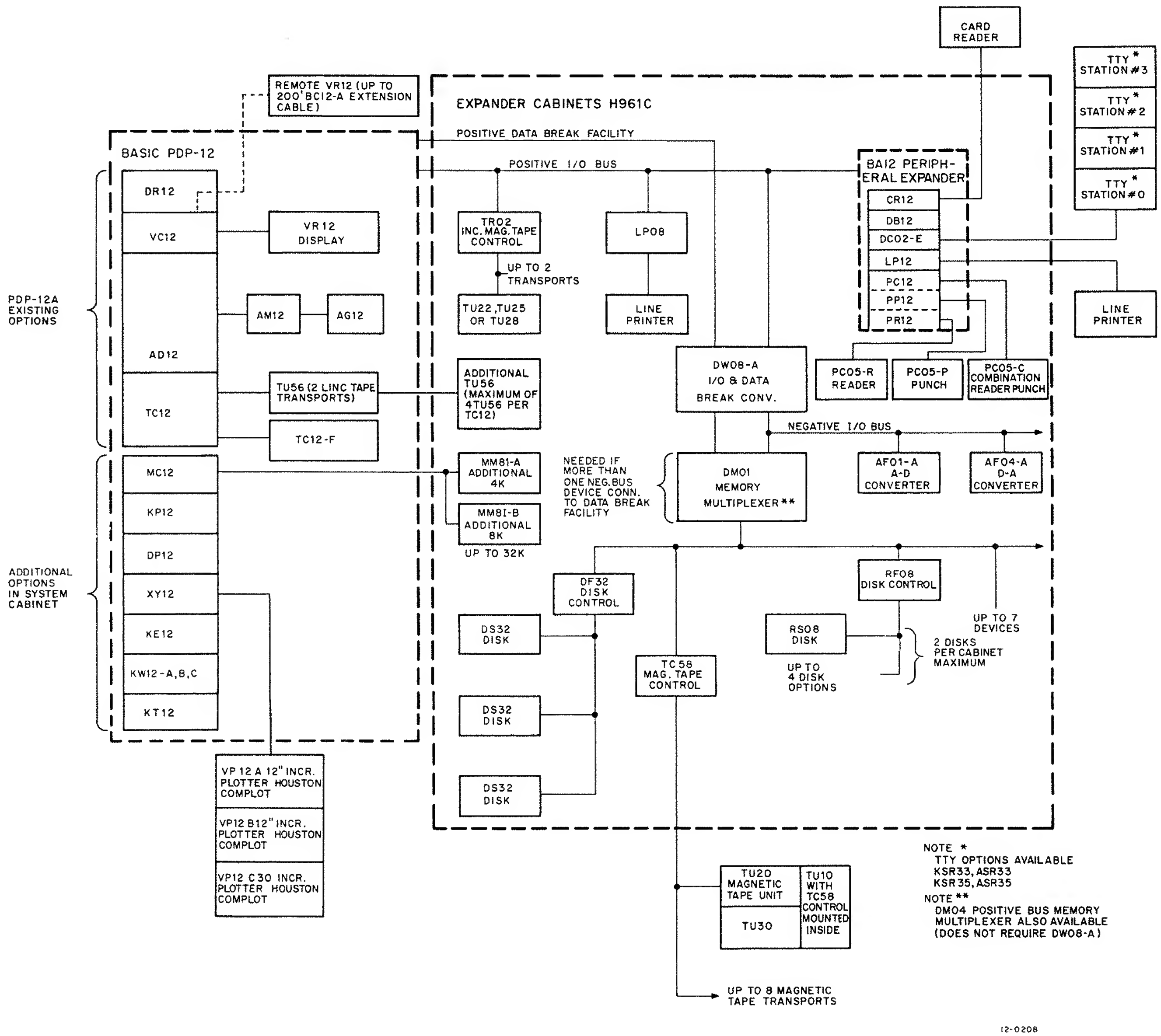


Figure 1-1. PDP-12 Functional Block Diagram

The PDP-12 is also equipped with a positive-logic, PDP-8/I-type, input/output (I/O) bus which can be used for interfacing all 8-family peripherals and options, as well as the standard Teletype® Model 33 ASR.

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 AC_{00} during a 2's complement addition, the Link is complemented. It may be set or cleared independently of the AC under 8 mode control, and may or may not be included in shifting and rotating operations performed on the contents of the AC.
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 8 mode, the PC acts as a 12-bit counter; in LINC mode, it acts as a 10-bit counter.
Instruction Register (IR) 12 Bits	This register contains the complete binary code of the instruction being executed.
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.
Memory Buffer (MB) 12 Bits	All information passing between memory and other registers in the PDP-12 must go through the Memory Buffer Register, whether the transfer involves the central processor, an external I/O device, or another memory register.
Mode Status Register 1 Bit	This register indicates the current operating mode (LINC or 8) of the central processor.

® Teletype is the registered trademark of the Teletype Corporation.

Instruction Field Register (IF) 5 Bits

This register selects the memory field containing the executable program. In the LINC mode, it is used to designate one of up to thirty-two 1024-word segments. In the 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 modules of 4K words may be added, up to a total of eight, or 32,768 words. Within each module, the logical organization of memory depends on the operating mode. In the LINC mode, each module 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 8 mode, the memory field (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 8, are independent of each other, though they can 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. By using the I/O Handler (LINC-8 Simulator Trap Processor) 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 of data transfer to only specific segments of a given memory field. Other features include:

a. Tape Interrupt, which connects the tape processor status to the Program Interrupt.

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

c. Hold-motion, which allows a unit to remain in motion after it has been deselected.

I/O Bus Access – In LINC mode the user has immediate access to those devices activated by LINC instructions: analog inputs, Display, Relays, Sense Lines, and LINCtape. Devices 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, but the program continues to operate in LINC mode.

Special Functions – The LINC programmer may, by setting certain flip-flops,

a. Change the size of characters displayed on the CRT;

b. Enable the program trap, which intercepts certain LINC instruction codes;

c. Disable interrupts from the ASR-33;

d. Speed the sampling of analog inputs;

e. Clear the PDP-12 I/O status by generating an I/O PRESET pulse.

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 the LINC mode, execution of the instruction PDP causes the processor to change immediately to 8 mode operation, and all subsequent instructions are interpreted as PDP-8/I instructions. To switch from 8 mode to LINC mode, the IOT instruction LINC is used.

1.1.5 **Input/Output Facilities and Display**

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 paper tape, magnetic tape, A-D, and card readers, disk storage, and line printers.

Display – The Cathode Ray Tube 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 selectable 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 – A Data Terminal provides a flexible means of receiving analog inputs and controlling the operation of external equipment not internally interfaced to the PDP-12.

Analog Inputs – Sixteen analog inputs feed a 10-bit A-D converter. A single LINC mode instruction samples any one of the 16 channels. Eight of the inputs are taken from phone jacks mounted on the Data Terminal Panel and fed through preamplifiers to the converter. The remaining eight are taken from continuously variable 10-turn potentiometers which are also mounted on the panel. A second set of 16 channels, with preamplifiers, may be added to the basic facility.

Relay Buffer – Six relays, mounted on the data terminal panel, can be switched individually or in combination, by means of a LINC mode instruction. The relays may be used to start and stop operations in external equipment. The states of the relays can be read into the AC.

Auxiliary Scope Connector – A connector mounted on the Data Terminal Panel is wired to accept an auxiliary CRT for displaying information. All display information (X, Y and Z) available to the internal display is also available at the remote connector.

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

LINCtape – Two TU55 transports (or one TU56 dual drive transport) 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 nonstandard formats may be used. A special hardware option, TC12-F, permits the use of tapes with a different format, such as PDP-8 DECtape. 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 (or four dual drive TU56s).

Input/Output (I/O) Bus – This connecting facility provides the control and data transmission path between the central processor and any peripheral device attached to the bus. Some devices, such as paper tape readers and punches, line printers, and incremental plotters, transfer data 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 the 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 Clocks, Type KW12-A, B, or C

Incremental Plotter and Control, Type XY12

TTY/Dataphone®, Type DP12-A, B

With the inclusion of the BA12 Peripheral Expander and the DW08A I/O and Bus Converter, many other devices can be added to the PDP-12 I/O bus. The Peripheral Expander allows the addition of high-speed paper tape reader and punch, card reader, line printers, and optional communication interfaces. The Bus Converter provides for the addition of disk and IBM-compatible magnetic tape storage and A/D converters and associated multiplexers designed for the negative-logic PDP-8 I/O Bus.

Keyboard/Printer (Model 33 ASR) – An important means of direct communication between the user and the operating program is the Model 33 ASR Keyboard/Printer, standard on all configurations of the PDP-12. It is connected to the I/O bus, and can be accessed for input or output by programs in either operating mode. The Model 33 ASR 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 Model 33 ASR operates in full-duplex mode, that is data may be transmitted in both directions simultaneously.

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.
R	General representation of any register.
C (R)	The contents of register R.
$C(R_j)$	The content of bit j of register R.
$C(R_{j-n})$	The contents of bits j through n, inclusive, of register R.

® Dataphone is a registered trademark of A.T.&T.

$C(R_l)$	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 one's complement of the contents of register R.
Y	The effective address of an operand.
I	The Indirect address bit of an instruction. In the LINC mode, I represents bit 7; in the 8 mode, bit 3.
$C(R) \rightarrow C(S)$	The contents of register R replace those of register S.
$N \rightarrow C(R)$	The quantity N replaces the contents of register R.
$\vee$	Inclusive OR
$\not\vee$	Exclusive OR
$\wedge$	AND

CHAPTER 2
CONTROLS AND INDICATORS

This chapter describes the function of the controls and indicators of the PDP-12 computer console, Data Terminal, Type VR12 CRT Display, Type TU55 LINCtape Transport, and Teletype Model 33 ASR.

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 view of the console.

2.2 DATA TERMINAL

The Data Terminal is the area behind the door on the left front of the PDP-12. Normally, up to four separate panels are placed here. A storage rack to hold LINC tapes may be placed in any of the unused spaces of the Data Terminal area. The four standard panels are:

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

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

2.3 CRT DISPLAY, TYPE VR12

Table 2-11 lists the controls and indicators of the Type VR12 CRT Display. Figure 2-4 shows a front view of the CRT Display.

2.4 LINC TAPE TRANSPORT, TYPE TU55

Table 2-12 lists the functions of controls and indicators of the Type TU55 LINCtape transport. Figure 2-5 provides a front view of the transport.

2.5 TELETYPE, MODEL 33 ASR

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

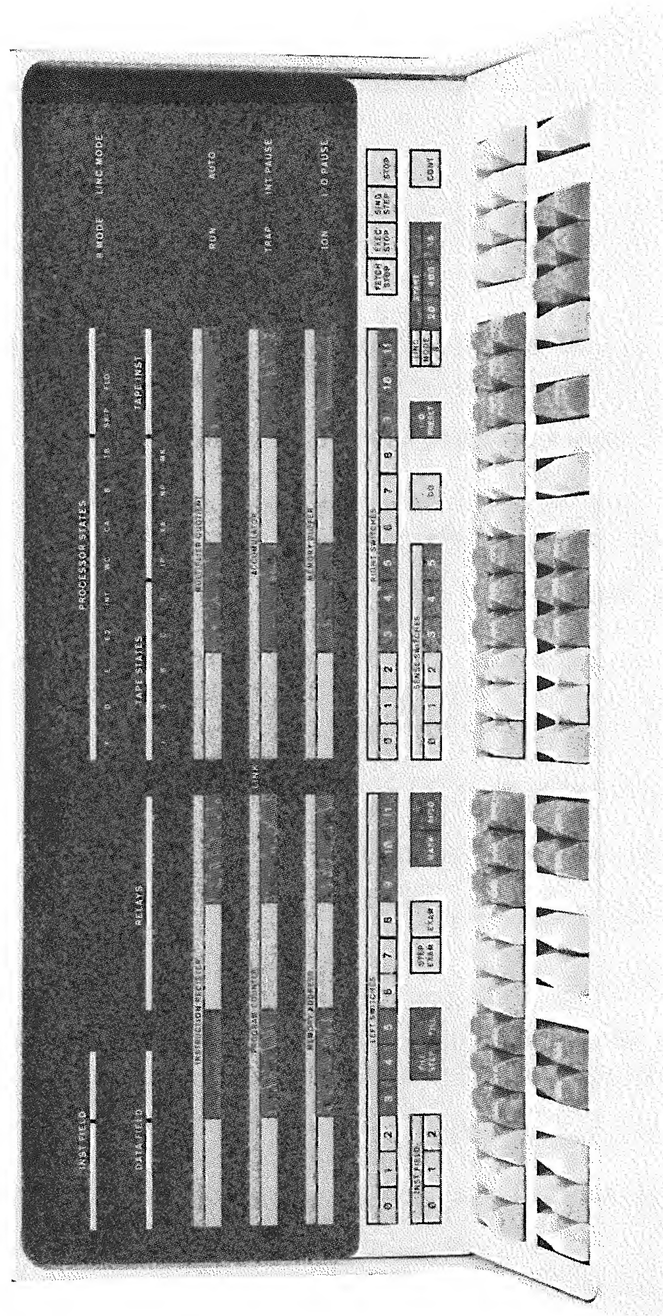


Figure 2-1. PDP-12 Operator console

Table 2-1. Central Processor Register Indicators

Indicator	Bits
INST 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
E 2	Instruction Execution 2
INT	Program Interrupt
WC	Word Count
CA	Current Address
B	Break
TB	Tape Break

Table 2-3. Central Processor Miscellaneous Indicators

Indicators	Interpretation When Lit
SKIP	Skip Flip-Flop is set
FLO	Overflow Flip-Flop is set
8 MODE	Processor is in the PDP-8 Mode
LINC MODE	Processor is in the LINC Mode
RUN	Processor is running
AUTO	Auto Restart Flip-Flop is set
TRAP	Trap flip-flop is set
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
I	Idle
S	Search
B	Block
C	Check Word
T	Turn Around

Table 2-5. Tape Processor Miscellaneous Indicators

Indicator	Interpretation	Function
IP	In Progress	Indicates that a tape operation is In Progress.
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 pulse and the I/O BUS INITIALIZE pulse clear all I/O device flags and operations. The AC and Link are cleared.
DO	This switch causes the processor to perform one instruction. 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). In the 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 Instruction Field.
START 400	This switch causes the processor to start at location 400 of the currently selected Instruction Field.
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	The contents of the Left Switches are transferred into the Memory Address register. The contents of the absolute core address designated by the Left Switches are displayed in the Memory Buffer register.
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 over 10 bits in LINC mode and 12 bits in 8 mode.
FILL	The contents of the Left Switches are transferred into the Memory Address register. The contents of the Right Switches are deposited into the memory location whose absolute address is designated by the Left Switches and the Instruction Field switches.
FILL STEP	Depressing this switch causes the contents of the Right Switches to be deposited into the memory location whose address is in the Memory Address register. Releasing this switch increments the contents of the Memory Address register. This incrementing extends over 10 bits in the LINC mode or 12 bits in the 8 mode. The Memory Buffer register then displays the contents of the location specified by the new contents of the Memory Address register.
MODE	This switch determines the mode (LINC or 8) to which the processor will be set when the I/O PRESET switch is activated. The MODE switch is effective only when the computer is not running.

Table 2-6. Function of Computer Console Keys (cont)

Key	Function
AUTO	This switch sets the AUTO RESTART flip-flop if it is held down at the same time one of the following keys is actuated: STEP EXAM, FILL STEP, DO, or CONT. The AUTO RESTART flip-flop causes the central processor to start automatically at the end of a variable time delay (determined by the console controls) after the central processor stops for any of the following reasons: a. SING STEP switch activated b. FETCH STOP address match c. EXEC STOP address match d. The end of a STEP EXAM operation e. The end of a FILL STEP operation f. The end of a DO switch operation The Auto Restart Flip-Flop is cleared by any of the following conditions: a. STOP switch pressed while processor is running b. DO, FILL STEP, or STEP EXAM switch activated and the AUTO switch not pressed c. A processor HLT instruction executed (either mode) d. I/O PRESET pulse is generated To start a program in AUTO RESTART mode: a. Start the program with the SING STEP switch up. With the AUTO key depressed, depress CONT.

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 EXEC STOP and FETCH STOP functions, and the instruction to be performed when the DO switch is used.
RIGHT SWITCHES	12	The contents of these switches form a 12-bit word which can be read into the accumulator with the LINC mode instruction RSW (516) or the 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 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, EXEC STOP, and FETCH STOP functions.
SENSE SWITCHES	6	These switches are individually interrogated by LINC mode skip instructions, thereby enabling console 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 an 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.
SING 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 Inst Field switches designate the three most significant bits of the address.

Table 2-8. Individual Console Toggle Switches (cont)

Switch	Function
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.

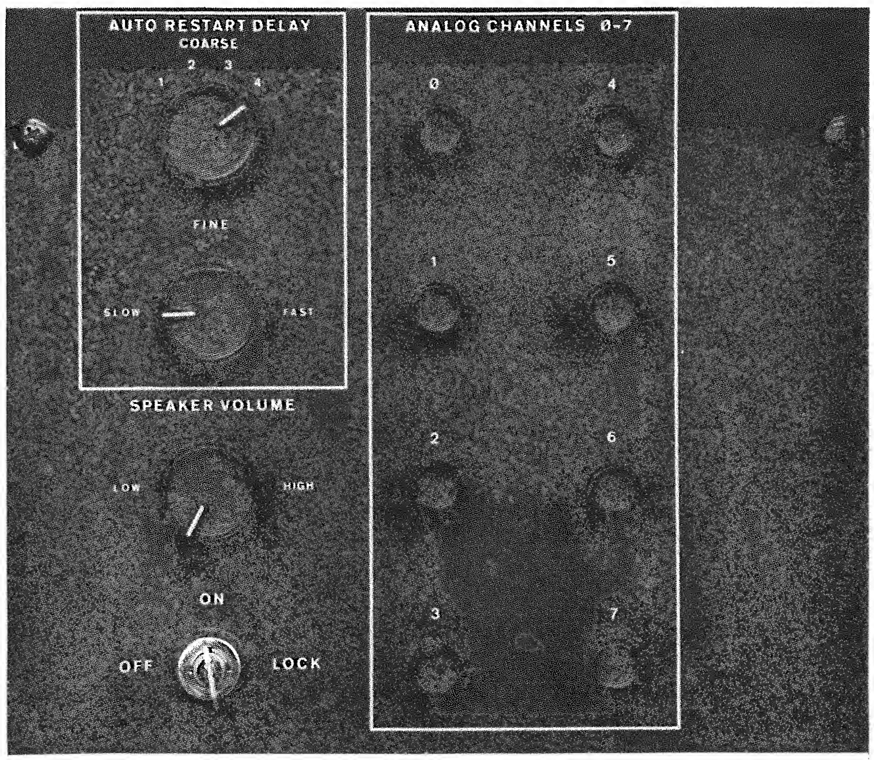


Figure 2-2. Analog Knobs and Power Switch Panel

Table 2-9. Analog Knobs and Power Switch Panel

Panel Controls	Function
OFF/ON/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 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. Only the Left Switches, Right Switches, and Sense Switches remain operative.
SPEAKER VOLUME	The volume of the speaker is controlled by this knob. (The speaker, which is driven by AC bit 0, is added to the system when the AD12 is included in the system configuration.)
AUTO RESTART DELAY	These two knobs control the delay period of Auto Restart after a processor stop due to an EXEC STOP, FETCH STOP, or SING STEP operation. (See AUTO switch description in Table 2-6.)
COARSE and FINE	The COARSE delay selects overlapping ranges from 2.4 μs to .38 sec. The FINE control gives variation within a range of 20:1 of the selected COARSE delay.
ANALOG CHANNELS 0-7 (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 continually-variable parameters within the range of $\pm 512_{10}$ for program usage.

Table 2-10. Relay and Analog Input Panel

Terminals	Function
ANALOG CHANNELS 10-17 (Input Jacks)	These are 3-conductor phone jacks providing ± 1 Volt input connections for AD12 Analog-to-Digital converter channels 10_8-17_8.
RELAY REGISTER Contacts	One set of form C contacts for each of the six system relays is available at the binding posts.
EXTENSION SCOPE	A 24-pin connector for an extension scope provides for remote operations, multiple displays, or photographing of display output. See Appendix G for pin connections and drive characteristics.

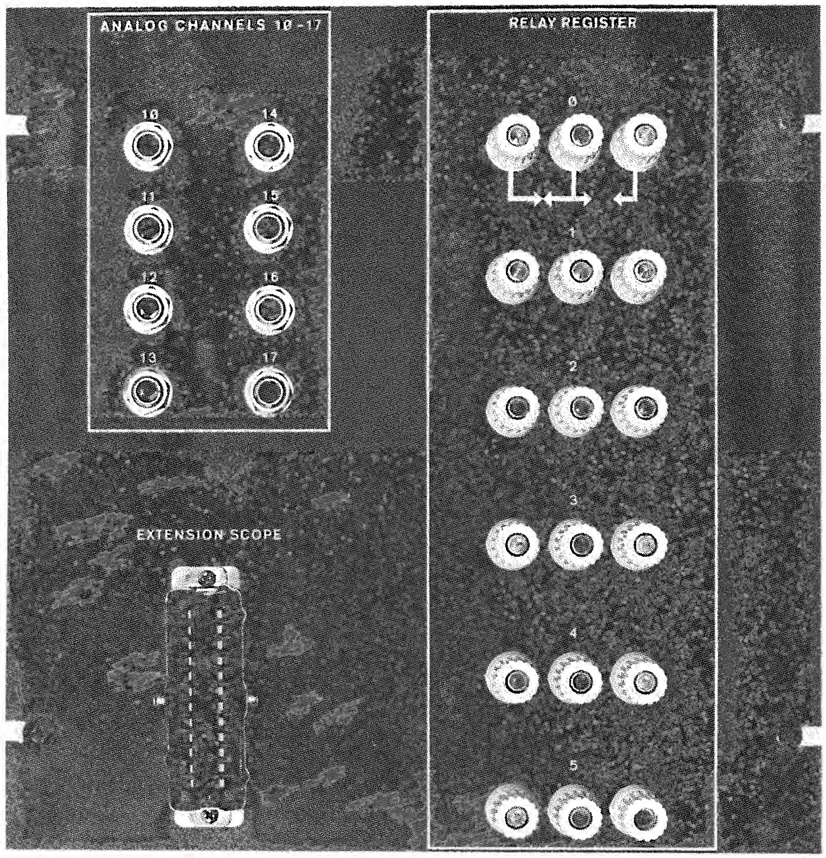


Figure 2-3. Relay and Analog Input Panel

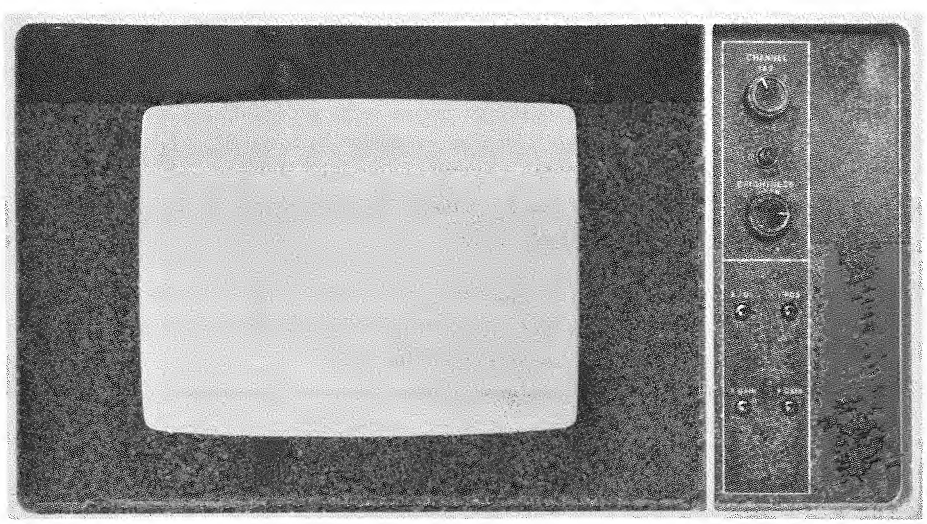


Figure 2-4. Type VR12 CRT Display

Table 2-11. VR12 Display Scope Controls

Control	Function
CHANNEL	Selects one or both channels for scope display.
BRIGHTNESS	Controls level of brightness.
X GAIN	Controls horizontal size of display.
X POS	Controls horizontal position of display.
Y GAIN	Controls vertical size of display.
Y POS	Controls vertical position of display.

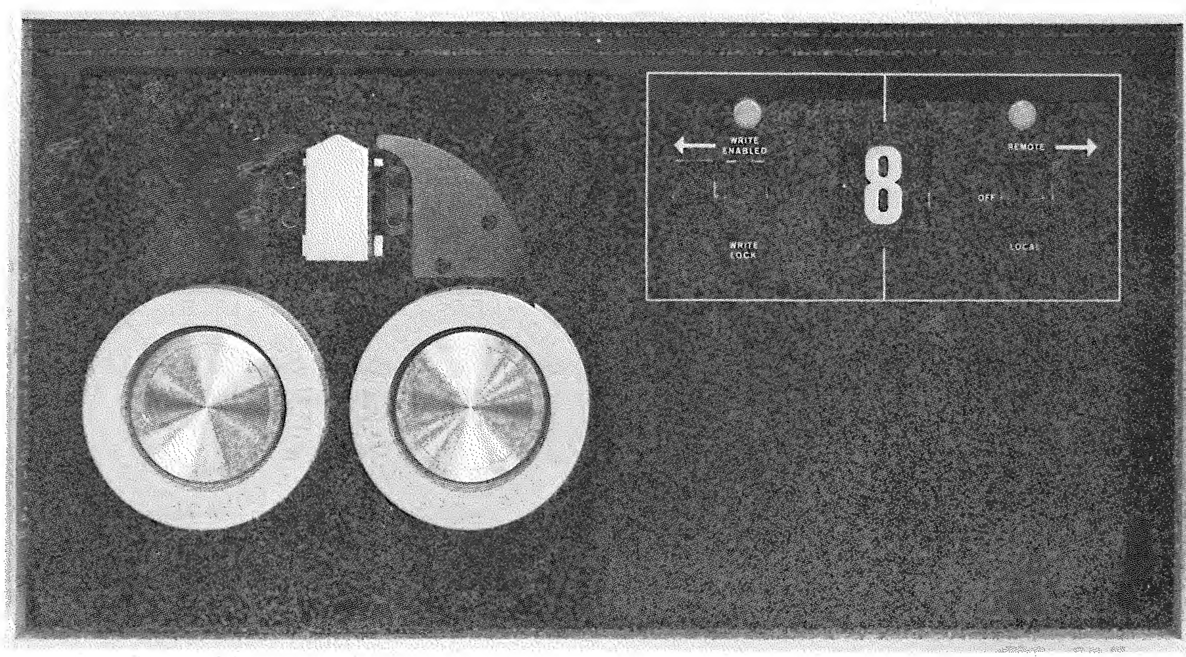


Figure 2-5. TU55 Tape Transport Control Panel

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 writing. If TC12 control system is commanded to write on tape while the WRITE LOCK is set, the control signals a TAPE NOT OK.
WRITE ENABLED	Lights when WRITE ENABLE/WRITE LOCK switch is in the WRITE ENABLE position.
Unit Selector 0 1 2 3 4 5 6 7	When this selector is dialed to one of the numerals and the REMOTE/OFF/LOCAL switch is set to REMOTE, the central processor may gain access to the unit.
OFF LINE	When the selector is dialed to OFF LINE, the transport cannot be selected by the tape control. NOTE Some earlier units were equipped with a unit selector position 8, which is the same as unit 0.
REMOTE/OFF/LOCAL Switch	
REMOTE	Permits tape processor to control the transport.
OFF	Removes power from reel motors and releases the brakes. This permits the operator to change the tape.

Table 2-12. TU55 Tape Transport Controls and Indicators (cont)

Control or Indicator	Function
LOCAL	Permits the forward and reverse tape-motion switches to provide tape motion in direction of the arrows. The transport cannot be selected.
REMOTE Indicator	Lights only when transport is selected by the tape processor.
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 simultaneously, reverse motion takes place.

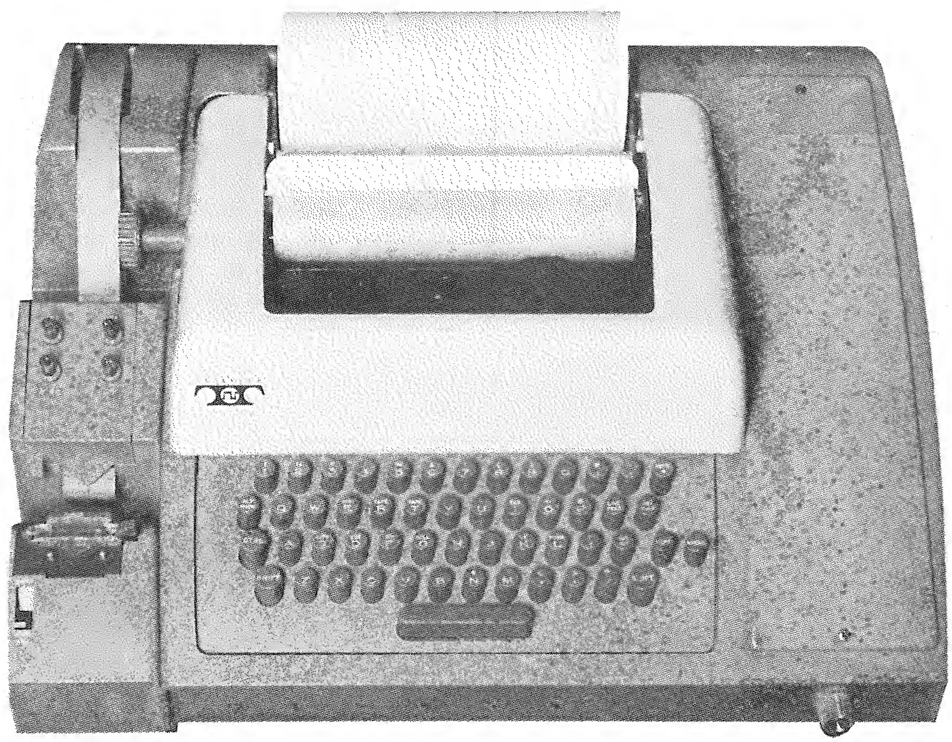


Figure 2-6. Teletype Model 33 ASR

Table 2-13. Teletype Model 33 ASR Controls

Control	Function
Punch Controls	
REL Pushbutton	Disengages the tape in the punch to allow removal or loading.
B SP Pushbutton	Backspaces the tape in the punch one space, allowing manual correction or rubout of the character just punched.
ON and OFF Pushbuttons	ON engages the punch for operation under local or program control. OFF removes the punch from control.
Reader Control	
START/STOP/FREE	In the FREE (lowest) position, the tape feed wheel is disengaged, and tape can be loaded or unloaded. In the STOP (center) position, the wheel is engaged but the reader cannot be operated. In the START (highest) position, the reader can be operated under local or program control.
LINE/OFF/LOCAL Switch	In the LINE position, the Teletype is energized and connected as an I/O device to the computer. In the OFF position, the Teletype is not energized. In the LOCAL position, the Teletype is energized for off-line operation, and signal connections to the processor are disconnected. Both LINE and LOCAL use of the Teletype require that the computer OFF/ON/LOCK switch is ON.

CHAPTER 3
LINC MODE PROGRAMMING

3.1 ORGANIZATION OF MEMORY

3.1.1 General

The LINC mode instruction set deals with two 1024_{10}-word Memory Fields. The INSTRUCTION FIELD is the 1024_{10}-word section of memory from which programs are executed, and in which data may be directly or indirectly accessed. The DATA FIELD is a second 1024_{10}-word section of memory to which the LINC instruction set allows only indirect reference for data manipulation and storage. The physical locations of the Data Field and Instruction Field within the maximum $32{,}768_{10}$-word memory are specified by the contents of the 5-bit Data Field Register and the 5-bit Instruction Field Register. These are set and modified under program control or from the console; they need not be adjacent, or in any particular order, and can even be identical. With respect to a LINC program, addresses within a Field remain constant, regardless of the actual location of the Field. Addresses within the Instruction Field are 0000 through 1777_8; addresses within the Data Field are 2000_8 through 3777_8. Thus, no matter where they are assigned, the two fields may be considered logically contiguous.

The PDP-8 instruction set (described in Chapter 4) divides memory into 4096_{10}-word Fields, which are specified by the most significant 3 bits of the Instruction Field and Data Field Registers. Therefore, the term Field designates a 1024_{10}-word segment of memory in LINC mode, and a 4096_{10}-word segment of memory in 8 mode. In Figure 3-1, the division of the first 4096_{10} words of memory is shown, assuming LINC mode, INST FIELD = 01, DATA FIELD = 03.

ABSOLUTE ADDRESS (OCTAL)	0000-1777	2000-3777	4000-5777	6000-7777
FIELD		INST		DATA
LINC ADDRESS (OCTAL)		0000-1777		2000-3777

12-0178

Figure 3-1. Assignment of LINC Addresses within Memory

3.1.2 **Program Counter**

The Program Counter acts as a 10-bit counter in the 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 is incremented to 4000, and 7777 to 6000. This 10-bit indexing is very common in LINC mode operations.

3.1.3 **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 the three bits of the corresponding 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.

3.1.3.1 **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 coordinate during execution of DSC.
0001-0017	As β-registers, used by 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 is assigned to the lowest segment of memory (that is, when C(IF) = 00), the following registers are also reserved:

Field Address	Use
0000 0001	PDP-8 Interrupt locations (Paragraph 4.4)
0040	Holds return address after a program interrupt during LINC mode operation.
0041	Location to which control is transferred after a program interrupt during LINC mode operation.
0140	Holds return address after an instruction trap.
0141	Location to which control is transferred after an instruction trap.

3.1.3.1 **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.

3.2 MEMORY ADDRESSING METHODS

3.2.1 General

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. 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 address* of the desired data. The LINC instruction set provides for both types of addressing.

3.2.2 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 Paragraph 3.3. 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.

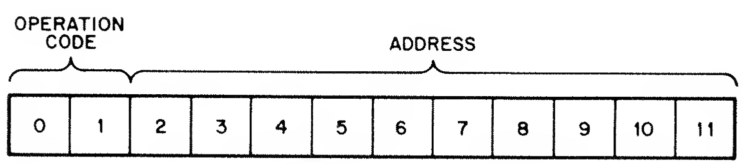


Figure 3-2. Direct Address Instruction Format

3.2.3 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 addresses 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.

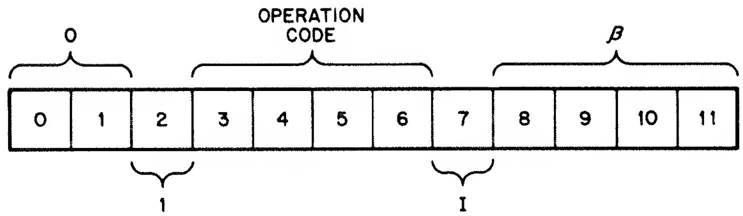


Figure 3-3. β-Class Instruction Format

3.2.3.1 **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	C(R)=2345
	I=0, β=00	
	Destination of C(AC):	Location 2345 (Location 345 in the Data Field)

(2) STA I 0	Octal code: 1060 I=1, β=00 Destination of C(AC):	 Location R
(3) STA 12	Octal code: 1025 I=0, β=12 Destination of C(AC):	C(0012)=3456 Location 3456 (Location 1456 in the Data Field)
(4) STA I 12	Octal code: 1072 I=1, β=12 Destination of C(AC):	C(0012)=3456 Location 3457 (Location 1457 in the Data Field) (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.

ADDRESS	OCTAL		CONTENTS	REMARKS
0007	1500		1500	/REPLACED BY 1501 AFTER INDEXING
			*1000	
1000	1100		ADA	/INDIRECT THROUGH 1001, ADD N1 to C(AC)
1001	1477		1477	/ADDRESS OF N1
1002	1120		ADA I	/DIRECT TO 1003; ADDS N2 TO C(AC)
1003	3211	N2,	3211	
1004	1107		ADA 7	/INDIRECT THROUGH 7 TO 1500, ADDS N3
1005	1127		ADA I 7	/INDIRECT THROUGH 7, INDEXED, ADDS N4
			*1477	
1477	1234	N1,	1234	
1500	1235	N3,	1235	
1501	4321	N4,	4321	

3.2.3.2 **β-Register Indexing** – When the β-indexing scheme is used (I = 1, $\beta \neq 00$), 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.

3.2.4 Addressing: α-Class

Three LINC mode instructions – SET, XSK, and DIS – have specialized memory reference schemes. Although each of them accesses 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 3.3. DIS is described in Section 3.4.

3.3 LINC MODE INSTRUCTIONS
(Complete list is provided in Appendix A.)

3.3.1 Instruction Formats

There are three basic LINC mode instruction formats.

3.3.1.1 **Direct Address (See Figure 3-2)** – This class consists of the three instructions ADD, STC, and JMP.

3.3.1.2 **Indirect Address, β-class (See Figure 3-4)** – This class consists of 15 β-class instructions, with operation codes between 1000 and ~~1740~~.
1737

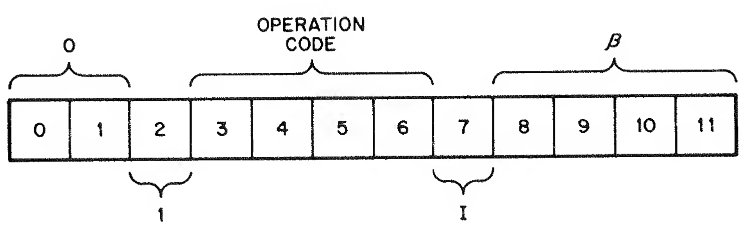


Figure 3-4. β-Class Format

3.3.1.3 **α-class and Others (See Figure 3-5)** – There are 16 basic instructions in this group, and they have operation codes between 0000 and 0777. Each of these instructions has up to 32 variants, depending on the contents of bits 7-11.

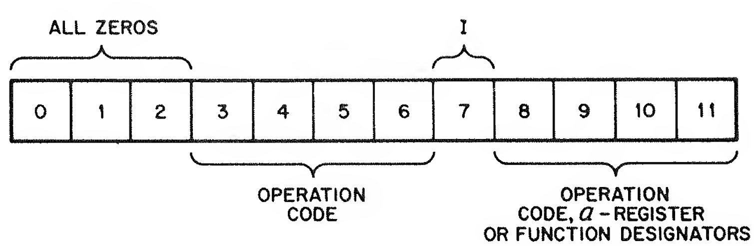


Figure 3-5. α-Class and Non-Memory Reference Format

3.3.2 **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
α-class Operations	SET, XSK
Program Control	JMP
Shift and Rotate	ROL, ROR, SCR
Skips	APO, AZE, LZE, QLZ, FLO, SNS, SXL, KST, SKP, STD, TWC
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/IOTs, DJR
Special Functions	ESF, SFA

Instructions related to the Display, Data Terminal, and LINCtape are described in Sections 3.4, 3.5, and 3.6.

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.)

3.3.3 **Full-Word Instructions**

3.3.3.1 **Full-Word Data Transfers** – These three instructions move complete 12-bit words between the Accumulator and Memory.

STC Store and Clear (Direct Address)

Form:	STC Y
Octal code:	4000+ Y
Execution time:	3.2 μs
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 (β-Class)

Form:	LDA I β
Octal code:	1000 + 20I + β
Execution time:	4.8 μs; 3.2μs. 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 contents of the AC are lost; the contents of Y are unchanged.

STA   Store Accumulator (β-Class)

Form:	STA I β
Octal code:	1040 + 20I + β
Execution time:	4.8 μs; 3.2 μs. 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 contents of Y are lost; the contents of the AC are not changed.

3.3.3.2 **Full-Word Arithmetic** – The instructions ADD, ADA, and ADM use one's complement arithmetic. If, as a 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 one'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 (-2073)
                1 0341          end-around carry; 1 added to sum.
                     1
                  0342
```

In either case, the Link is not affected.

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

3.3.3.2.1 **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 FLOW flip-flop is set to 1. If no overflow occurs, the FLOW flip-flop 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.

3.3.3.2.2 **Instructions**

ADD   Add to Accumulator (Direct Address)

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

ADM Add to Memory (β-Class)

Form:	ADM I β
Octal code:	1140 + 20I + β
Execution time:	4.8 μs; 3.2 μs. 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 one's complement addition; the sum is left in both the AC and Y. The previous contents of both registers are lost; the Link is not changed.

LAM Link Add to Memory (β-Class)

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

NOTE

This description 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 two's complement addition; the sum is left 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 two'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 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	
		+6517	
		1 2463	carry; Link is set to 1.

Results:

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

MUL Multiply (β-Class)

Form:	MUL I β
Octal code:	1240 + 20I + β
Execution time:	9.6 μs; 8 μs. 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_0.

The multiplier and multiplicand are treated as 12-bit one'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 FLOW 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 $C(\beta_0) = 1$, 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 AC_{1-11} and the absolute value of the low order 11 bits of the product appears in MQ_{0-10}. 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 appears 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 Paragraph 3.3.11). pg 3-21

Examples: (all octal form)

(1) Integers

(a.)	0432	C(AC)			
	x0006	C(Y)			
	00003234	product	C(AC)=3234		C(MQ)=6470
(b.)	2764				
	x0153				
	00476374		C(AC)=2374		C(MQ)=4770
(c.)	2764				
	x7624	(= −153)			
	77301403	(= −00476374)	C(AC)=5403	(= −3154)	C(MQ)=4770

(2) Fractions

(a.)	0432	C(AC) = .1064		
	x0006	C(Y) = .0814		
	00003234	product = .0001516	C(AC)=0000	C(MQ)=6470
(b.)	2764	C(AC) = .575		
	x0153	C(Y) = .0326		
	00476374	product = .011747	C(AC)=0117	C(MQ)=4770
(c.)	2764	C(AC)		
	x7624	C(Y) (-0153)		
	77301403	(-00476374)	C(AC)=7660	C(MQ)=4770

3.3.4 **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 (β-Class)

Form: BCL I β
Octal code: 1540 + 20I + β
Execution time: 4.8 μs; 3.2 μs. 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.

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

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

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

BSE Bit Set (β-Class)

Form: BSE I β
Octal code: 1600 + 20I + β
Execution time: 4.8 μs; 3.2 μs. 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 as follows:

$$\begin{array}{cc|cc} & & \multicolumn{2}{c}{C(AC_j)} \\ & & 0 & 1 \\ \hline & 0 & 0 & 1 \\ C(Y_j) & & & \\ & 1 & 1 & 1 \end{array}$$

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 (β-Class)

Form: BCO I β
Octal code: 1640 + 20I + β
Execution time: 4.8 μs; 3.2 μs. 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 as follows:

$$\begin{array}{cc|cc} & & \multicolumn{2}{c}{C(AC_j)} \\ & & 0 & 1 \\ \hline & 0 & 0 & 1 \\ C(Y_j) & & & \\ & 1 & 1 & 0 \end{array}$$

The Boolean statement of this relation is AC $\veebar$ Y.

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

3.3.5 Full-Word Comparison

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

SAE Skip If Accumulator Equal To Operand (β-Class)

Form: SAE I β
Octal code: 1440 + 20I + β
Execution time: 4.8 μs; 3.2 μs. 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 (β-Class)

Form:	SRO I β
Octal code:	1500 + 20I + β
Execution time:	4.8 μs; 3.2 μs. when I = 1 and β = 00
Operation:	Bit 11 of the operand is tested. If it is a 0, skip is set and the operand is rotated right one place. If bit 11 is a 1, no skip takes place, but a rotate right one place still occurs.

Example:

ADDRESS	OCTAL		CONTENTS	REMARKS
			*20	
0020	1520	P,	SRO I 0	/THE OPERAND IS IN REG 21
0021	3725	P1,	3725	/OPERAND BIT 11=1, SO NO SKIP
0022	0016	P2,	NOP	/PROGRAM CONTINUES FROM HERE
0023	0016	P3,	NOP	

After the test is performed, the contents of P1 are rotated right one place; the result, which is retained in P1, is 5752. If the instruction in register P were then to be executed again, the skip would occur, because the new contents of bit 11 of P1 equal 0. The program would then proceed from register P3, skipping the instruction in P2.

3.3.6 Half-Word Operations

3.3.6.1 **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 is transferred or compared between the designated half of the operand and the right half of the AC.

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.

a. 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, and the left half of the AC is cleared.

b. 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, and the left half of the AC is cleared.

c. 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, and the left half of the AC is cleared.

d. 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 then 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 is then 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(β) 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, 0370 1/2, 0371, 0371 1/2, 0372, etc. (Or, in octal, 0370.0, 0370.4, 0371.0, 0371.4, 0372.0, etc.)

LDH Load Half

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

STH Store Half

Form:	STH I β
Octal code:	1340 + 20I + β
Execution time:	4.8 μs; 3.2 μs 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 + β
Execution time:	4.8 μs; 3.2 μs 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.

3.3.7 α-Class Operations

Each of these instructions uses the registers 0000-0017 in a unique way. A third α-class instruction, DIS, is described in Paragraph 3.4, CRT Display.

SET Set α-Register

Form:	SET I α
Octal code:	0040 + 20I + α
Execution time:	6.4 μs; 4.8 μs 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 μs

Operation: If I = 1, the contents of the designated α-register are incremented by 1, using 10-bit two's complement addition as in β-class indexing. If I = 0, α is 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.

3.3.8 **Program Control**

JMP Jump (Direct address)

Form: JMP Y
Octal code: 6000 + Y
Execution time: 3.2 μs when Y $\neq$ 0000; 1.6 μs when Y = 0000
Operation: The quantity Y is placed in PC_{2-11}, and the next instruction is taken from register Y. The program proceeds from that point.

If Y $\neq$ 0000, 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. 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. (See the DJR instruction, Paragraph 3.3.15.4, for use of JMP in conjunction with the DJR instruction.)

[illegible] LINC
routine using JMP

Example

ADDRESS	OCTAL		CONTENTS	REMARKS
			*0000	
0000	0000	0000,	0000	/WILL CONTAIN 6573 AFTER JMP 175 IS /EXECUTED
0001	0000			
			*0175	
0175	0016		NOP	/START OF LINEAR SUBROUTINE WHICH
0176	0000		...	/CONTAINS NO JMP INSTRUCTIONS
0177	0000		...	
			*0242	
0242	6000		JMP 0	/RETURN FROM SUBROUTINE
0243	0000			
0244	0000			
			*0572	
0572	6175		JMP 175	/JUMP TO SUBROUTINE AT REGISTER 0175
0573	0017		NOP	/SUBROUTINE RETURNS TO THIS LOCATION
0574	0000		.	
0575	0000		.	
			.	

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, 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 3.3.14. Memory Address Control), the first JMP Y ($Y \neq 0000$) 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.

3.3.9 **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 1-1/2 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 3-7 and 3-8. In all these operations, the Link is included when I = 1 and excluded when I = 0. Execution times depend on the number of positions shifted.

ROL Rotate Left

Form:	ROL I N
Octal code:	0240 + 20I + N, $0 \leq N \leq 17_8$
Execution time:	1.6 - 6.4 μs
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.

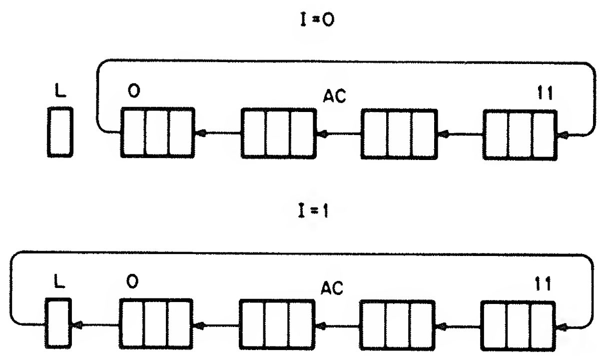


Figure 3.6. Rotate Left

ROR Rotate Right

Form: ROR I N
Octal code: 0300 + 20I + N, $0 \leqslant N \leqslant 17_8$
Execution time: 1.6 - 6.4 μs
Operation: The contents of the AC are shifted 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.

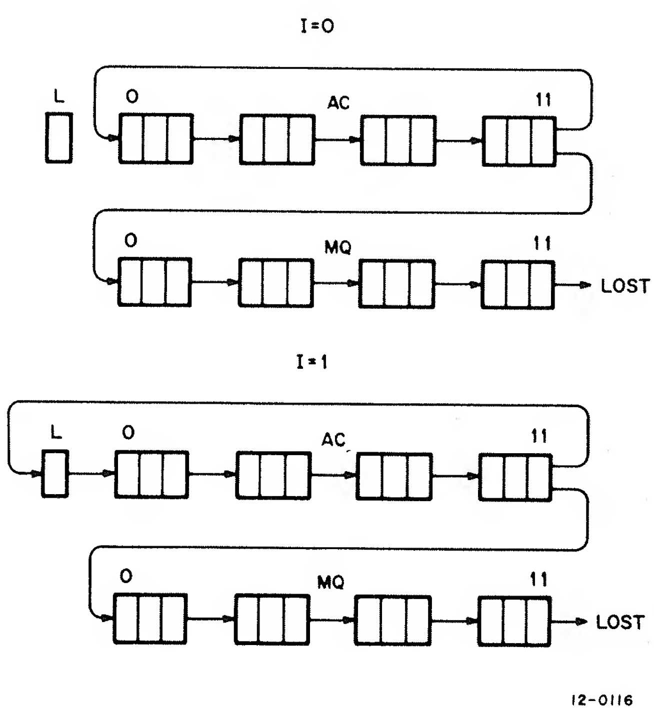


Figure 3-7. Rotate Right

SCR Scale Right

Form: SCR I N

Octal code: 0340 + 20I + N, $0 \leqslant N \leqslant 17_8$

Execution time: 1.6 - 6.4 μs

Operation: The contents of the AC are shifted right N places. The sign bit (contents of AC_0) is not changed, and is placed 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. The result of the operation:

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

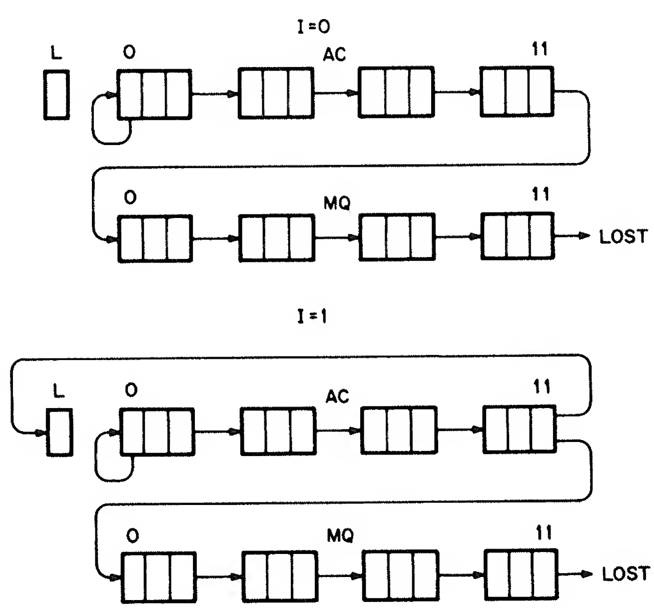


Figure 3-8. Scale Right

3.3.10 **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.

The skip instructions (with the exception of the unconditional skip instruction, SKP) have an associated "skip condition", such as a register being cleared or one of the external digital inputs ("external levels") being asserted. If the skip condition is met (e.g., the register is cleared or the external level is asserted) at the time the instruction is executed, a skip occurs. This means that the next instruction is taken from the second location following the skip instruction, rather than from the usual location. Normally, the skip instruction is followed by a single-word instruction that is skipped or executed according to whether or not the tested condition was met. No distinction is made between single- and double-word instructions; if the skip instruction were followed by a double-word instruction, control would be transferred to the second word of the instruction if a skip occurs. Generally skip instructions should not be followed by double-word instructions. In the skip instructions discussed here, (but not in the "addressable" skips: SHD, SAE, and SRO), the i bit inverts the sense of the skip. That is, if the i bit is zero, the instruction skips only if the skip condition is met. If the i bit is a one, the skip occurs only if the condition is not met.

APO Accumulator Positive

Form:	APO I
Octal code:	0451 + 20I
Execution time:	1.6 μs
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 μs
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 μs
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 μs
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 μs
Condition:	The FLOW flip-flop is set to 1. When overflow occurs as the result of an addition (ADD, ADA, ADM, or LAM), the FLOW flip-flop is set to 1. If overflow does not occur as a result of the above instructions, the FLOW flip-flop is cleared.

SKP Skip Unconditionally

Form:	SKP I
Octal code:	0456 + 20I
Execution time:	1.6 μs
Condition:	The next instruction is skipped unconditionally. (In LINC-8 and early PDP-12 assembly programs SKP was defined as 466.)

The following skips test for various external input conditions.

IBZ LINCtape Inter-Block Zone

Form:	IBZ I
Octal code:	0453 + 20I
Execution time:	1.6 μs
Condition:	If the selected LINCtape unit is in one of the inter-block zones and the tape is moving, the next instruction is skipped. This instruction will sense an IBZ only if the tape is in motion (i.e., only after a tape instruction with "I = 1").

SNS Sense Switch

Form:	SNS I N
Octal code:	0440 + 20I + N, $0 \leqslant N \leqslant 5$
Execution time:	1.6 μs
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 μs FOUR
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. When nothing is connected to the External Level Lines, they are preloaded to +3V. 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 via the I/O Bus cables (see Chapter 5).

The three defined levels and their mnemonics are:

SXL	I	14	(STB)	8 tape block flag set
SXL	I	15	(KST)	Key Struck (See below)
SXL	I	16	(STD)	Tape Instruction Done (See Paragraph 3.6.9)
SXL	I	17	(TWC)	Tape Word Complete (See Paragraph 3.6.14)

KST Key Struck

Form: KST I
Octal code: 0415 + 20I
Execution time: 1.6 μs
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.)

3.3.11 Miscellaneous

These instructions perform various tasks. All are self-contained and require no memory references.

HLT Halt

Octal code: 0000
Execution time: 1.6 μs 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 μs
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 μs
Operation: The contents of the AC are complemented. Bits containing 0s are changed to contain 1s, and vice versa. No other registers are affected.

NOP No Operation

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

QAC Place MQ in AC

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

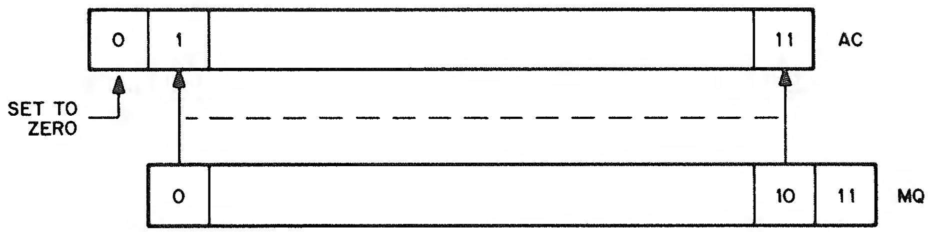


Figure 3-9. QAC Transfer Path

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

OCTAL CODE	INSTRUCTION	REMARKS
	*0020	
0005	QAC	/C(MQ0-10) PLACED IN AC1-11
0241	ROL I	/ROTATE C(AC) LEFT 1 PLACE, WITHOUT LINK.
0475	QLZ I	/SKIP IF C(MQ11) = 1
6026	JMP .+3	/C(MQ11) = 0. JUMP TO THIRD REGISTER BEYOND.
1620	BSE I	/C(MQ11) = 1. SET AC11 EQUAL TO 1.
0001	0001	/OPERAND TO SET AC11

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

3.3.12 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 μs
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 μs
Operation: The contents of the Right Switches Register are placed in the AC. The previous C(AC) are lost.

3.3.13 Mode Control

PDP Switch To The 8 Mode

Octal code: 0002
Execution time: 1.6 μs
Operation: Beginning with the next succeeding instruction, the central processor will operate in the 8 mode; all subsequent instructions are interpreted as PDP-8 operations. A similar instruction, LINC (6141), in the PDP-8 mode instruction set, causes a switch to LINC mode.

3.3.14 **Input/Output Bus**

In addition to the input and output devices controlled directly by LINC instructions (see Paragraphs 3.4, 3.5, and 3.6), the LINC mode program also has direct access to any device connected to the PDP-12 I/O Bus. By using the special two-word enabling instruction, IOB, any 8 mode IOT instructions can be included within a LINC program sequence.

IOB I/O Bus Enable

Form:	IOB (first word) IOT (second word)
Octal code:	0500 (first word)
Execution time:	5.9 μs
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 waits for the high-speed reader flag and then reads the character buffer

OCTAL CODE	INSTRUCTION	REMARKS
	....	
	* 0020	
0500	IOB	/ENABLE I/O BUS
6011	RSF	/SKIP IF HIGH-SPEED READER FLAG IS SET
6020	JMP .-2	/NOT SET. GO BACK TWO SPACES
0500	IOB	/FLAG IS SET. ENABLE THE BUS, AND...
6016	RRB	/READ THE CHARACTER
	....	

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

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

3.3.15 **Memory Address Control**

The two memory Fields 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, or in any particular order. 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.

3.3.15.1 **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.

3.3.15.2 **Save Field Register (SF) 10 Bits** – 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 may be resumed. The contents of the SF may be read into the AC. (See also the Program Interrupt discussion in Paragraph 3.3.15.4.) For historical reasons the SF is also called Interrupt Buffer in some places.

3.3.15.3 **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 μs
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 called subroutine is located in a memory field other than the current one. The subroutine may pick up the field information needed in obtaining arguments and generating subroutine returns by interrogating the Save Field Register.

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 is fetched and executed in the newly selected Instruction Field. This allows the Save Field Register to be used for cross field subroutine linkage in programming which uses the Program Interrupt. (See the DJR instruction, Paragraph 3.3.15.4, for an explanation of DJR/LIF/JMP combinations.)

Note: LIF is abrogated by PDP

Because LINK Instruction Trap (see Paragraph 3.3.17) also uses the Save Field Register for program linkage, an instruction which will be trapped must not be given between an LIF and the next JMP Y (Y $\neq$ 0000) if the cross field reference will ultimately need the Save Field Register linkage information.

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 μs

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 μs

Operation: The contents of the Instruction Field Register are ORed into bits AC_{6-10}. The remaining AC bits are unaffected, and the contents of the IF are unchanged.

NOTE

When executed in LINC mode, the three IOT instructions, RIF, RDF, and RIB, are the only cases where an IOT has a slightly different function than when executed in 8 mode. For these instructions, all five bits of the IF and/or DF are read into the AC when in LINC mode, while only the most significant three bits of each are used in 8 mode.

IOB
RDF Read Data Field

Octal code: 0500
6214
Execution time: 5.9 μs
Operation: The contents of the Data Field Register are ORed into bits AC_{6-10}. The remaining AC bits are unaffected, and the contents of the DF are not changed.

3.3.15.4 **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. With Interrupt enabled, the following sequence of events will occur when a flag is set:

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 (Save Field Register).

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, restore the Memory Field registers, re-enable the interrupt, and jump back to the interrupted program at the point where the Program Interrupt occurred.

Whenever a change of LINC Instruction Field occurs, the Program Interrupt is inhibited (between Steps 1 and 2 above) 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.

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

IOB
ION Interrupt On

Octal code: 0500
6001
Execution time: 5.9 μs
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. The 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 without losing its place.

IOB
IOF Interrupt Off

Octal code: 0500
6002
Execution time: 5.9 μs
Operation: The Interrupt is disabled. The facility is disabled immediately; subsequent requests 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 a Program Trap.

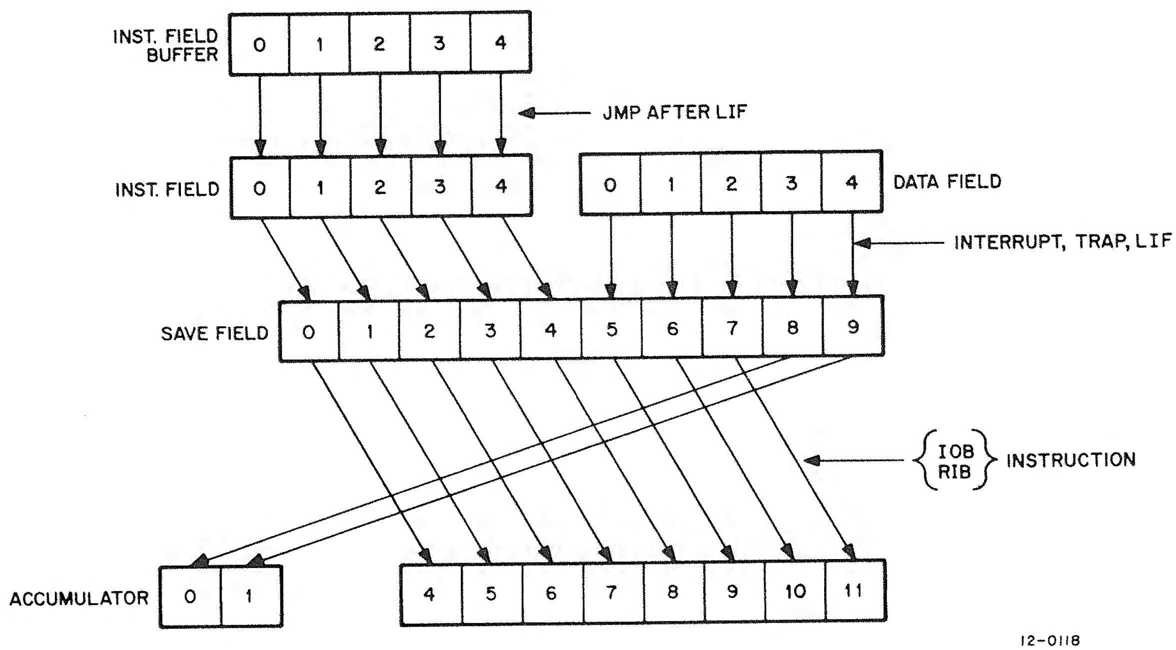


Figure 3-10. Data Path: IB, IF, DF, and AC

IOB
RIB Read Interrupt Buffer

Octal code: 0500
6234

Execution time: 5.9 μs

Operation: The contents of the Interrupt Buffer (Save Field Register) are ORed into bits AC_{0-1} and AC_{4-11}, as shown in Figure 3-10. AC_{2-3} 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. (If inhibit were not provided, a waiting interrupt request could destroy the contents of the Save Field Register. The first JMP instruction executed after a trap or change of Instruction Field reenables the Interrupt.)

IOB
RMF Restore Memory Fields

Octal code: 0500
6244

Execution time: 5.9 μs, including IOB

Operation: The contents of SF_{5-9} are placed in the Data Field Register, and the contents of SF_{0-4} 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.

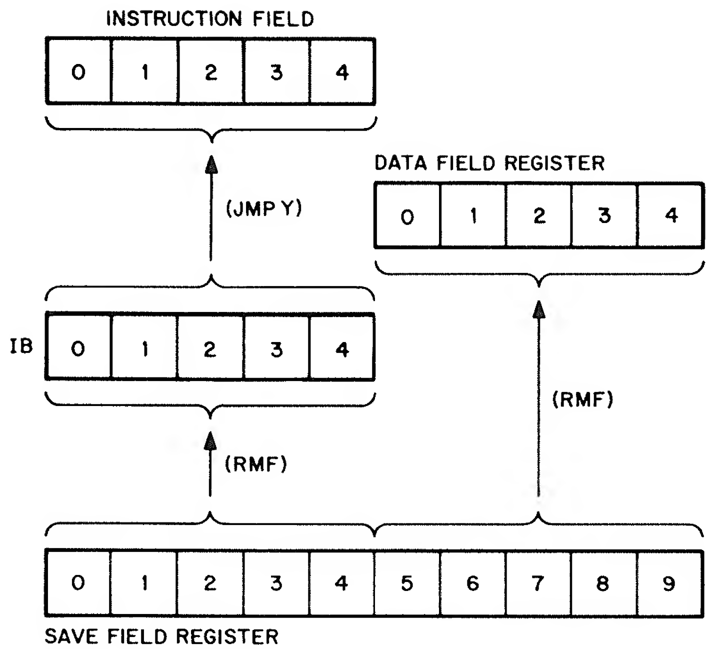


Figure 3-11. Data Path, RMF Instruction

DJR Disable JMP Return

Octal code:	0006
Execution time:	1.6 μs
Operation:	DJR sets a flip-flop, preventing contents of Location 0000 from being changed 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 instructions. This is useful because an interrupt can occur within a LINC mode subroutine that uses Location 0000 of its IF to retain the subroutine return; hence, it must not be destroyed.

NOTE

The DJR instruction causes an exception to the normal rule concerning JMP 0s. A JMP 0 following a DJR and LIF instruction causes a change of instruction fields, hence a jump to location 0 of the new field. This arrangement is necessary to facilitate return from an interrupt subroutine when the PC was pointing to location 0 of the main program prior to the interrupt. In the following example, if 0532 is replaced with 0, this feature of the DJR instruction is more clearly illustrated.

issued? used?

Because of this feature, extra care is required in the use of DJR instructions in subroutines. A DJR instruction cannot be indiscriminately sued when entering a subroutine if it is not known whether a JUMP≠0 will be used prior to the use of a JUMP 0 required for return to the main routine. Such indiscriminate use of a DJR instruction could result in a JMP to 0 of the new field rather than to the correct location which is accessed by going through the 0 location of the subroutine field.

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 sets up a return jump enabling the program to get back to the point of the break. Next, it identifies the cause of the request and services the condition. Finally, it restores the conditions prevailing at the time the interrupt occurred and returns to the main program. The following sequence shows how these tasks may be accomplished.

ADDRESS	OCTAL		CONTENTS	REMARKS
			*0040	
0040	0532		0532	/CONTENTS OF PC AT TIME OF INTERRUPT
0041	4000		STC ACSAV	/SAVE C(AC), THEN CLEAR AC
0042	2040		ADD 0040	/SAVED ADDRESS (0532) TO AC
0043	1620		BSE I	/MAKE JMP INSTRUCTION: C(AC)
				/V C(0044)
0044	6000		6000	/OCTAL CODE OF JMP INST.
0045	4071		STC RTN	/STORE JMP 532 AT END OF SERVICE
				/ROUTINE
0046	0000			/MAIN PART OF SERVICE ROUTINE, IF
				/NECESSARY
0047	0000			/OTHER ACTIVE REGISTERS (MQ,L,ETC)
0050	0000			/SHOULD BE SAVED ALSO
0051	0000			
0052	0016		NOP	
0053	0016		NOP	/THE REST OF THE ROUTINE
0054	0016		NOP	
0055	0016		NOP	
0056	0000			
0057	0000			
0060	0500		IOB	/EXIT SEQUENCE, ENABLE IOT TIMING
				/CHAIN
0061	6244		RMF	/. . .AND RESTORE MEMORY FIELDS,
				/(07 TO IB)
0062	4076		STC TEMP	/CLEAR AC WITHOUT DISTURBING MQ AND L
0063	2000		ADD ACSAV	/RESTORE ORIGINAL C(AC)
0064	0006		DJR	/SET PROCESSOR SO THAT NEXT JMP INST
0065	0000			/WILL NOT STORE IN LOCATION ZERO OF
0066	0000			/MEMORY BANK TO WHICH JMP AT "RTN"
				/WILL GO
0067	0500		IOB	
0070	6001		ION	/RE-ENABLE INTERRUPT
0071	6532	RTN,	JMP 0532	/JMP TO ORIGINAL FIELD
0072	0000			
0073	0000			
0074	0000			
0075	0000			
0076	0000	TEMP,	0000	
		RMF=6244		
		ION=6000		

3.3.16 **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 AC_{2-7}, as shown in Figure 3-12.

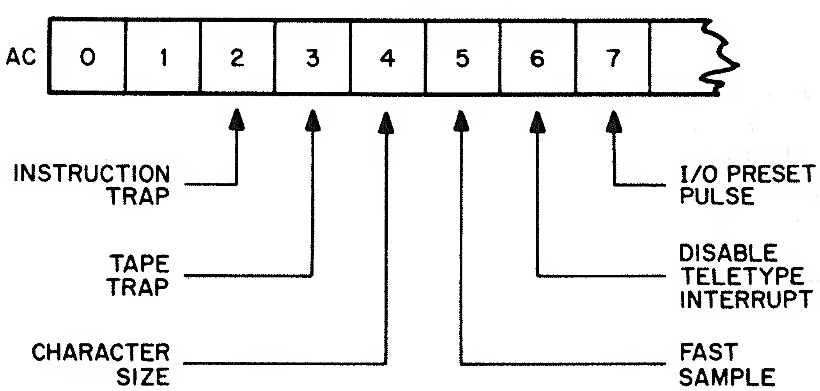


Figure 3-12. Special Functions

These functions have the following characteristics:

1. *Instruction Trap Enable* – Causes an immediate program interrupt to register 0141 when an undefined LINC instruction code is encountered.

The instruction trap is described in detail in Paragraph 3.3.17.

2. *Tape Trap* – When this function and Instruction Trap Enable are both set, a program interrupt to register 0141 will occur whenever a LINCtape instruction or one of the other trapped codes is encountered. The LINCtape instruction is not executed. Tape Trap is described in detail in Paragraph 3.6.13.

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 Paragraph 3.4.1.

4. *Fast Sample* – This function reverses the order of events of the SAM instruction; i.e., read the converter buffer and initiate a new conversion, then continue without pausing (see Paragraph 3.5.1).

5. *Disable Teletype Interrupt* – Interrupt requests from the ASR-33 Keyboard or Printer are inhibited. No program interrupt will occur when either TTY flag is set even if the Interrupt Facility is enabled (see Chapter 6).

6. *Generate I/O Preset* – If this bit is set when the enabling instruction (ESF) is executed, an I/O PRESET pulse is generated clearing all device flags, disabling the Interrupt, clearing the Tape Extended Operations Buffer, and generating the TAPE PRESET pulse. Other Special Functions are cleared, or in the case of CHARACTER SIZE, set to full size. 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 given. All the Special Functions except I/O PRESET are controlled by flip-flops set from the designated AC bits; the states of these flip-flops may be examined at any time.

FOR HOW LONG ?

ESF Enable Special Functions

Octal code: 0004
Execution time: 1.6 μs
Operation: The contents of AC_{2-6} are placed in their respective 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. All Special Function bits are cleared (except ESF 04, Full Size Character, which is set to a 1) by I/O PRESET, either from the console or program control.

SFA Place Special Function Flip-Flops in AC

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

3.3.17 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 $C(AC_2) = 1$), any 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 placed in the Save Field Register. The subroutine beginning at 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. (Also, see Paragraphs 3.3.15.3 and 3.3.15.4 for interaction with Program Interrupt.

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 LINC codes 1700-1737 are considered as β-class instructions. Therefore, when either a 1700 or 1720 instruction is encountered, the address contained in location 140 is the address of the trapped instruction incremented twice (trapped instruction address +2). All other undefined codes will cause location 140 to contain the address of the trapped instruction incremented once (trapped instruction + 1).

Probably the most common use of the Instruction Trap is in the execution of programs written for the LINC-8. A LINC-8 Trap Simulator is provided in the basic PDP-12 software package. Close study of this program will be most helpful for the programmer wishing to use the Trap facility. The Trap facility is further useful for developing device-independent software.

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

3.3.17.2 **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 an interrupt request, which also causes the contents of the IF and DF to be placed in the SF (also see Paragraphs 3.3.15.3 and 3.3.15.4).

3.4 CRT DISPLAY, TYPE VR12

The 6.5 inch x 9 inch rectangular screen of the PDP-12 CRT 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 inches; the vertical distance is 0.0127 inches. 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.

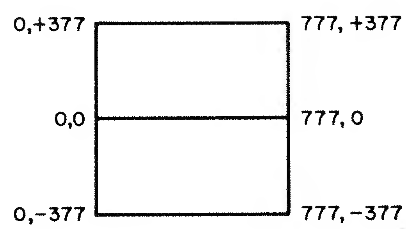


Figure 3-13. CRT Grid

The display system is fully buffered. Coordinates 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 multiplexed intensification channels can be specified. A switch on the VR12 front panel allows either or both channels to be displayed.

Below the channel selector is a variable knob which allows the user to change the intensity of the displayed points. A level control located within the VR12 presets the maximum brightness level, preventing spot burns.

A 24-contact connector on the Data Terminal Panel allows the user to connect an auxiliary scope (VR-12A CRT Display, Tektronix 561, or similar unit) for remote display of the same information sent to the main screen. The channel selectors can be independently set so that each scope displays one of the channels, thus allowing independent simultaneous displays on two scopes.

A complete set of connection points for the VC12/VR12 display system is shown in Appendix G.

The output drive capability of the D-A converters is 0v to -5.85v 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 3.0%. 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 0v D-A point is equivalent to the lower left hand corner of the display screen.

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.

3.4.1 **Point Displays**

DIS Display (α-Class)

Form:	DIS I α
Octal code:	$0140 + \alpha, \quad 0 \geqslant \alpha \geqslant 17_8$
Execution time:	3.2 μs; 23 μs for completion of display

A single point on the screen is intensified. The vertical coordinate is specified by AC_{3-11}; the horizontal coordinate by bits 3-11 of the designated α-register. If bit 0 of the α-register is set to 0, the point will be displayed on Channel ~~0~~ 1; if $C(\alpha_0) = 1$, the point will be displayed on Channel ~~1.~~ 2

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

3.4.2 **Character Displays**

DSC Display Character (β-Class)

$\beta = 0$
or
$2 \leq \beta \leq 17_8$

Form:	DSC I β
Octal code:	$1740 + 20I + \beta$, ~~$\beta \leqslant 2 \leqslant 17_8$~~
Execution time:	4.8 μs when I = 1, β = 00; 6.4 μs when I = 0 or β ≠ 00 Control completion time 20-56 μs.
Operation:	A vertical 2 x 6 array of points is displayed according to a pattern word stored in register Y (Y is defined as with other β-class instructions). 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 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 coordinate is held in the Accumulator. The horizontal coordinate 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.

AC is changed

(2) The contents of AC_{3-6} are placed in the display control Y-buffer; $\boxed{AC_{7-11} \text{ are set to } 30_8}$.

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

The foregoing operations take 4.8 or 6.4 μs 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 in the order shown in Figure 3-14. The time required to scan and display the points varies according to the number of points to be intensified. Reaching the first point requires 20 μs, then 1 μs for each point to be left dark and 3 μs for each point to be displayed. This action continues until all points are intensified.

Because of the manner in which the Y-coordinate is used, full-size character arrays may start only at coordinates which are multiples of 40_8; e.g., 000, 040, 100, -100. Since the array itself is 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.

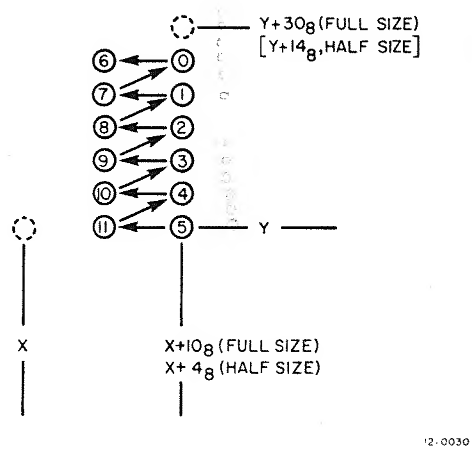


Figure 3-14. Display Pattern for DSC

3.4.3 **Half-Size Characters**

If the programmer clears the CHARACTER SIZE bit of the Special Function Register [ESF with $C(AC_4) = 0$], all increments are by two grid positions, rather than four. AC_{3-7} provides the initial Y coordinate; after the two coordinates have been transferred to the display control's buffers, the contents of AC_{8-11} will be 14_8, and the X coordinate 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. I/O PRESET sets this bit of the Special Function Register to a 1 (full-size characters).

3.4.4 **Character Set**

Any character that can be represented on a 4 x 6 grid (24 points) 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. Nondisplayed characters have patterns of all zeros. The table entries, each consisting of two words, are arranged in order of ASCII codes.

Table 3-1. ASR-33 Character Set Display Pattern

External ASCII	Internal Code	Character	Pattern Words	External ASCII	Internal Code	Character	Pattern Words
245	45	%	3114 0643	274	74	<	0412 2100
246	46	&	5166 0526	275	75	=	1212 1212
211	47	TAB	0000 0000	276	76	>	0021 1204
250	50	(	3600 0041	277 300	77 00	?	4020 2055
251	51	)	4100 0036	301	01	A	4477 7744
252	52	*	2050 0050	302	02	B	5177 2651
253	53	+	0404 0437	303	03	C	4136 2241
254	54	,	0500 0006	304	04	D	4177 3641
255	55	–	0404 0404	305	05	E	4577 4145
256	56	.	0001 0000	306	06	F	4477 4044
257	57	/	0601 4030	307	07	G	4136 2645
260	60	0	4136 3641	310	10	H	1077 7710
261	61	1	2101 0177	311	11	I	7741 0041
262	62	2	4523 2151	312	12	J	4142 4076
263	63	3	4122 2651	313	13	K	1077 4324
264	64	4	2414 0477	314	14	L	0177 0301
265	65	5	5172 0651	315	15	M	3077 7730
266	66	6	1506 4225	316	16	N	3077 7706
267	67	7	4443 6050	317	17	O	4177 7741
270	70	8	5126 2651	320	20	P	4477 3044
271	71	9	5122 3651	321	21	Q	4276 0376
272	72	:	2200 0000	322	22	R	4477 3146
273	73	;	4601 0000	323	23	S	5121 4651

Table 3-1. ASR-33 Character Set Display Pattern (cont)

External ASCII	Internal Code	Character	Pattern Words	External ASCII	Internal Code	Character	Pattern Words
324	24	T	4040	335	35	]	0000
			4077				7741
325	25	U	0177	336	36	↑	0000
			7701				0000
326	26	V	0176	212	37	LINE	0000 2504
			7402			FEED	0000 0416
327	27	W	0677				
			7701	240	40	SPACE	0000
330	30	X	1463				0000
			6314	241	41	!	7500
331	31	Y	0770				0000
			7007	242	42	"	7000
332	32	Z	4543				0070
			6151	215	43	RETURN	0000 0307
333	33	[	4177				0000 3045
			0000	244	44	$	4731
334	34	\	0204				4275
			1020				

3.5 DATA TERMINAL

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 Connector	24-pin connector for an auxiliary CRT

3.5.1 Analog Inputs

The AD12 Analog-Digital Converter and Multiplexer consists of 16 input channels, a Sample and Hold, a multiplexer, and a 10-bit A-D converter. Eight of the channels are for external inputs via phone jacks. These feed through preamplifiers to the multiplexer. The acceptable voltage range of these inputs is ± 1v with a sensitivity of approximately 2 mv/count.

The other eight channels are controlled by continuously variable knobs mounted on the Data Terminal. The knobs give ten turns stop-to-stop however, 7 turns provide the full 10-bit range to the converter (1-1/2 to 2 turns from each extreme is beyond the A-D range of -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 μs (Normal mode); 1.6 μs (Fast sample mode)

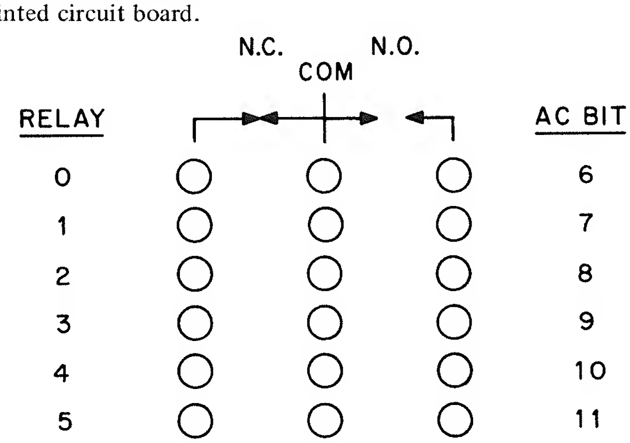
Operation: Input channel N is selected. In normal mode, the voltage level present at the input is sampled and converted to a 10-bit number (including sign), which is assembled in the converter buffer. When the conversion is complete, the contents of the buffer are transferred into AC_{3-11}. The sign is placed in AC_{0-2}. (ALL THREE)

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, unless the KW12-A Real Time Interface is selected to mode 4, 5, 6, or 7; in this case only, the KW12-A may initiate the A-D conversion. 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.

move comma

3.5.2 Relays

Six DPDT relays mounted on the Data Terminal 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. One set of form C contacts for each of the relays is available at the binding posts on the Analog Input Panel as indicated in Figure 3-15. A second set of contacts is brought out to split lugs on the relay printed circuit board.



12-0209

Figure 3-15. Relay Terminals and Corresponding AC Bits

ATR AC to Relays

Octal code: 0014
Execution time: 1.6 μs
Operation: The contents of AC_{6-11} are transferred to the Relay Buffer; the state of each relay is determined as follows: AC is unchanged.

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 μs
Operation: The contents of the Relay Buffer are transferred into AC_{6-11}. If the relay is off, the corresponding bit will be 0; if it is on, the bit will be 1. Bits AC_{0-5} are not changed.

3.6 LINCTAPE TYPE TC12

The basic LINCtape system consists of either two DECtape Transports Type TU55 or one DECtape Transport Type TU56 controlled by a fully 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.

3.6.1 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 reel of Standard format LINCtape has a capacity of 512 blocks, for a total of $131{,}072_{10}$ words of data. (In other formats, this capacity may be extended to $225{,}000_{10}$ words.)

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.

Figure 3-16B represents a typical block of 256 data words preceded and followed by control and identifying information. The serial bit sequence on the Mark Track, is decoded so that the control can determine whether the adjacent data bits form true data, checksum words, guard words, etc. The symbols BM, CM, GM, etc. (defined below), refer to these Mark Track bit patterns.

93 ips 120 μsec/word
360 bpi 32.16 msec/block
33.3K bps

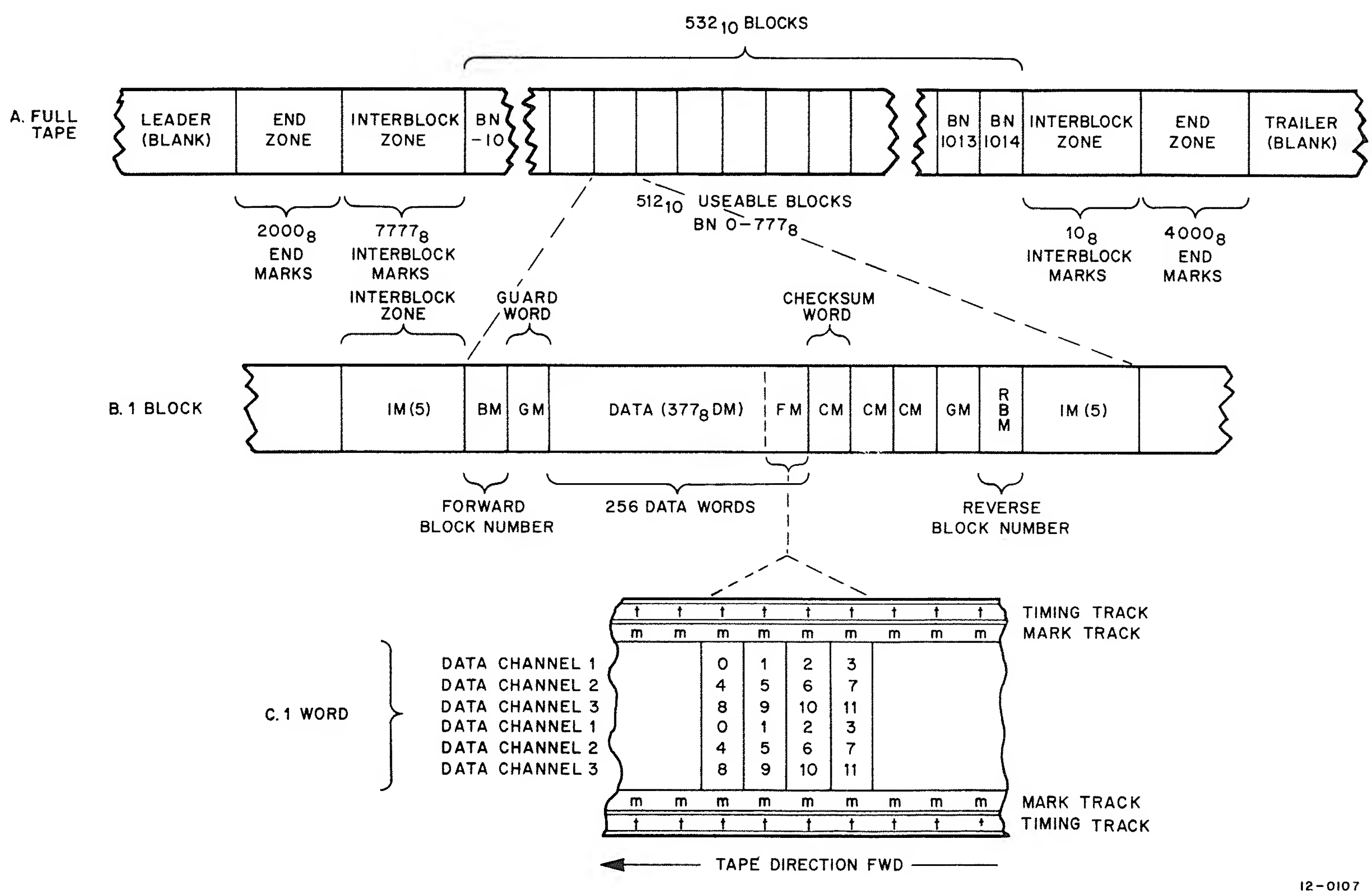


Figure 3-16. Standard LINCtape Format

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 (*BM*)	This identifies the block. On a standard LINCtape, block numbers are sequential, from 0000 through 0777_8.
Guard Word (*GM*)	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 (*DM,FM*)	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 tape processor to write the checksum in the next word position.
Checksum (*CM*)	This is the two'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 transfer.
Check Words (*CM*)	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 (*RBM*)	This is the Block Number that identifies the block when tape is being searched in the reverse direction.

Subprocessor

The LINCtape processor controls all information transfers 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 been initiated, 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.

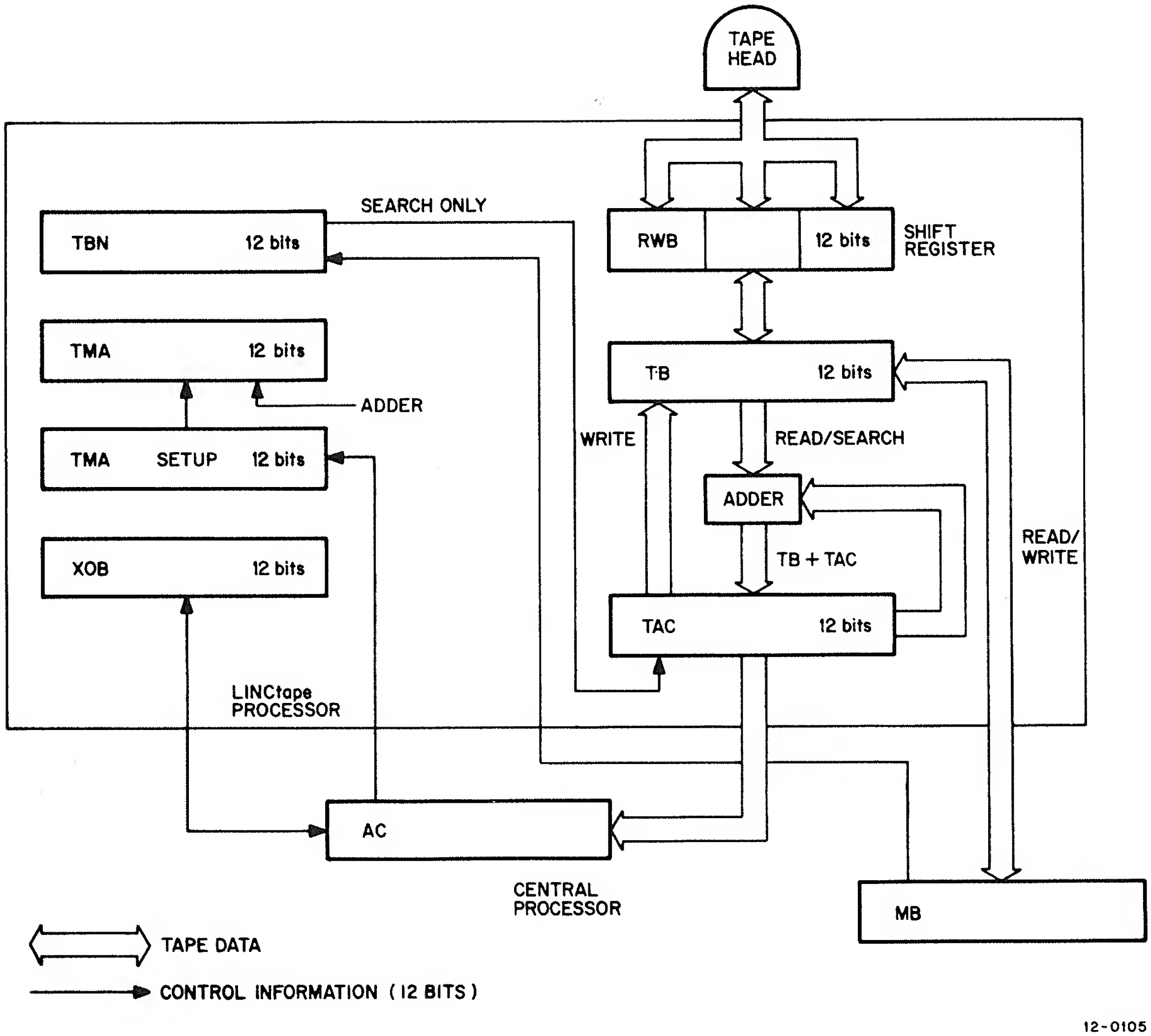


Figure 3-17. LINCtape Processor Information Paths

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. When 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. When reading, this sum is added to the Checksum read from tape, to determine whether the transfer was completed accurately. When 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 number of the first block to be accessed.

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 beginning of a tape instruction; in Standard mode, the MBLK and TBLK information in the second tape instruction word are used to determine the $\boxed{\text{initial}}$ contents of TMA. The TMA is incremented by 1 for each data word transferred.
In other words, LDF can be changed without altering TMA after the transfer is begun?

TMA Setup Register, 12 Bits – In Extended Address mode, the 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, the no-pause condition, hold motion, and extended units.

3.6.2 **Programming**

The tape transfer operations are the same for both Standard and Extended Operation modes. Data may be read or written in single blocks or groups of contiguous blocks (in the extended address mode, only single blocks are transferred), 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 tape 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.

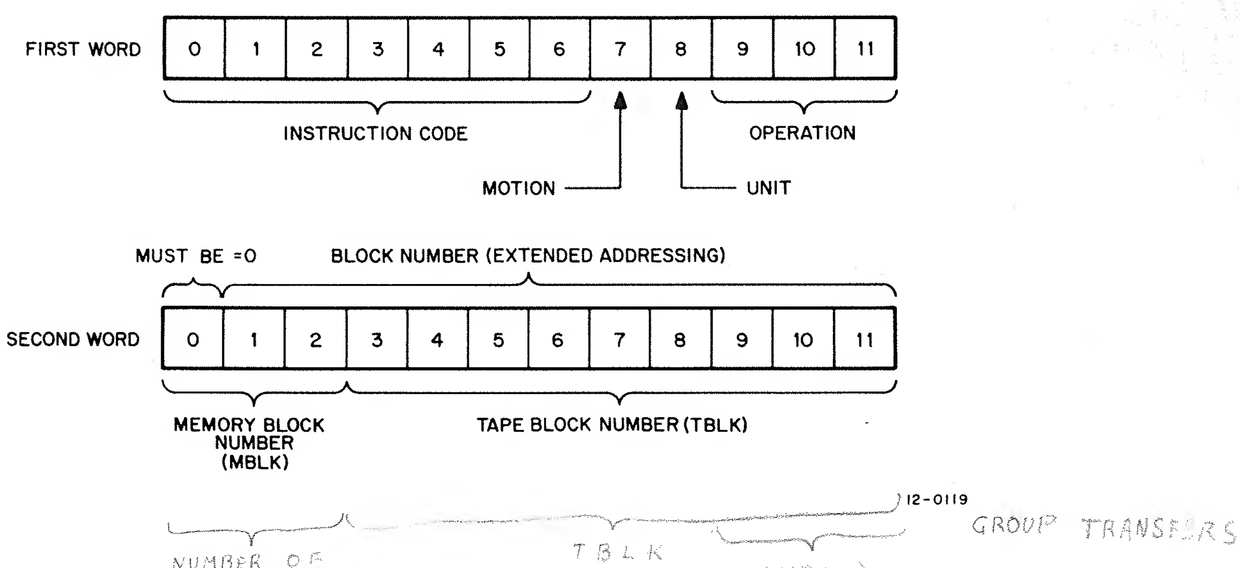


Figure 3-18. LINCtape Instruction Format

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 0 is selected; if this bit is a 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 Paragraph 3.6.3 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 blocks 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 LINC 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
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 in Paragraph 3.6.4.) The two non-transfer instructions, MTB and CHK, use the second word in still different ways.

3.6.3 **Tape Motion**

Tape is read or written in the forward direction. It may be searched in either the forward or backward direction. 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 a 1, the tape is left moving in the direction in which 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 a 0, the tape processor enters the Turnaround State at the completion of the operation, which is the end of the Checkword State. When a block mark is encountered, the tape stops, leaving the Turnaround State and entering the Idle State. If a subsequent tape instruction occurs while the tape processor is in the Turnaround State, searching begins with the tape moving forward.

If the tape has stopped, a subsequent tape instruction will begin with tape motion in the backward direction. It will search backwards until a block number has been encountered. This first block number is used to determine the required direction to complete the instruction.

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.

3.6.4 **LINCtape Instructions**

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

Data sum	two's complement sum of all 256 data words in a block
Checksum	two'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 $\neq 7777_8$
MBLK, TBLK	As shown in Figure 3-18

NOTE

> It is assumed that Extended Address and NO PAUSE are not used in the following discussion.

RDE Read Tape

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

RDC Read and Check

Form:	RDC I U
Octal code:	0700 + 20I + 10U
Execution time:	3.2 μs
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 μs
Operation:	Block TBLK is read from tape into the memory block designated by the three low-order bits of TBLK; e.g., tape block 773 is read into memory block 3, tape block 027 into memory block 7. The next consecutive tape blocks are read into successive memory blocks. Tape block 000 follows tape block 777, and memory block 0 follows memory block 7. MBLK is the number of additional consecutive blocks to be transferred.

Example: Transfer blocks 202-205 from unit 1 to memory, leaving the unit in motion at the end.

```
Instruction      Octal
.....            .....
RCG I 1          0731      /READ AND CHECK GROUP INST
3202             3202      /MBLK = 3, THE NUMBER OF ADDITIONAL BLOCKS
                           /TBLK = 202.
```

should be U or 10

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.

WRI Write Tape

Form:	WRI I U
Octal code:	0706 + 20I + 10U
Execution time:	3.2 μs
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.

WRC Write and Check

Form:	WRC I U
Octal code:	0704 + 20I + 10U
Execution time:	3.2 μs
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 and PAC; otherwise, the operation is repeated. ?

WCG Write and Check Group

Form:	WCG I U
Octal code:	0705 + 20I + 10U
Execution time:	3.2 μs
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 next MBLK memory blocks are copied into the next successive tape blocks. Memory block 0 follows memory block 7, and tape block 000 follows block 777. The scheme is identical with that for RCG. If the transfer is successful, the transfer check (7777) is left in the TAC and AC; otherwise, the operation is repeated.

The following two instructions do not transfer data.

MTB Move Toward Block

Form:	MTB I U
Octal code:	0703 + 20I + 10U
Execution time:	3.2 μs
Operation:	Subtract the first tape block number encountered (or reverse block number, if the tape is moving backwards) 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 motion = 0 when this instruction is given, it starts moving backward; otherwise it continues in the direction in which it had been going. The MBLK bits of the second word are ignored.

CHK Check one Tape Block

Form:	CHK I U
Octal code:	0707 + 20I + 10U
Execution time:	3.2 μs
Operation:	Find block BN, form its transfer check and leave it in the TAC and AC. The information on tape and the contents of memory are unchanged.

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

TAC Tape Accumulator to AC

Octal code:	0003
Execution time:	1.6 μs
Operation:	The contents of the Tape Accumulator are placed in the central processor AC. The previous C(AC) are lost; C(TAC) are unchanged. This instruction is used to verify transfer Checksums for NO PAUSE tape operations. Because this instruction uses the tape ADDER data path, it must not be given while a tape operation is in progress.

3.6.5 **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. The function of these bits is described in Paragraphs 3.3.6 through 3.6.12.

3.6.6 ?

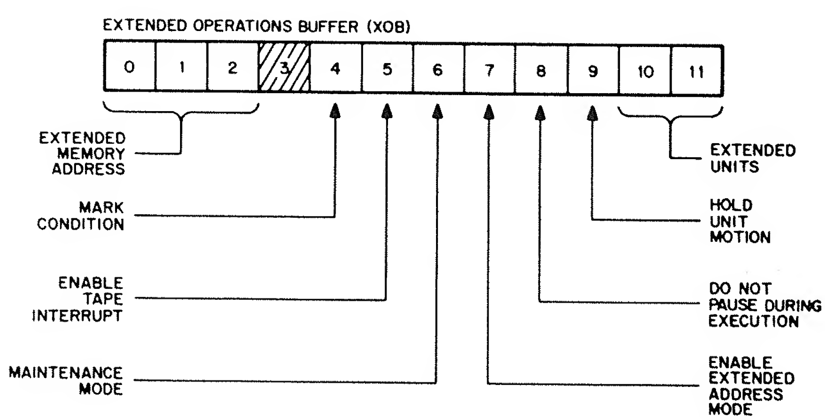


Figure 3-19. Extended Operations Buffer Bit Assignments

AXO AC to XOB

Octal code: 0001
Execution time: 1.6 μs
Operation: The contents of the AC are placed in the Extended Operations Buffer. The previous C(XOB) are lost; C(AC) are unchanged. Changing these states while a tape operation is in progress may cause incorrect execution.

XOA XOB to AC

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

This instruction must not be given when the tape is in progress because the data path includes the tape ADDER.

3.6.6 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 field, regardless of the settings of the Memory Field Registers. The first memory address affected in the data transfer is placed in the TMA Setup Register by the program, using the instruction TMA. When the Extended Address Mode is enabled (by setting bit 7 of the XOB to 1), all subsequent tape transfers are executed as follows. 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 an 11-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 4096_{10} memory field bank boundaries; address 7777 is followed by address 0000.

NOTE

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

TMA Load TMA Setup Register

Octal code: 0023
Execution time: 1.6 μs
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. This instruction must not be given when the tape is in progress because the data path includes the tape ADDER.

Example: Read the contents of block 365 of unit 0 into memory, beginning at absolute location 10540_8 (0540_8 in the second 4096_{10}-word bank of memory).

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

The data will be read into registers 0540_8 - 1137_8 of the second 4096_{10}-word bank of memory.

3.6.7 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 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	0
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

3.6.8 Tape Interrupt Enable

When this bit (XOB_5) is set, a program interrupt will occur whenever INTERRUPT is enabled and the TAPE DONE flag is set (i.e., the tape operation is completed). 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 the 8 mode, the interrupt uses registers 0001 and 0000.) TAPE DONE is set by the completion of a tape instruction and cleared by the execution of a new tape instruction or the tape maintenance instruction LMR (IOT 6151) with $AC_4(1)$. (See Paragraph 3.6.9 and Appendix E.)

3.6.9 **No Pause Condition**

Normally, the central processor waits until a tape operation is finished before proceeding. Such delays may be 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 flag to determine when the operation has finished. The Tape Done flag is set when a tape operation has been completed and is cleared at the beginning of the next tape instruction. The flag can also be sensed and cleared by the following instructions. When in the no pause condition, a second tape instruction should not be used until the previous tape instruction has been completed.

IOB
LMR *Load Maintenance Register*

Octal code: 0500
6151

Execution time: 5.9 μs (LINC mode)

Operation: The maintenance IOT (see Appendix E) is used to test and clear the TAPE DONE flag.

(AC_{05} = 1) Skip if Tape Done Flag is set.
(AC_{04} = 1) Clear Tape Done Flag

Do not have other AC bits on a 1, as this instruction has many additional functions.

STD *Skip if Tape Done*

Form: STD I
Octal code: 0416 + 20I
Execution time: 1.6 μs
Operation: If I = 0, skip the next instruction if no tape operation is in progress; otherwise, execute the next instruction. If I = 1, skip if an 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 in central processor programming. When NO PAUSE is set, the Transfer Check is not placed in the accumulator at the end of tape instruction. The instruction TAC (see Paragraph 3.6.4) should be used to recover the Transfer Check.

3.6.10 **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 caution. 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.

NOTE

> With the Hold Unit Motion and No Pause bits both set, it is impossible to do two back-to-back tape instructions and enable both units 0 and 1 for simultaneous motion.

3.6.11 **MARK Condition**

This bit (XOB_4) is used in conjunction with the MARK switch on the operator's console to allow the MARK 12 program (see Chapter 7 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.

3.6.12 **Maintenance Mode**

When XOB_6 is set, all timing signals and data are prevented from entering the tape control registers from the reader-writers. Instead, signals generated by 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 and is not intended for general use. See Appendix E for a list of Maintenance Mode tape control instructions.

3.6.13 **Tape Trap**

Whenever the TAPE TRAP and INSTRUCTION TRAP Special Functions are enabled (ESF with AC_{2-3} set), 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 (LINC-8 Trap Simulator) to ensure compatibility. Also, device-independent software can make use of Tape Trap to substitute other mass storage devices, such as disks for LINCtape.

3.6.14 **Tape Word Skip**

TWC Skip on Tape Word Complete

Form:	TWC I
Octal code:	0417 + 20I
Execution time:	1.6 μs
Operation:	This instruction is used when formatting a tape using the MARK 12 Program.

In reading consecutive blocks, a subsequent tape instruction must be issued within about* a millisecond after the completion of the first tape operation in order to prevent the tape motion's reversing between operations.

* 1196 μsec is flakey; 1164 looks pretty safe

CHAPTER 4
8 MODE PROGRAMMING

8-mode programming of the PDP-12 is covered in this chapter, which is divided into six sections: Organization of Memory, Memory Addressing Methods, PDP-8 Instructions, Program Interrupt, Extended Arithmetic Element, and Extended Memory.

4.1 ORGANIZATION OF MEMORY

4.1.1 Organization

In the 8 mode, the basic 4096-word memory is divided into 32 *pages* 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 the 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 field in 8 mode is shown in Figure 4-1.

4.1.2 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 4.2.2). 4.2.3

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

12-0121

Figure 4-1. Organization of Memory, 8 Mode

4.1.3 Extended Memory

Additional 4K memory fields are organized in the same manner as the basic field. The Memory Field registers determine the assignment of fields (see Paragraph 4.6).

4.2 MEMORY ADDRESSING METHODS

4.2.1 Direct Addressing

In the 8 mode, all memory reference instructions have the same structure, which is 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: 3350. The page address is 150; bit 4 (Page bit) 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 bit set to 0, the complete 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.

4.2.2 **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 5–11 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 field.

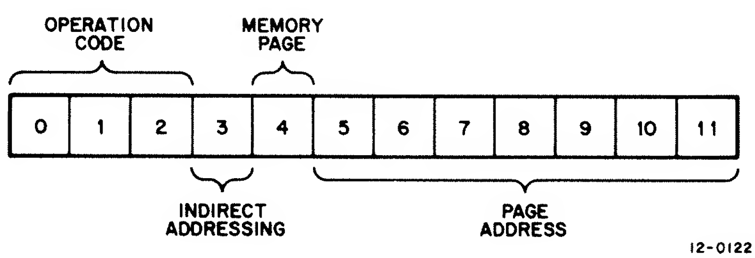


Figure 4-2. Memory Reference Instruction Format

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

Examples:

a. 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)

b. 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 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.

4.2.3 **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; 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
END,	HLT	/WHEN FINISHED, STOP; AC CONTAINS THE SUM
. . . .		
LIST,	777	/ADDRESS-1 OF TABLE OF ADDENDS
COUNT,	-100	/COUNT OF TABLES 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, C(10) is again incremented by 1, to 1001, for the next operand. At the end of the sequence, C(10) = 1077.

4.3 8 MODE INSTRUCTIONS (See Appendix B)

4.3.1 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, 1.6 microseconds is added to the execution time.

DCA Deposit and Clear Accumulator

Form:	DCA Y
Octal code:	3000 + Y
Execution time:	3.2 μs
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 Two's Complement Add to Accumulator

Form:	TAD Y
Octal code:	1000 + Y
Execution time:	3.2 μs
Operation:	The contents of Y are added to the contents of the AC, using two'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 μs
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 operation is performed on corresponding bits of each operand, 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 μs
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 μs
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 μs
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.

4.3.2 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, program halt, and access to the console switches.

Operate class instructions are microprogrammable; 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.

4.3.2.1 **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 4.3.2.2.

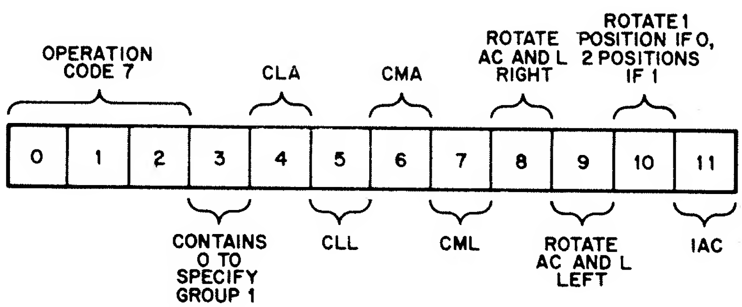


Figure 4-3. Group I Operate Class Instruction Format

NOP No Operation

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

CLA Clear Accumulator

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

CLL Clear Link

Octal code: 7100
Operation: The content of the Link is cleared to 0.

CMA Complement Accumulator

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

CML   Complement Link

Octal code: 7020
Operation: The content of the Link is complemented.

IAC   Increment Accumulator

Octal code: 7001
Operation: The contents of the AC are incremented by 1, using two's complement arithmetic. A carry out of bit 0 complements the Link.

RAR   Rotate Accumulator Right

Octal code: 7010
Operation: The contents of the AC and Link, taken as a 13-bit register, are rotated right one position. A bit 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.) This is the equivalent of two RAR instructions.

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.) This is equivalent to two RAL instructions.

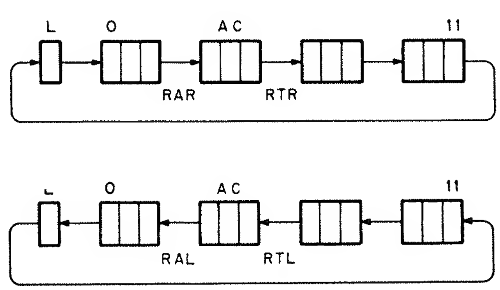


Figure 4-4. Rotation Scheme for RAR, RTR, RAL, RTL

4.3.2.2 **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 two's complement of the original C(AC). The two's complement of a number is defined as the one'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:

Octal Code	Combination				Resulting C(AC)
7201	CLA	IAC			0001
7326	CLA	STL	RTL		0002
7325	CLA	STL	IAC	RAL	0003
7307	CLA	CLL	IAC	RTL	0004
7327	CLA	STL	IAC	RTL	0006
7344	STA	CLL	RAL		7776 (-2)
7346	STA	CLL	RTL		7775 (-3)
7330	CLA	STL	RAR		4000 (-4000)
7332	CLA	STL	RTR		2000
7333	CLA	STL	IAC	RTR	6000 (-2000)
7352	STA	CLL	RTR		5777
7350	STA	CLL	RAR		3777

4.3.2.3 **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 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 a 0, the logical OR of the tested conditions will cause a skip; if this bit is a 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 4.3.2.4.

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

1. Skips
2. CLA
3. OSR
4. HLT occurs after all other specified operations have been performed.

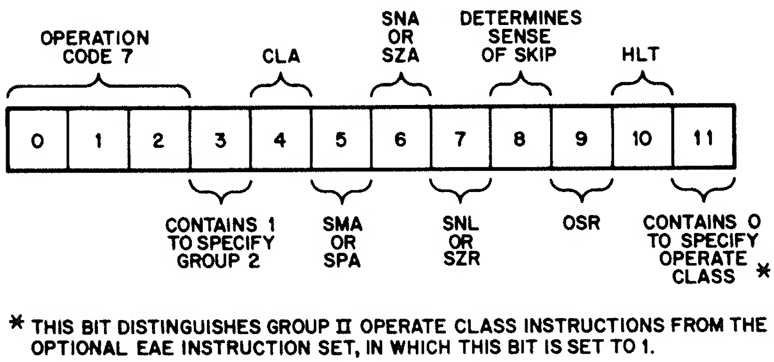


Figure 4-5. Group II Operate Class Instruction Format

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 unconditionally skipped.

SNL Skip On Non-Zero Link

Octal code: 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

two's complement of such a number is its positive counterpart. In this sense, 7777 is equivalent to -1; 4000 is equivalent to -4000. The two's complement sum of a number and its two'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.

4.3.2.4 **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.
		A skip will occur if and only if
SNA SZL	7470	C(AC)≠0 and C(L)=0
SNA SPA	7550	C(AC)≠0 and C(AC_0)=0
SPA SZL	7530	C(AC_0)=0 and C(L)=0
SNA SPA SZL	7570	C(AC)≠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.

4.3.2.5 **Input/Output Transfer Class** – These instructions, all of which have the basic operation code of 6000, are used to service peripheral devices, enable and disable the program interrupt, operate the memory extension control, change from 8 to LINC programming mode, and provide maintenance operations for the LINCtape subprocessor. Most of these instructions are described in Chapter 6 with their associated devices. The program interrupt and memory extension control are discussed with their respective instructions in Paragraphs 4.4 and 4.6 of this chapter.

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

LINC   Switch to LINC Mode

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

4.4 PROGRAM INTERRUPT

4.4.1 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 (PI) 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 that flag is raised:

1. The instruction in progress at the time of the PI 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 PI facility is disabled.

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

ION   Interrupt On

Octal code:	6001
Execution time:	4.25 μs
Operation:	The PI facility is enabled immediately after the instruction following the ION has been executed. If a PI 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 a PI service routine can return to the interrupted program before a subsequent PI request destroys the contents of register 0000.

IOF   Interrupt Off

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

4.4.2 Using the Program Interrupt

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

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

The PI 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 PI.

NOTE

Refer to Paragraph 3.3.15 for the discussion of additional aspects of Program Interrupt during LINC mode programming.

4.5 EXTENDED ARITHMETIC ELEMENT TYPE KE12

4.5.1 Operation

The Extended Arithmetic Element (EAE), Type KE12, adds a complete automatic multiplication and division facility to the PDP-12. Programming is provided by a class of 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.

4.5.2 EAE Instructions

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

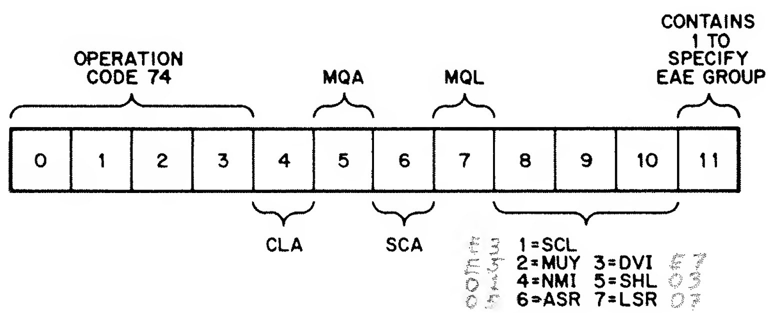


Figure 4-6. EAE Instruction Format

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, SHL, ASR, LSR

CLA Clear AC

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

CAM Clear AC and MQ

Octal code: 7621
Execution time: 1.6 μs
Operation: The AC and the MQ is cleared.

MQA Place MQ in AC

Octal code: 7501
Execution time: 1.6 μs
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 (Paragraph 3.3.11).

MQL Load MQ from AC

Octal code: 7421
Execution time: 1.6 μs
Operation: The MQ is cleared. 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.

SCA Step Counter to AC

Octal code: 7441
Execution time: 1.6 μs
Operation: The contents of the step counter are ORed inclusively with the contents of AC_{7-11}; the result is left in the AC. To obtain the actual step count, SCA is combined with CLA (combined operation code: 7641).

All other 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 μs
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: 9.0 μs
Operation: The number in the MQ is multiplied by the number in register p + 1. At the conclusion of this instruction 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.
plus the original contents of the AC

DVI Divide

Octal code: 7407
Execution time: 4.0 μs to 10.0 μs
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 (not microprogrammable)
Execution time: 1.6 μs + 0.40 μs/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 $C(AC_1)$ or until 6000 0000 is contained in the combined AC and MQ. 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, which is the exponent of the normalized floating point fraction. The shift path is shown in Figure 4-7.

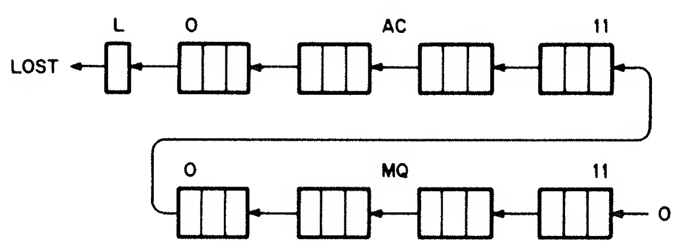


Figure 4-7. Shift Path for NMI, SHL

SHL Shift Left

Octal code: 7413
Execution time: 3.2 μs + 0.40 μs/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 to that for NMI, as shown in Figure 4-7.

ASR Arithmetic Shift Right

Octal code: 7415

Execution time: 3.2 μs + 0.40 μs/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 reproduced 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.

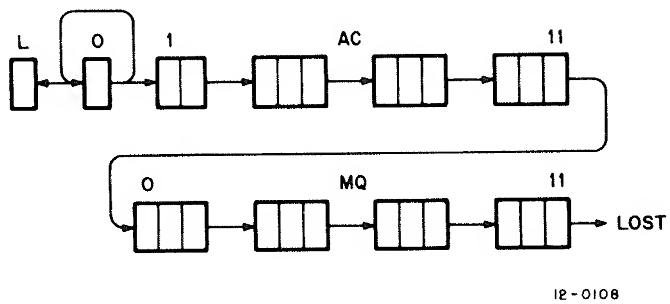


Figure 4-8. Shift Path for ASR

LSR Logical Shift Right

Octal code: 7417

Execution time: 3.2 μs + 0.40 μs/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. Bits shifted out of the MQ_{11} are lost; zeros are shifted into the Link and down the register. The shift path is shown in Figure 4-9.

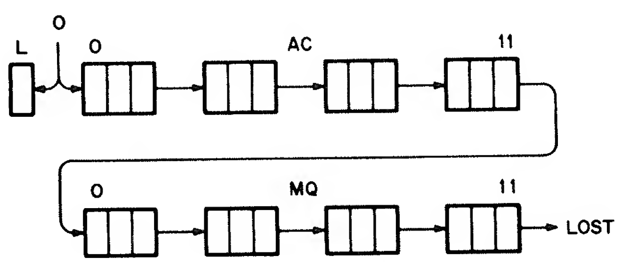


Figure 4-9. Shift Path for LSR

4.5.3 EAE Programming

4.5.3.1 **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 location (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 9.0 microseconds; at the end of this time, the next instruction is executed.

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

a. 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,         0000            /MULTIPLIER
                DCA  HIGH       /STORE HIGH ORDER PRODUCT
                MQA             /LOAD AC WITH LOW ORDER PRODUCT
```

b. 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 and 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,         0000            /MULTIPLIER
                DAC HIGH
                TAD SIGN
                RAR             /LOAD LINK WITH SIGN INDICATOR
                MQA
                SNL             /IS PRODUCT NEGATIVE?
                JMP LAST        /NO, MULT DONE, EXIT
                CLL CMA IAC     /YES
                DCA LOW         /COMPLEMENT RESULT
                TAD HIGH
                CMA
                SZL             /LINK USED TO COUPLE CARRY
                                /FROM BIT 12 TO BIT 11
                                /OF DOUBLE-LENGTH PRODUCT
                IAC
                DCA HIGH
                SKP
LAST,           DCA LOW
```

4.5.3.2 **Division** – Division is performed as follows:

1. Load the least significant 12 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 most significant 12 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 location (p + 1). This operation takes a maximum of 10.0 microseconds; when complete, a 12-bit quotient is held in the MQ, the 12-bit remainder is in the AC, and the Link holds 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, then repeat the divide routine.

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

a. Division of 24-Bit Unsigned Numbers

Enter with the low order 12 bits of the dividend in the AC and the high order 12 bits of the dividend in memory location labeled HIGH 12. The divisor is in memory location labeled DIVISOR upon entry. Exit with the quotient in the AC and the remainder in location labeled REMAIN.

```
                CLL                     /CLEAR LINK FOR OVERFLOW CHECK.
                MQL                     /LOAD MQ WITH LOW ORDER DIVIDEND
                TAD HIGH 12             /LOAD AC WITH HIGH ORDER DIVIDEND
                DVI                     /INITIATE DIVIDE
DIVISOR, (12 BIT DIVISOR HERE)
                SZL                     /OVERFLOW?
                JMP EXIT                /YES – EXIT
                DCA REMAIN              /NO – STORE REMAINDER
                MQA                     /AND LOAD AC WITH QUOTIENT
```

b. Division of 24-Bit Signed Numbers

Enter with the low order 12 bits of the dividend in memory location labeled LOW 12 and the high order 12 bits of the dividend in memory location labeled HIGH 12. The 12 bit divisor is in location labeled DIVISOR. Exit with the unsigned remainder in location labeled REMAIN, and the signed quotient in the AC.

```
                CLA  CLL                /CLEAR AC AND LINK
                TAD  HIGH 12            /LOAD AC WITH HIGH ORDER DIVIDEND
                SMA  CLA                /DIVIDEND NEGATIVE?
                JMP .+12                /NO – SKIP NEGATION
                TAD  LOW 12             /YES – LOAD AC WITH LOW ORDER DIVIDEND
                CMA  IAC                /NEGATE IT AND
                DCA  LOW 12             /STORE IT BACK
                TAD  HIGH 12            /LOAD AC WITH HIGH ORDER DIVIDEND
```

```
            CMA                     /NEGATE IT
            SZL                     /WAS THERE A CARRY FROM LOW ORDER?
            IAC                     /YES – INCREMENT HIGH ORDER
            DCA  HIGH 12            /AND STORE IT BACK
            CLL  CML                /SET LINK TO 1 FOR SIGN CHECK
            TAD  DIVISOR            /LOAD AC WITH DIVISOR
            SPA                     /IS IT POSITIVE?
            CMA  CML  IAC           /NO – NEGATE IT
            DCA  DIVISOR            /AND STORE IT BACK
            SNL                     /CHECK LINK FOR SIGN OF RESULT
            CMA                     /POSITIVE – STORE -1 IN SIGN
            CLL                     /NEGATIVE – STORE 0 IN SIGN
            DCA  SIGN
            TAD  LOW 12             /GET LOW ORDER DIVIDEND
            MQL                     /STORE IT IN MQ
            TAD  HIGH 12            /LOAD AC WITH HIGH ORDER DIVIDEND
            DVI                     /DIVIDE
DIVISOR, (DIVISOR STORED HERE)
            SZL                     /OVERFLOW?
            JMP  EXIT               /YES – EXIT ON OVERFLOW
            DCA  REMAIN             /NO – STORE UNSIGNED REMAINDER
            MQA                     /LOAD AC WITH QUOTIENT
            ISZ  SIGN               /SHOULD QUOTIENT BE NEGATIVE?
            CMA  IAC                /YES – NEGATE IT
```

4.6 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 Paragraph 3.3.15 in relation to LINC mode memory control. In the 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 five bits long; the two low-order bits of the 5-bit total pertain only to LINC mode programming operations. In 8 mode the Save Field register (Interrupt Buffer) uses only six bits; the four low-order bits are unused.

4.6.1 Registers

4.6.1.1 **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 to 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.

4.6.1.2 **Data Field Register (DF), 3 Bits** – These three bits serve as an extension of the Memory Address register for determining which memory field contains the operands to be accessed by the memory reference instructions AND, TAD, DCA, and ISZ when indirect addressing is used. The Data Field and Instruction Field may be set to the same field.

4.6.1.3 **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.

4.6.1.4 **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 PI 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.

4.6.1.5 **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 field to be accessed.

4.6.2 Instructions

All Extended Memory IOT instructions require 4.3 microseconds for execution.

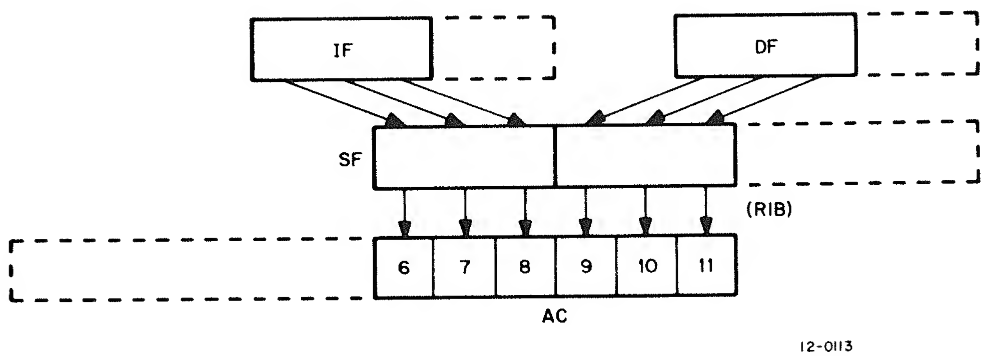


Figure 4-10. Data Path to SF and AC

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 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 ORed into AC_{6-8}. The other bits of the AC are unaffected.

RIF Read Instruction Field

Octal code: 6224

Operation: The contents of the Instruction Field register are ORed into AC_{6-8}. 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 ORed into AC_{6-8}; bits 3-5 (DF) are ORed into 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 field 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.

4.6.3 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
Indirect	Page	IF	DF	Address
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 contents of 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, in the location specified by the contents of the effective address.

4.6.3.1 **Autoindexing** – When any memory field is used as an Instruction Field, registers 10-17 of that field 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 field 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.

4.6.3.2 **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  KCDF          /CDF INSTRUCTION TO AC
                 RDF                /C(DF) TO AC 6-8 FORMS CDF 10 (6211)
                 DCA  RESDF         /STORE IN SEQUENCE TO RESTORE DF
                 TAD  KCDF          /CDF TO AC
                 RIF                /C(IF) TO AC 6-8 FORMS CDF 20 (6221)
                 DCA  SETDF         /STORE IN SEQUENCE TO SET DF
SETDF,           0000               /SETS DF TO CURRENT IF
                 CIF  40            /SET IF BUFFER TO SUBROUTINE FIELD 4
                 JMS  I  SUBADR     /JUMP TO SUBROUTINE IN FIELD 4
RESDF,           0000               /RESTORES ORIGINAL DF (FIELD 1)
                 ....
SUBADR,          SUBRTN             /ABSOLUTE ADDRESS OF SUBROUTINE
KCDF,            CDF                /CONSTANT

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

SUBRTN,          0                  /C(PC) FROM CALLING PROGRAM
                 TAD  KCIF          /CIF INSTRUCTION TO AC
                 RDF                /C(DF) TO AC 6-8 FORMS CIF 20 (6222)
                 DCA  RESIF         /STORE IN SEQUENCE TO RESTORE CALLING FIELD
                 ....
                 ....
                 ....
RESIF,           0000               /SETS IF BUFFER TO RESTORE CALLING FIELD
```

```
                JMP I SUBRTN      /JUMP BACK TO CALLING PROGRAM
KCIF,           CIF               /CONSTANT
```

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.

4.6.3.3 **Program Interrupt** – If, when the PI facility is enabled, a PI 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 register, a PI 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 DESCRIPTION

Because the processing power of a 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. This chapter defines the interface characteristics of the computer thus allowing the user to design and implement any electrical interfaces required to connect I/O 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 existing or future patent rights. Nor do the descriptions contained herein imply the granting of licenses to use, manufacture, or sell equipment constructed in accordance herewith.

The PDP-12 contains a central processor and core memory composed of Digital's M Series TTL circuit modules. These circuits have an operating temperature range exceeding the limits of 50° F to 110° F, so air-conditioning is not required. Standard 115V, 50/60-Hz 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 disk 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 appropriate interfaces. 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. The interface of any device to the computer does not require any modification to the central processor, and thus 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 I/O device can exercise this control. Transfers controlled by the computer, hence under control of its stored program, are called programmed data transfers.

Transfers made under control of the external I/O devices through the data break facility are called data break transfers.

Programmed Data Transfers

The majority of I/O transfers occur under program control. To transfer and store information under program control takes about six times as much computer time as under the data break facility. 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 programmed data transfers.

Data Break Transfers

Devices which operate at very high rates of speed or which require very rapid response from the computer use the data break facilities. Use of these facilities permits an external I/O device to insert or extract words from the computer core memory, almost arbitrarily bypassing all program control logic. Because the computer program has no cognizance of such transfers, 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 disk 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 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. The contents of the program counter are stored in this location, and subroutine for servicing the I/O device beginning in the next core memory location is executed. After completion of the data transfer, the interrupted program is resumed by returning to the location specified by the contents of the program counter, which were saved when the interrupt occurred. 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 at all other times is ground. Similarly, AC CLEAR L is at ground for assertion, and positive otherwise.

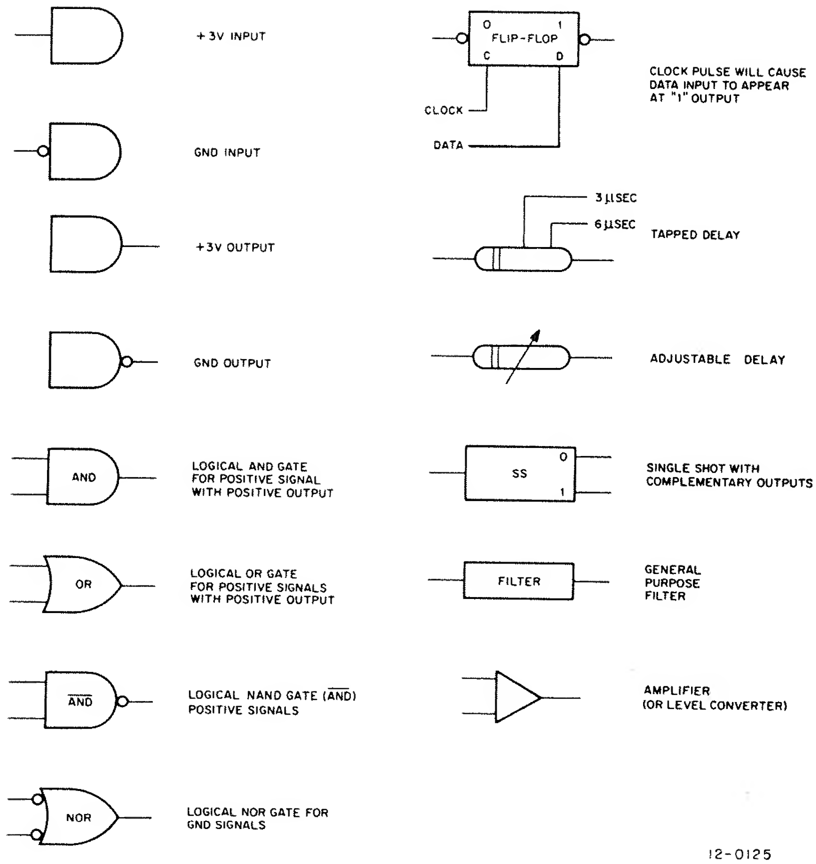


Figure 5-1. Logic Symbols

Signals originating at flip-flops are defined 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 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 kHz = 1/6.8 μsec. when status checking (end, transfer check, etc.) is not 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 are connected 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 to service an I/O device.

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

a. Each device must be able to sample the select code, generated by the computer during an IOT instruction*, and, when selected, must be able to produce sequential IOT command pulses in accordance with the computer-generated IOP pulses. Circuits which perform these functions in the peripheral device are called Device Selectors (DS). Figure 5-2 shows the decoding of the IOT instruction.

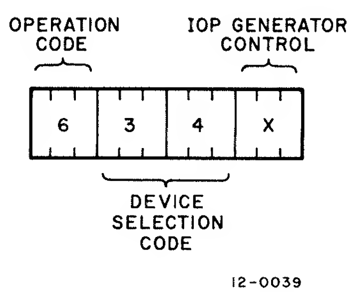


Figure 5-2. IOT Instruction Decoding

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.

---

*Appendix C provides a listing of IOT instructions.

c. Each device sending input data to the computer must contain gating circuits at the output of the transmitting register capable of sampling the information in the computer 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 that can pulse the computer I/O skip bus (IOS) 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 I/O equipment. All instructions stored in core memory as a program sequence are read into the memory buffer register (MB) and the instruction register (IR) 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-microsecond 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.

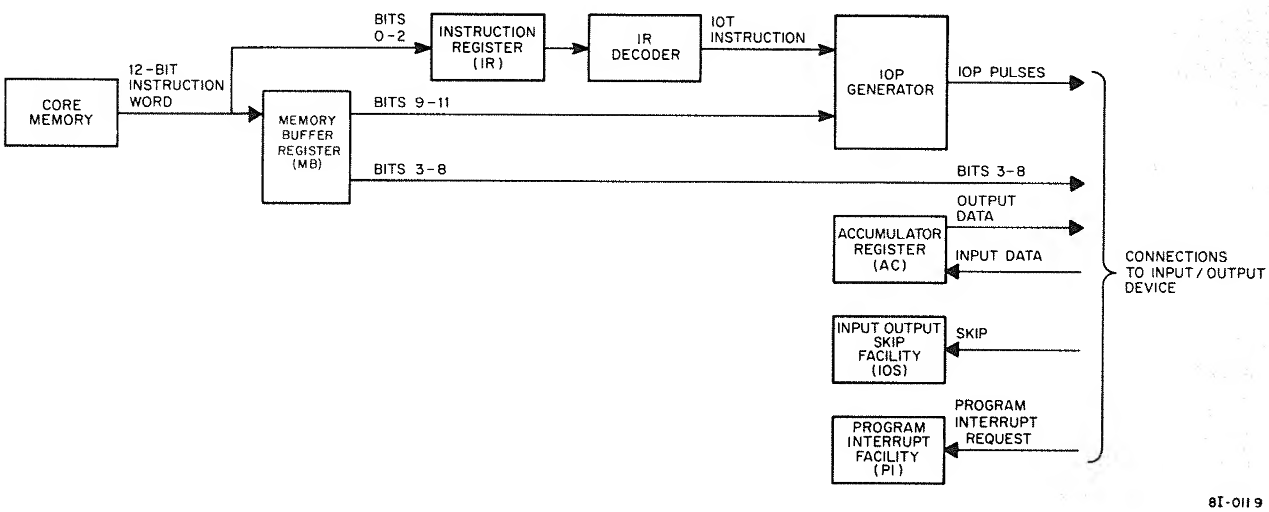


Figure 5-3. Programmed Data Transfer Interface Block Diagram

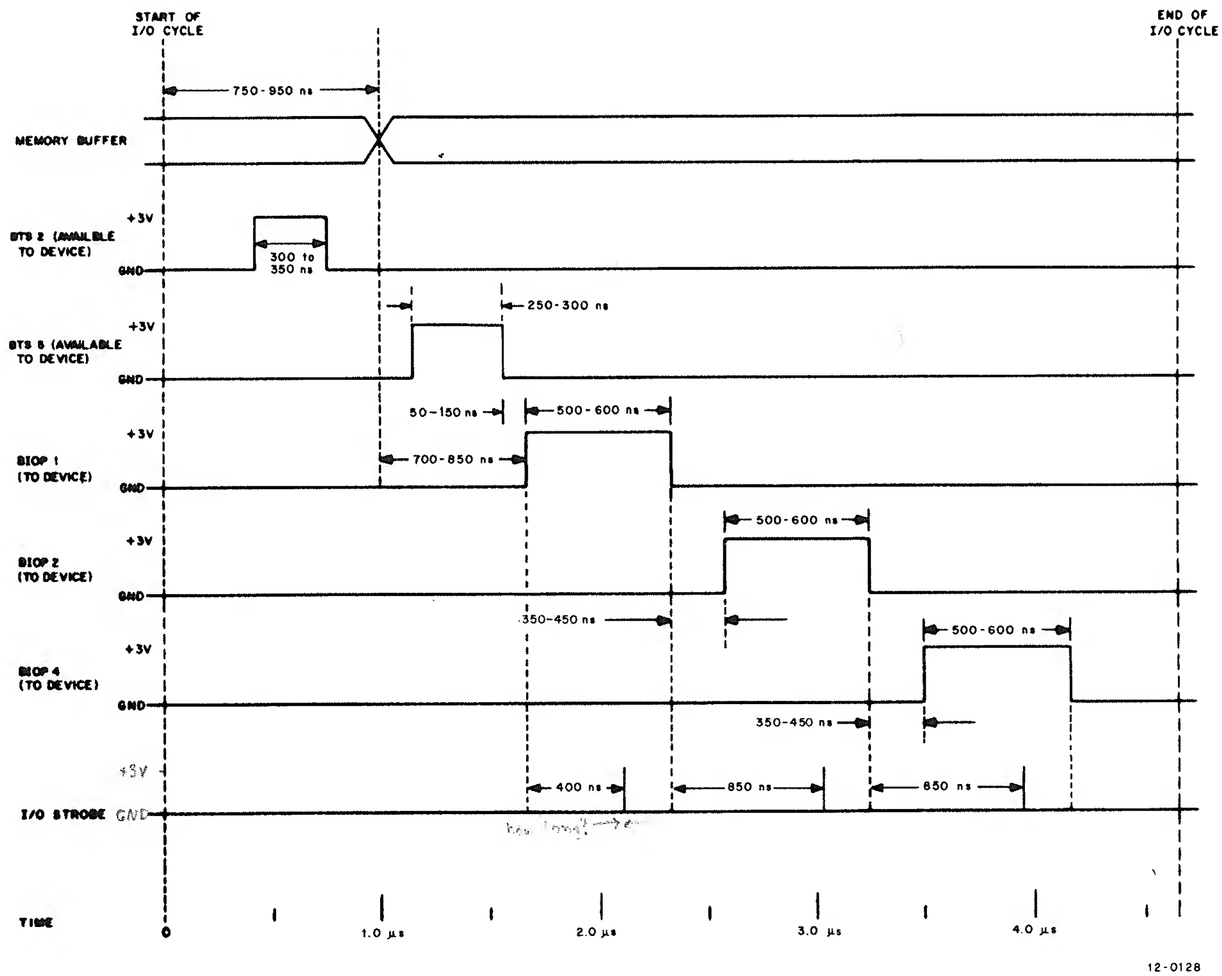


Figure 5-4. Programmed Data Transfer Timing

Figure 5-4 indicates the timing of programmed data transfers.

Devices which require immediate service from the computer program, or which require too much 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 is ANDed 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 does not make a 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.

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 the 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.

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). Therefore, one I/O device can 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.

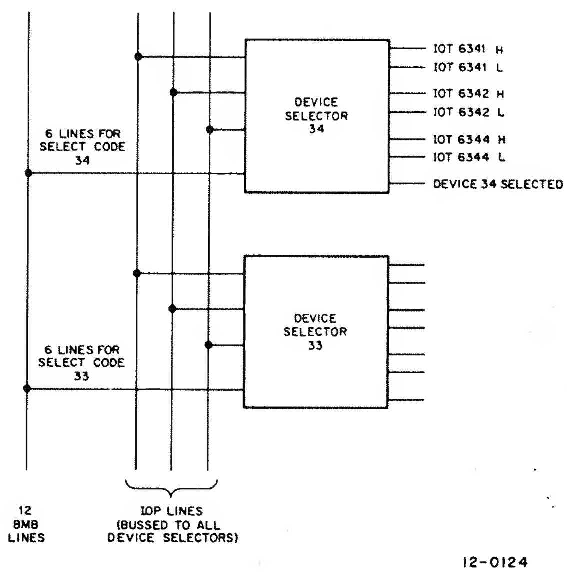


Figure 5-5. Generation of IOT Command Pulses by Device Selectors

The logical representation for a typical channel of the DS, using channel 34, is shown in Figure 5-6. An 8-input AND gate is wired to receive the appropriate signal outputs from the MB_{3-8} for select code 34, which activates the channel. In the DS module, 6 input pins are connected 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 AND gate when establishing the select code. The positive output of the AND gate indicates when the IOT instruction selection selects the device, and can, therefore, enable circuit operations with the device. This output also enables three power NAND gates, each of which produces a ground output pulse if the corresponding IOT pulse occurs. The ground 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 negative IOT pulses to produce complementary IOT output pulses.

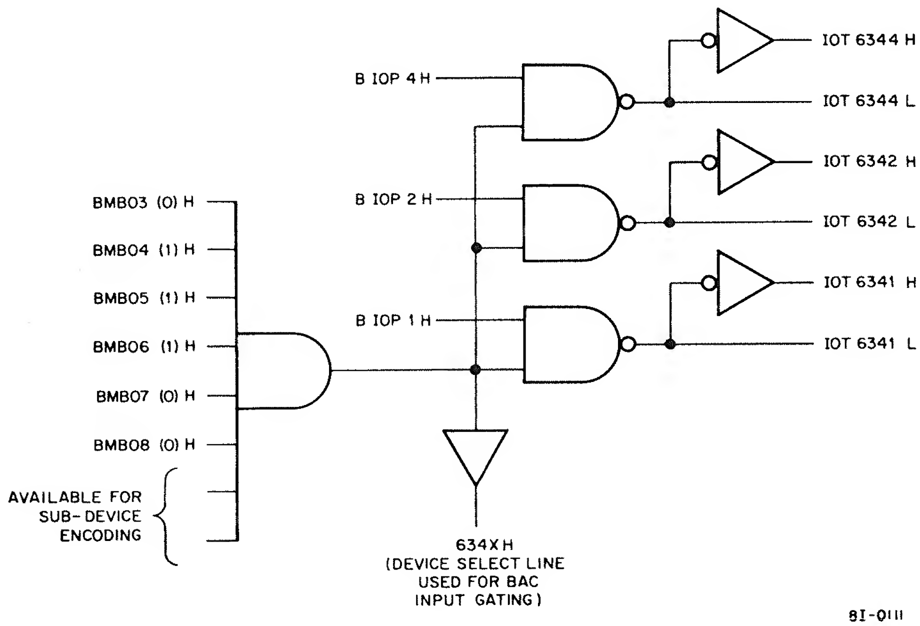


Figure 5-6. Typical Device Selector (Device 34)

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 contents of the program counter are 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.

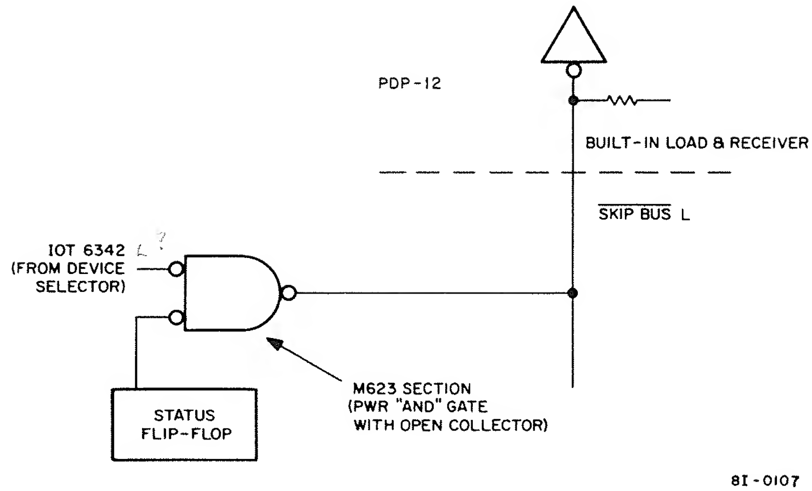


Figure 5-7. Use of IOS to Test the Status of an External Device

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 location.

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. The I/O bus input 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 input bus terminal to ground sets the corresponding AC flip-flop. 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.

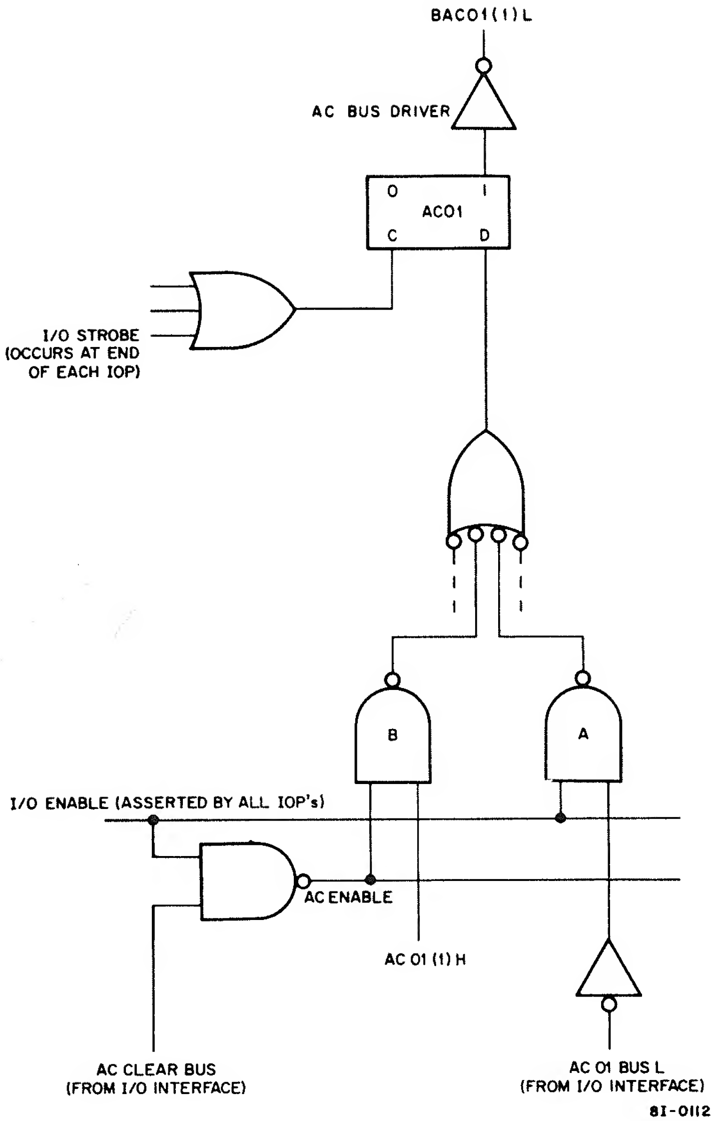


Figure 5-8. Accumulator Input or Output

5.1.5 Input Data Transfers

When a device is ready to transfer data into the PDP-12 accumulator, it sets a flag. The program senses the ready status of the flag and issues an IOT instruction to read the contents 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.

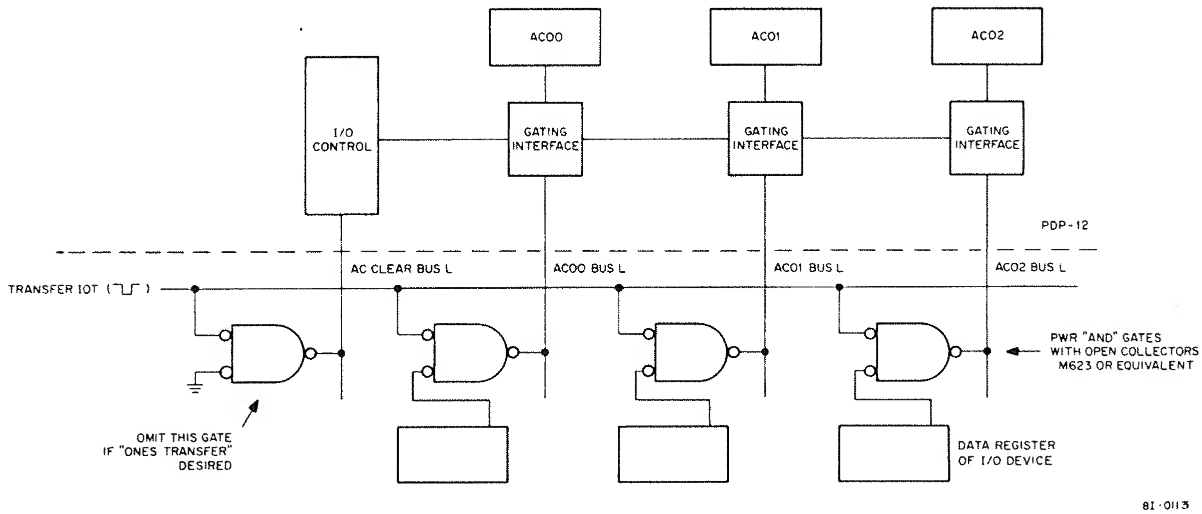


Figure 5-9. Loading Data into the Accumulator from an External Device

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.

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 a program interrupt occurs, program control transfers to a subroutine that determines which device requested the interrupt, and initiates an appropriate service routine.

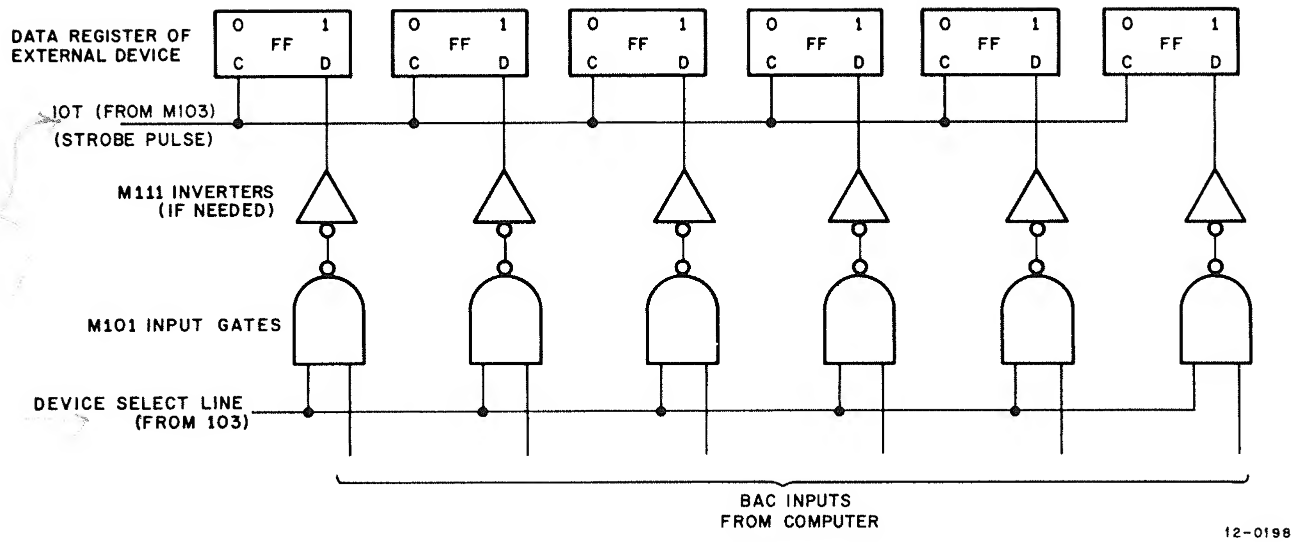


Figure 5-10. Loading a Six-Bit Word into an External Device from the Accumulator

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

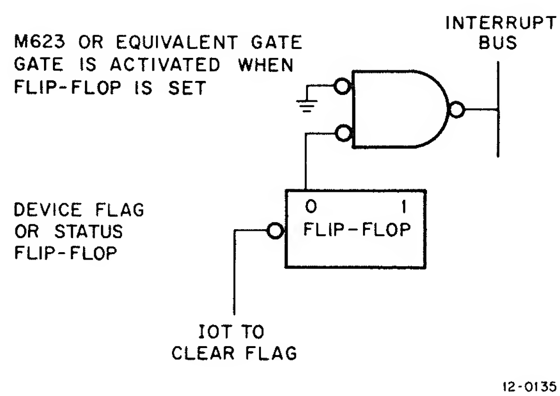


Figure 5-11. Program Interrupt Request Signal Origin

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 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
        3002   RMF               /RESTORE MEMORY FIELDS
        3003   ION               /TURN ON INTERRUPT
        3004   JMP I 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.

I/O Skip

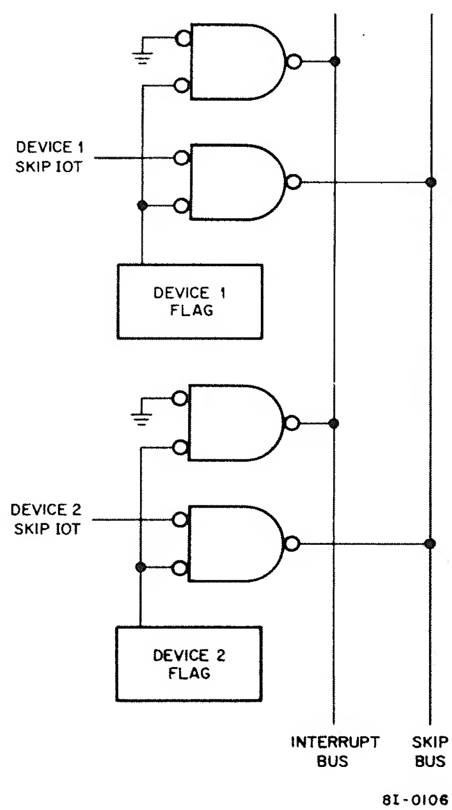


Figure 5-12. Multiple Inputs to IOS and PI Facilities

5.1.7.1 **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 identify the device making the request. When a program interrupt occurs, a routine is entered from address 0001 in 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.

The following program example illustrates how the program interrupt routine determines the device requesting service (example in 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 SR54	/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,	DCA TEMP	/SAVE AC
	IOT XX	/TRANSFER DATA FROM DEVICE BUFFER TO AC
	DCA 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 - DCA I 10 sequence might be replaced by a TAD I 10 - then - IOT sequence.

5.2 MULTI-LEVEL AUTOMATIC PRIORITY INTERRUPT

The KF12B Multi-Level Automatic Priority Interrupt is designed to reduce the central processor overhead during the servicing of program interrupts. It is prewired in the EP section of the PDP-12 in racks P and R and utilizes approximately 55 M series modules. There are three major services provided automatically by the KF12B.

a. Automatic determination of device priority and vectoring of interrupt service routines.

b. Automatic saving and restoring of all major registers and machine status which include the following:
   PC, AC, IF, DF, MQ, LINK, FLOW, UF, MODE and the current processor level.

c. Automatic stacking of the saved parameters permitting multiple levels of interrupts.

Storing, or stacking, of parameters is called *Pushing* and restoring the CP to its original status prior to an interrupt is called *Popping*. The CP is in the break state for the duration of each operation. It takes five break cycles for each *Push* and five break cycles for a *Pop*. This does not affect the normal operation of the data break facility in the PDP-12. One data break device can be handled without the addition of a multiplexer. The KF12B has the lowest priority on the bus and break requests from another device are acknowledged during push and restore operations. The KF12B control has its own timing generator and is asynchronous with computer timing. A free running 5 mHz oscillator provides the various clocking pulses. An M155 decoder provides the enable levels to enable data on the bus.

5.2.1 Interrupts

Up to 15 levels of interrupts can be accommodated with each level having a two-word vector address. The interrupts can be accepted from a prewired option or from up to six external devices. A priority is assigned to each interrupt by a jumper module, M905 at location R16. Level 0 has the highest priority. Interrupts of a higher priority can occur after executing the first instruction in the interrupt service routine. When the KF12B is not enabled (API ON (0)), interrupts are processed through the interrupt cycle in the normal manner.

5.2.2 Push

When the level of the device requesting an interrupt is greater than the current machine level a Push operation is performed. The Push and Break Req flip-flops are set and the processor enters the Break cycle. The active registers and status levels are stored (pushed) in five consecutive memory locations specified by the contents of the STACK register. (Refer to Table 5-1.) The starting location of the stack is specified by the program (IOT 6776) and is automatically incremented during the push operation. The stack increments and decrements across field boundaries. The CP is always in 8 MODE at the completion of an interrupt-push operation. If the CP is in LINC mode when a Push occurs it is returned to LINC mode at the completion of a Restore command.

Table 5-1. STACK Register

Location	Data Stored
P	AC_{0-11}
P + 1	PC_{0-11}
P + 2	$MODE_0$; $FLOW_1$; $LINK_2$; MACHINE $LEVEL_{8-11}$
P + 3	MQ_{0-11}
P + 4	UF_1; IF_{2-6}; DF_{7-11}
P = Initial STACK address. NOTE: the subscript indicates the corresponding memory bits.	

5.2.3 Restore–"POP" (REST–IOT 6771)

Every interrupt subroutine should be terminated with a Restore command. This restores the major registers and machine status from the stack and resumes programming at the memory location specified by the program counter.

For every Push operation performed a Pop (restore) must be performed; however, the two operations do not have to occur in any particular sequence (see Figure 5-13). The Restore command should not be issued when the CP is in a non-interruptable state because an Interrupt Inhibit is set due to the LIF or CIF instruction, or SAVE PC is not set due to a DJR instruction.

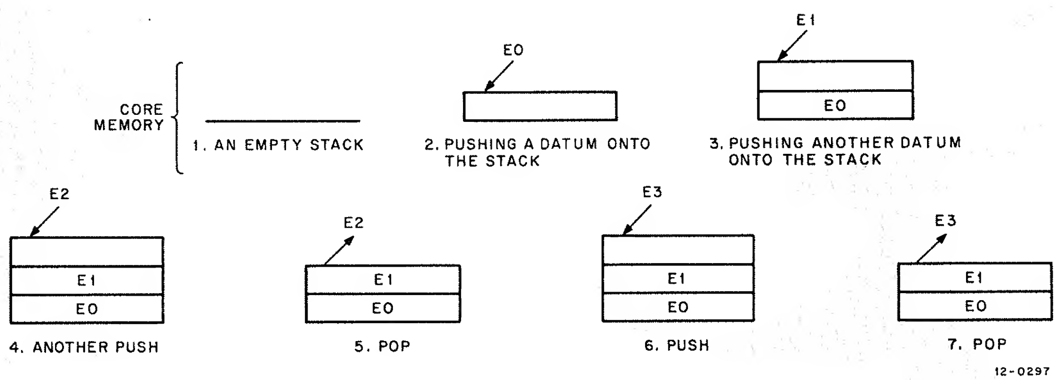


Figure 5-13. Illustration of Push and Pop Operations

5.2.4 Vectoring

Each of the 15 interrupt levels has an associated vector address to specify the appropriate interrupt service routine. The vector address is transferred to the PC during the Push operation, as shown in Figure 5-14. Vector bits 0, 1, and 2 specify the memory field and are set with AC bits 3, 4, and 5 by an IOT. Vector bits 3 through 9 specify the seven most significant bits of the MA (0–6) and are set with AC bits 0–6 by an IOT. The interrupt level specifies memory address bits 7 through 10 with level "0" setting these bits to zero. A vector address is always an even number address; therefore, each interrupt level is allotted two memory locations.

The following are the vector address assignments, which can reside in any memory field. Vector bits 3 through 9 (MA0–6) are set to 1s by the SVEC instruction:

Level	Address	Level	Address
0 Powerfail/restart	7740	8 Plotter control	7760
0	7741	8	7761
1 Line Tape	7742	9 RK08 disk	7762
1	7743	9	7763
2 Clock	7744	10	7764
2	7745	10	7765
3 Time-share option	7746	11	7766
3	7747	11	7767
4 TTY transmitter	7750	12	7770
4	7751	12	7771
keyboard 5 TTY receiver	7752	13	7772
5	7753	13	7773
6 Dataphone TTY rec.	7754	14	7774
6	7755	14	7775
7 Dataphone TTY trans	7756		
7	7757		

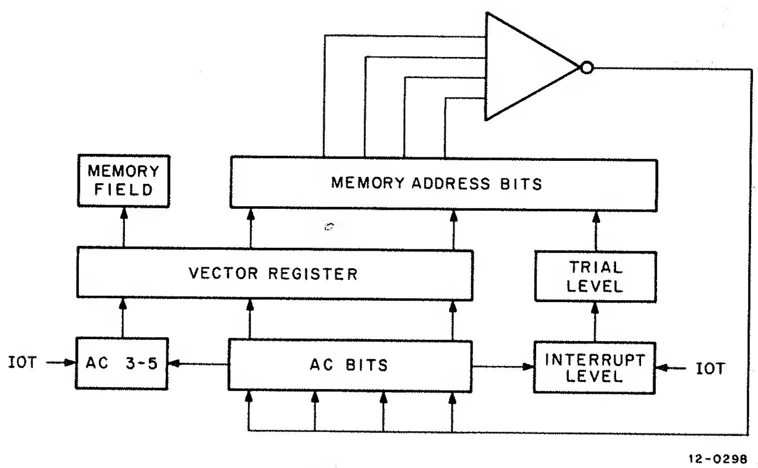


Figure 5-14. Vector Flow Diagram

5.2.5 Maintenance Logic

The maintenance logic included in the KF12B provides the capability of checking the major portion of the option for proper operation. Two IOT instructions simulate the 15 interrupt level inputs to check the priority logic and initiate the Push operation.

Instruction	Function
IOT 6051	AC0–11 to LEVELS 0–11 (1s transfer)
IOT 6052	AC 9, 10, 11 to LEVELS 12, 13, 14 (1s transfer)

The levels remain set for only one computer cycle. This feature allows enough time to initiate a Push function when the selected level has priority and API is enabled.

The KF12B features a new two-word instruction called push jump (PUSHJ, IOT 676X). This instruction permits jumping to subroutines across field boundaries in both Linc and PDP-8 Modes. The instruction causes the stacking of the active registers and machine status and automatically jumps to the memory location specified by the 15-bit address associated with the PUSHJ. The instruction code is (IOT) 676*X* and is similar to an IOT except that *X* defines the new memory field and the following location (P+1) specifies the 12-bit memory address of the subroutine.

The PC, which is saved on the stack during the execution of the PUSHJ, points to the location following the two-word instruction as shown in the following example:

Address	Instruction	Octal Code
15432	PUSHJ	6760
15433	1000	1000
15434*	CLA	7200

01000 – Programming is transferred to this memory location.

* Field and PC saved on stack. The program is resumed at this location following a Restore.

5.2.6 Programming

The following is a typical example of a program to service a Teletype interrupt, which is level 5.

```
START/CLA
        TAD FLDLEV
        SMLV                    /set stack and vector fields
        CLA                      and machine level
        TAD STACK
        SSTK                    /set stack
        CLA
        TAD VEC
        SVEC                    /set vector
        APION                   /enable KF12
        NOP
        JMP. -1                 /wait for interrupt

* 6412/JMS TTYR                 /go to teletype reader subroutine
* 6413/REST                     /restore to status prior to push
```

```
*VECTOR ADDRESSES

TTYR/XXXX                     /return address saved here
        CLA
        KRS                   /read keyboard buffer
        DCA I SAVE            /save word
        KCC                   /clear reader flag
        JMP I TTYR            /return to restore

FLDLEV/0017 <—— needs bit 7 if fields are to be set!
STACK/7000
VEC/6400
```

```
TTYR,   0000        /return addr saved here
        KRB         /clear AC, kybd bffr → AC, clear flag
        DCA I SAVE  /save word
        JMP I TTYR  /return to restore
```

5.2.7 **Programming Restrictions**

The REST and PUSHJ commands should not be issued when the processor is in a non-interruptable state due to the following conditions:

a. "Interrupt Inhibit" being set due to the execution of a CIF or LIF instruction;

b. The DJR instruction is being executed or Save PC is not set due to the previous execution of a DJR instruction.

If the REST command is issued and the CP is not in an interruptable condition, the Restore operation will not be performed until the condition becomes satisfactory. If the PUSHJ command is issued and the CP is not in an interruptable condition, the second word of the instruction is treated as a new instruction, and when the condition becomes satisfactory, the PUSHJ operation will not be properly executed. The ESF Disable Teletype instruction is not effective when the KF12B is enabled (API ON (1)); the TTY and DP12 interrupts will always be acknowledged. When the KF12B is not enabled (API ON (0)), the ESF instruction functions in the normal manner. Location zero is not saved on the stack; therefore, caution should be exercised when using this location in conjunction with a LINC JMP instruction.

5.2.8 **Instruction List**

Instruction	Function		Octal
ION	Enable software interrupt system		6001
IOF	Disable all interrupts		6002
APION	Enable API (KF12B) system Disable software interrupt The next instruction will be executed before an interrupt is processed.		6006
PUSH-J	PUSH JUMP to 15 bit address (XXXXX)	 (P+1)	676X XXXX
REST	Restore machine to previous level		6771
SMLV*	Set current machine level to AC 8–11. If AC7=1, set stack field to AC0–2 and vector field to AC3–5.		6772

IO PRESET SETS

$\underbrace{000}_{\text{STK FLD}=0}\ \underbrace{000}_{\text{VEC FLD}=0}\ x\underbrace{1\ 111}_{\text{MLV}=17}$

*The KF12B must be deselected to execute these instructions (i.e., API ON (0)).

Instruction	Function	Octal
RFLD	Read stack field into AC0–2, vector field into AC3–5, current level into AC8–11 (Read in complement form)	6773
RSTK	Read least significant 12 bits of current stack pointer address (Read in complement form)	6774
RVEC	Read vector bits 3–9 into AC bits 0–6, Trial levels into AC bits 7–10 (Read in complement form)	6775
SSTK*	Set the least significant 12 stack bits to AC0–11	6776
SVEC*	Set bits ~~309~~ 3-9 of the vector address (MA0–6) to AC 0–6	6777

Note: When the above instructions are given in LINC mode they should be preceded by an IOB instruction.

*The KF12B must be deselected to execute these instructions (i.e., API ON (0)).

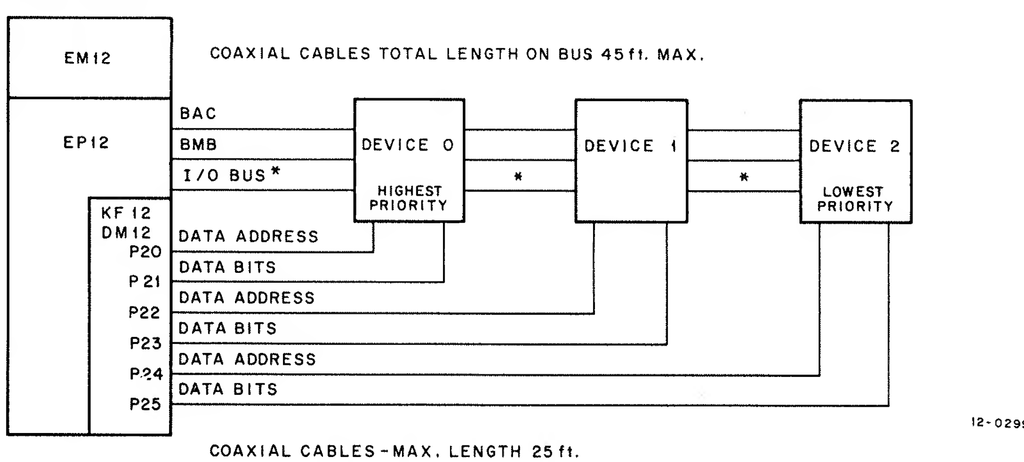
5.2.9 DM12

The DM12, which is prewired in the PDP-12, provides the capability of operating up to three data break devices in either three-cycle or single-cycle data break. Both three-cycle and single-cycle devices can be simultaneously controlled by the DM12. The KF12B is a prerequisite for the DM12 because the KF12B and DM12 use the same timing and control signals, which originate in the KF12B. The DM12 uses M series positive logic; all signals are clamped at 0V and +3V.

Because both options use the control and timing of the KF12B, the theory of operation is similar. When an external device issues a break request, the corresponding Level flip-flop is set. The three level flip-flops and the API flip-flop are decoded to determine priority. Either PR0, PR1, or PR2 is generated and the appropriate Data Address and control signals are enabled on the bus. The EN BRK flip-flop corresponding to the device that has been decoded is set at TP1, allowing the B BREAK signal to go to the device that has control of the bus. The B BREAK signal is used to control the data on the bus for transfer between memory and the external devices.

Both options use the same IOTs; thus, when testing the DM12 the KF12B must be disabled using jumpers that allow only the DM12 priorities to be enabled.

The DM12 priorities are determined by cable location. Two cables from each device are used for the data address, data bits, and status signals. The cables from the device having the highest priority are inserted in locations P20 and P21 as illustrated in Figure 5-15.



* If the priority interrupt system is used in the KF12B this cable is connected separately from each device. (ref. BS-KF12-CAB).

Figure 5-15. DM12 Cable Diagram

5.3 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. The Data Break is particularly well suited for devices which transfer large amounts of information in block form, and can be expanded to accommodate more than one device by using the DM01 or DM04 multiplexers. The DM01 will multiplex up to seven devices and is a negative bus option. The DM04 (positive bus) will multiplex three devices and up to three DM04s may be added for a total of nine devices.

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 6.5 million bits per second. Information flow to effect a Data Break transfer with an I/O device appears in Figure 5-16.

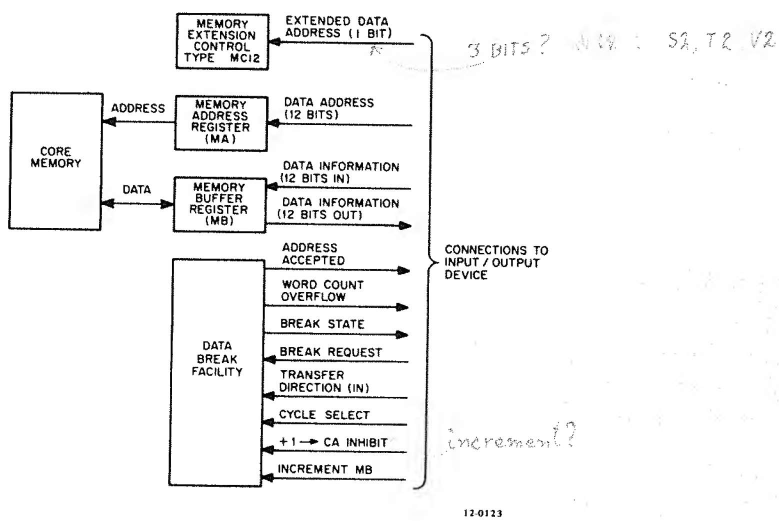


Figure 5-16. Data Break Transfer Interface Block Diagram

In contrast to programmed operations, the Data Break facilities permit an external device to control information transfers. Therefore, 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 input 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.

5.3.1 Single-Cycle Data Breaks

Single-cycle Data 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.

5.3.2 Input Data Transfers

Figure 5-17 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 5-18. The data register (DR in Figure 5-18) 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 5-17).

Appropriate output terminals of these registers are connected to the computer to supply ground potential to designate binary 1s. 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 5-18.

Other portions of the device interface, not shown in Figure 5-18, 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.

When the Break Request is recognized, 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 on 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 TP2 time of the data break cycle, the computer performs the transfer and returns to programmed operation.

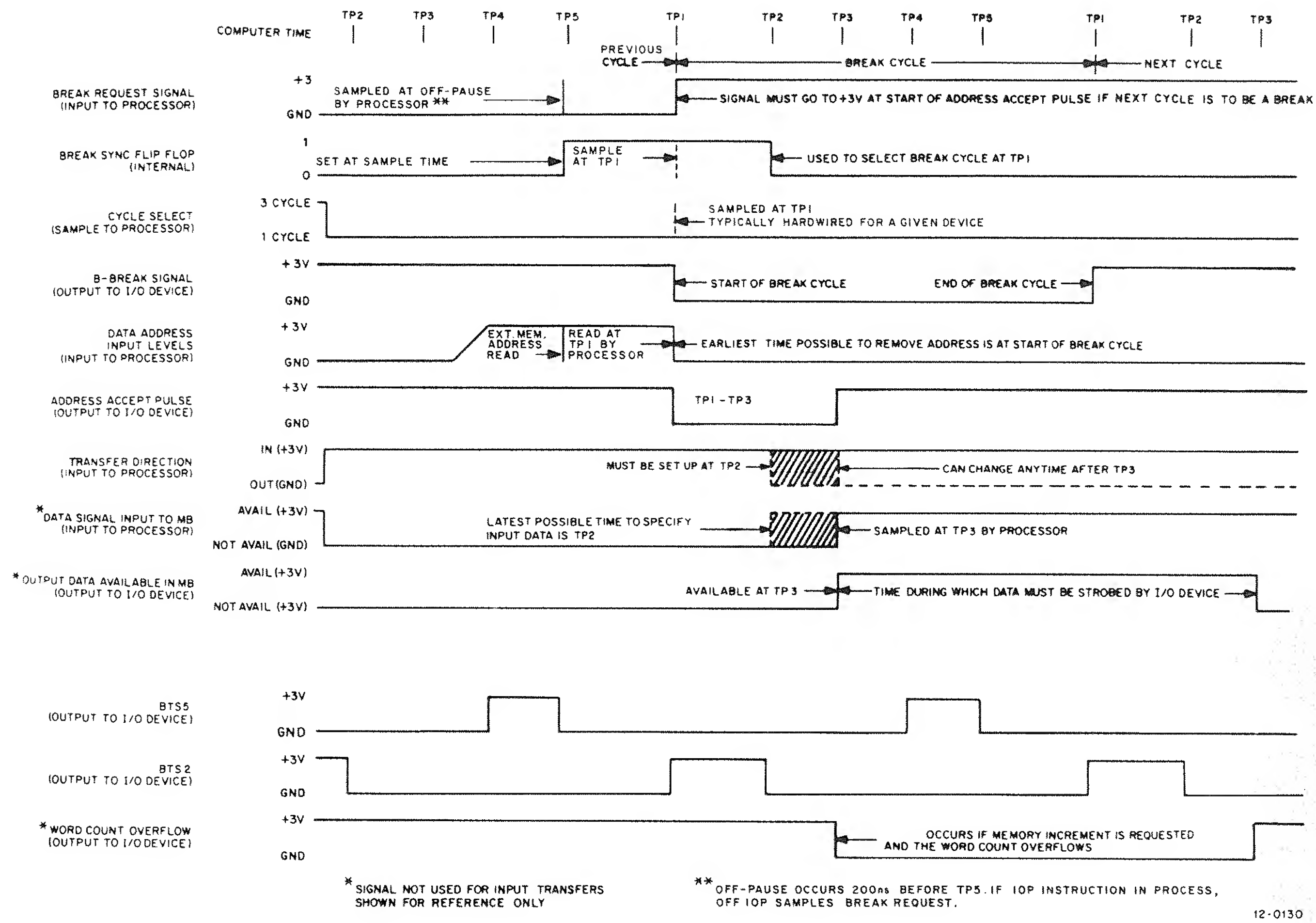


Figure 5-17. Single Cycle Data Break Input Transfer Timing Diagram

5.3.3 Output Data Transfers

Timing of operations occurring in a single-cycle output Data Break is shown in Figure 5-19. Basic logic circuits for the device interface used in this type of transfer are shown in Figure 5-20. 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 5-20) 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.

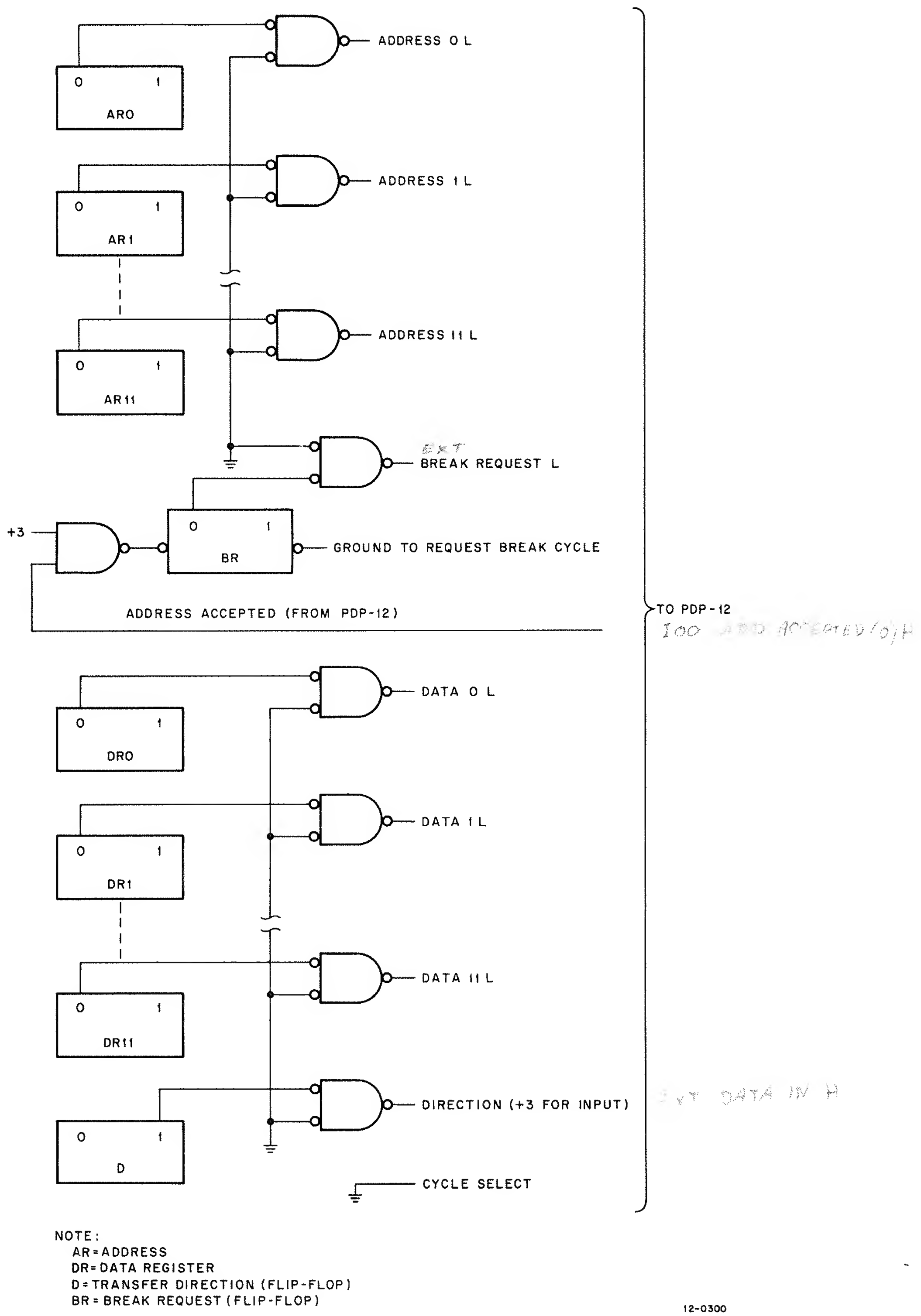


Figure 5-18. Device Interface Logic for Single-Cycle Data Break Input Transfer

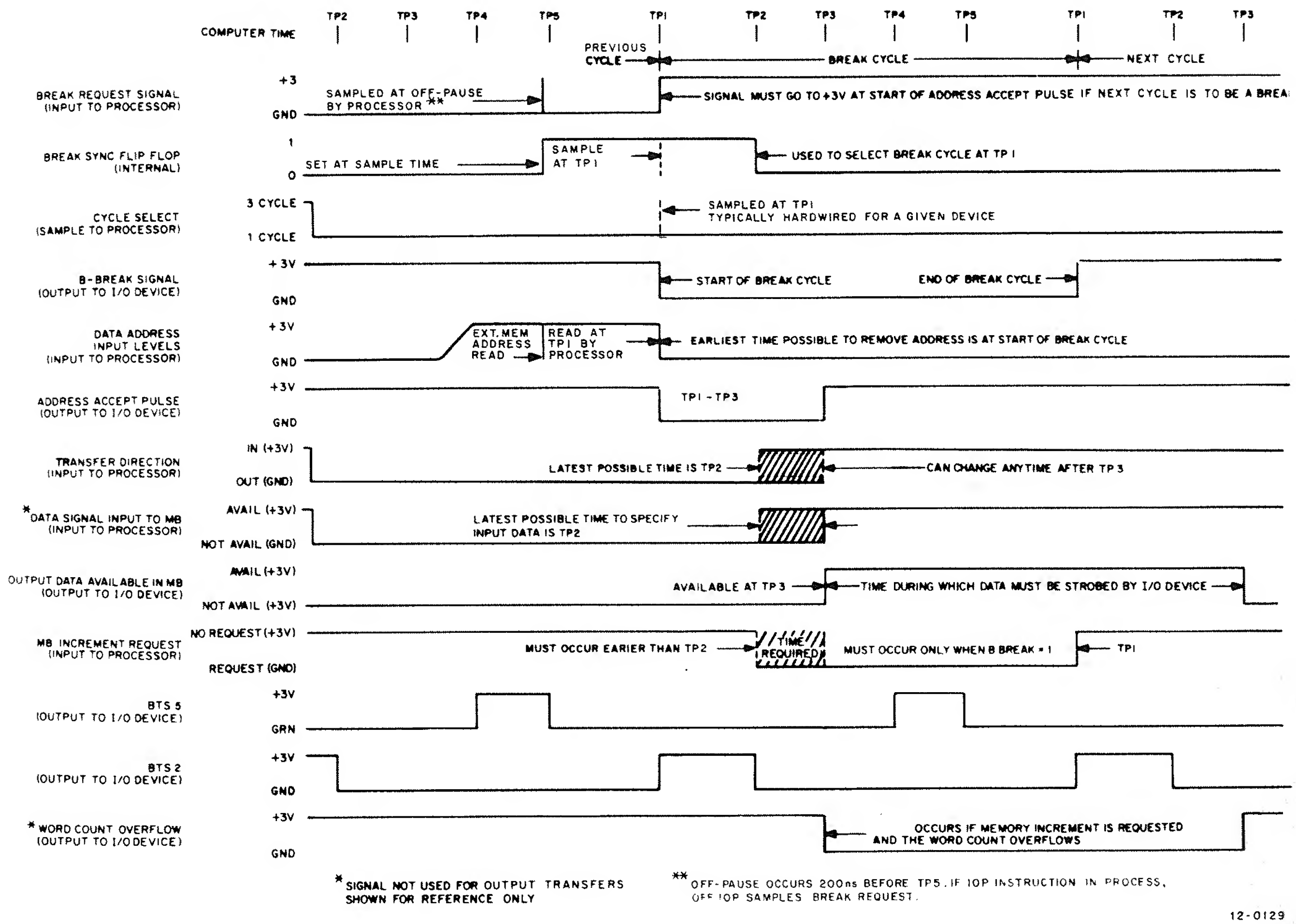


Figure 5-19. Single-Cycle Data Break Output Transfer Timing Diagram

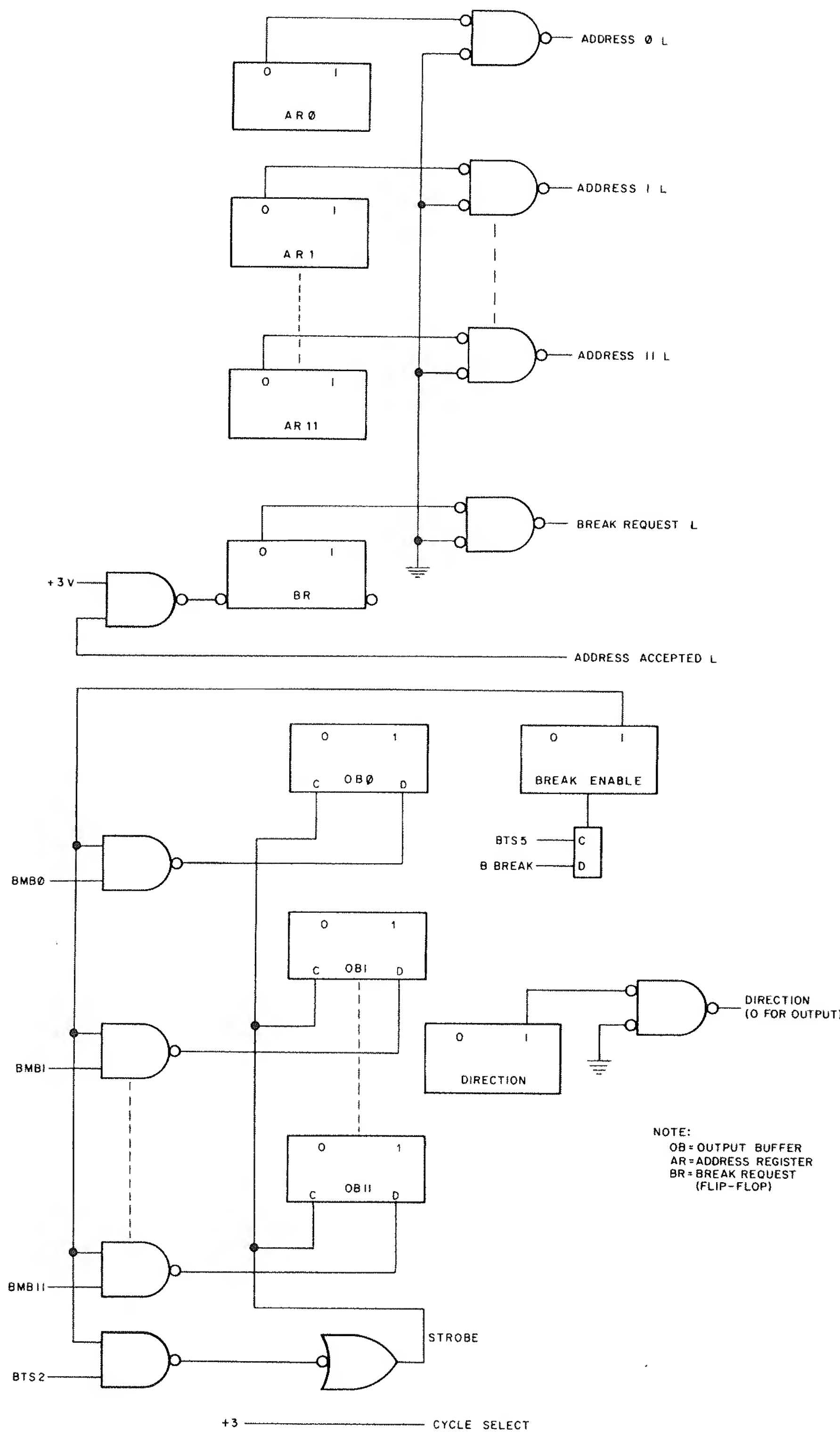


Figure 5-20. Device Interface Logic for Single-Cycle Data Break Output Transfer

When the Break Request is recognized 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 time TP3), the contents of the device-specified core memory address are 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 0s and +3v for binary 1s. The MB is changed at time TP3 of each computer cycle, so the data word is available in the MB for approximately 1.6 microseconds 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 5-20 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 loads 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 an input buffer when the Transfer Direction signal is +3v.

5.3.4 Memory Increment

In this type of Data Break the contents of core memory at a device-specified address are read into the MB, are incremented by 1, and are rewritten at the same address within one 1.6-microsecond 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 I/O device). Timing of memory increment operations appear in Figure 5-21.

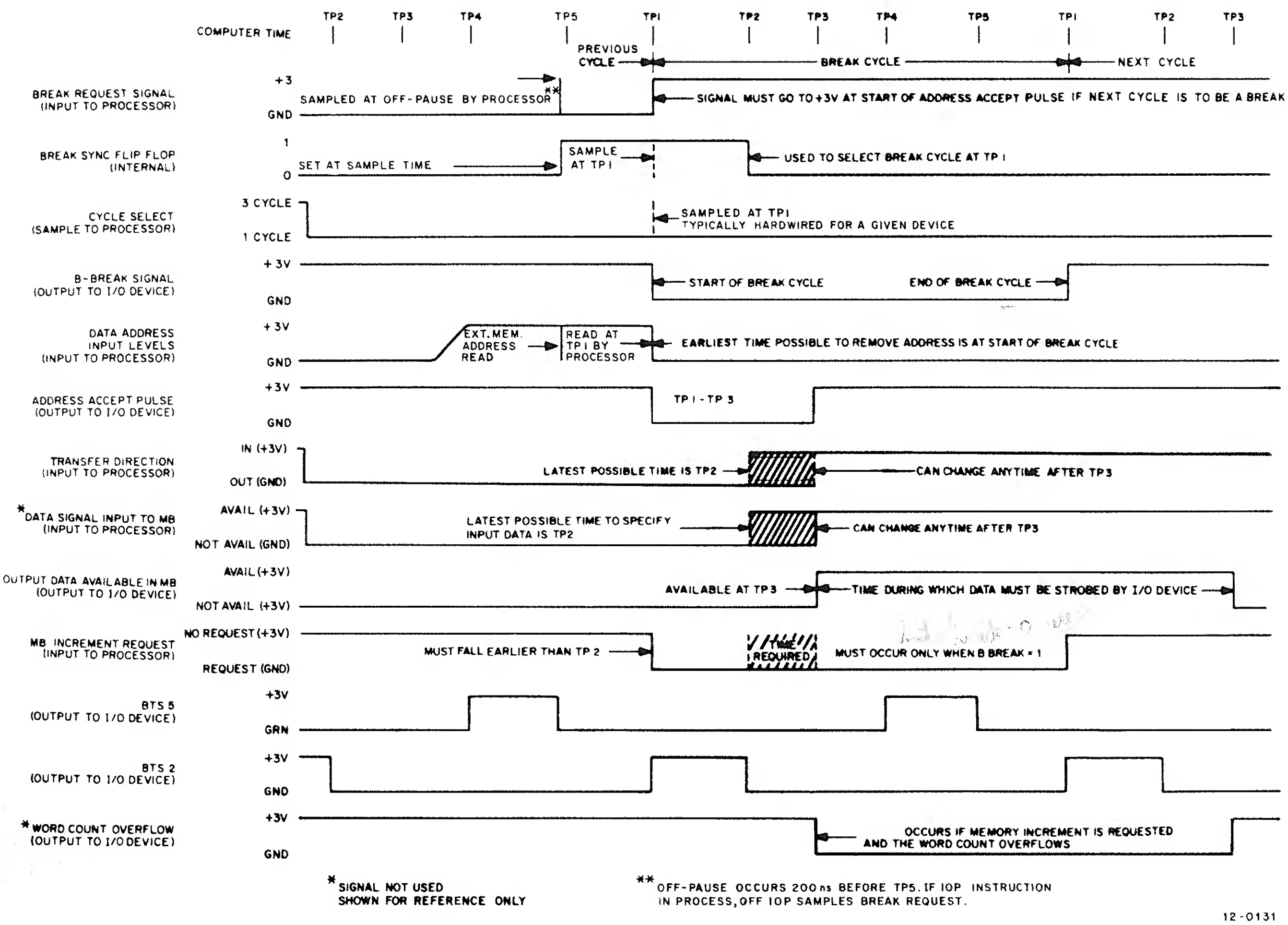


Figure 5-21. Memory Increment Data Break Timing Diagram

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 time TP3 of the Break cycle. The signal can be generated in the device interface by ANDing 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 time TS3 to increment the contents of the MB.

5.3.5 Three-Cycle Data Breaks

Timing of input or output three-cycle Data Breaks is shown in Figure 5-22. The three-cycle Data 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 kHz.

(625 kHz for single cycle)

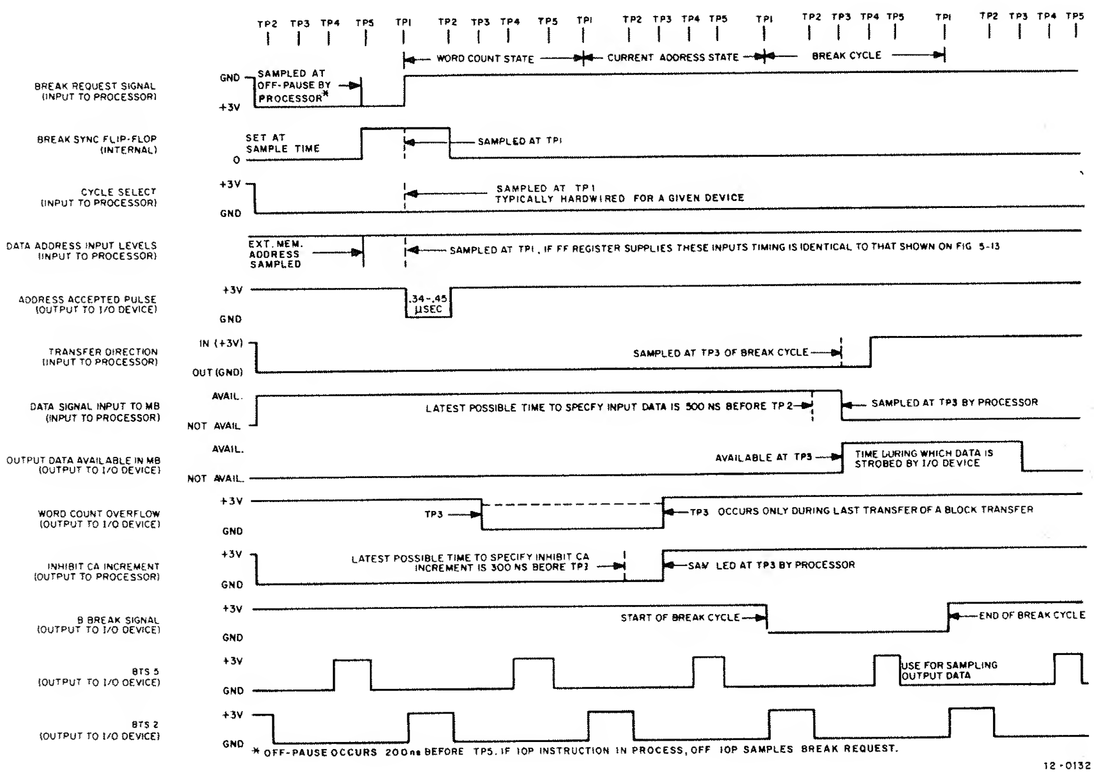


Figure 5-22. Three-Cycle Data Break Timing Diagram

The three-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 kHz. 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 requests to transfer data to or from core memory, the three-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 contents of the specified address are read from memory and 1 is added before rewriting. If the contents of this register become 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 contents of this register are 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 contents of the previously read current address are 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 three-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 three-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 three-cycle data break. These are:

Word Count Overflow – A level change from GND to +3V, from TP3 of the cycle requesting the word count to TP3 of the next cycle 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 three-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 containing the word count. The device requesting the break supplies this address as in the single-cycle Data Break, except that this address is fixed and can be supplied by wired ground and +3V signals, rather than from a register. Following the WC state, a Current Address (CA) 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 Break (B) state is entered to effect the transfer.

5.4 INTERFACE DESIGN AND CONSTRUCTION

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

5.4.1 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, their component circuits, and the associated acessories; 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.

5.4.1.1 **M111/M906 Positive Input Circuit (See Figure 5-23)** – 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.

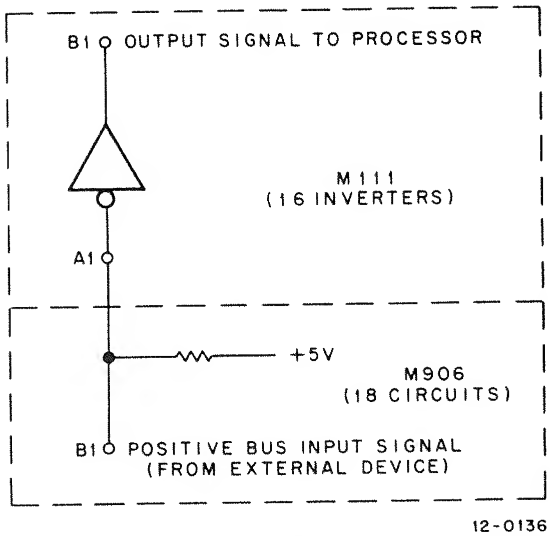


Figure 5-23. Typical M111/M906 Positive Input Circuit

5.4.1.2 **M516 Positive Bus Receiver Input Circuit** (See Figure 5-24) – Six four-input NAND gates with overshoot and undershoot clamp on one input of each gate. Pullup resistors connected to +5V are also provided.

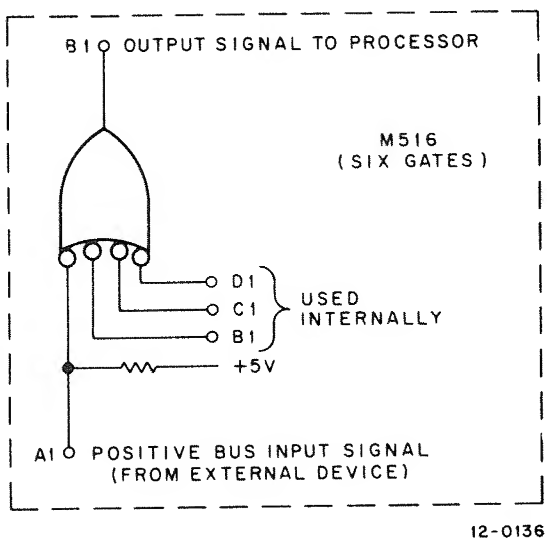


Figure 5-24. Typical M516 Positive Bus Receiver Input Circuit

5.4.1.3 **M623/M906 Positive Output Circuit** (See Figure 5-25) – The M623 Bus Driver module contains twelve circuits with negative NOR gates. 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.

OR

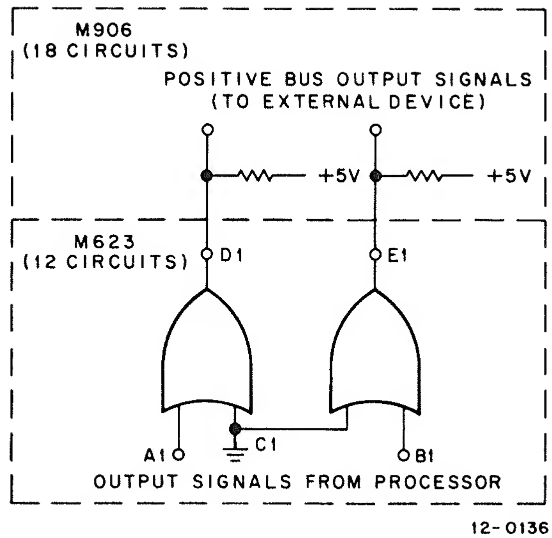


Figure 5-25. Typical M623/M906 Positive Output Circuit

5.4.1.4 **M660 Bus Driver Output Circuit (See Figure 5-26)** – Three circuits which provide low impedance 100-ohm terminated cable driving capability using M Series levels or pulses of duration greater than 100 nanoseconds. The 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, TS 5

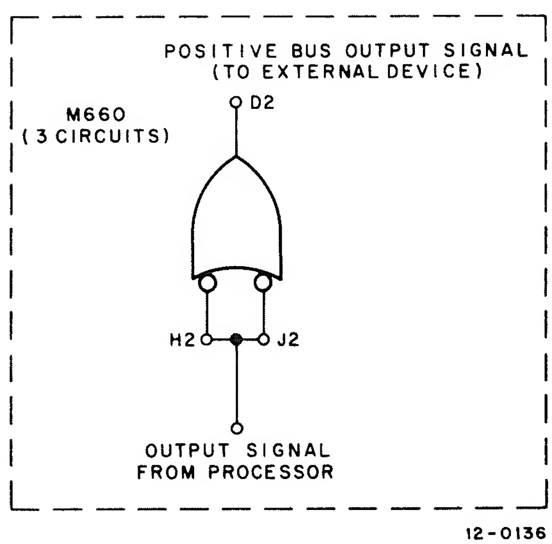


Figure 5-26. M660 Terminated Bus Driver Output Circuit

5.4.1.5 **Module Selection for Interface Circuits of Peripheral Equipment** – Two FLIP-CHIP modules are of particular interest in the design of equipment to interface with the PDP-12. Complete details on these and other FLIP CHIP modules can be found in the Digital Logic Handbook.

5.4.1.6 **M103 Device Selector (See Figure 5-27)** – 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.

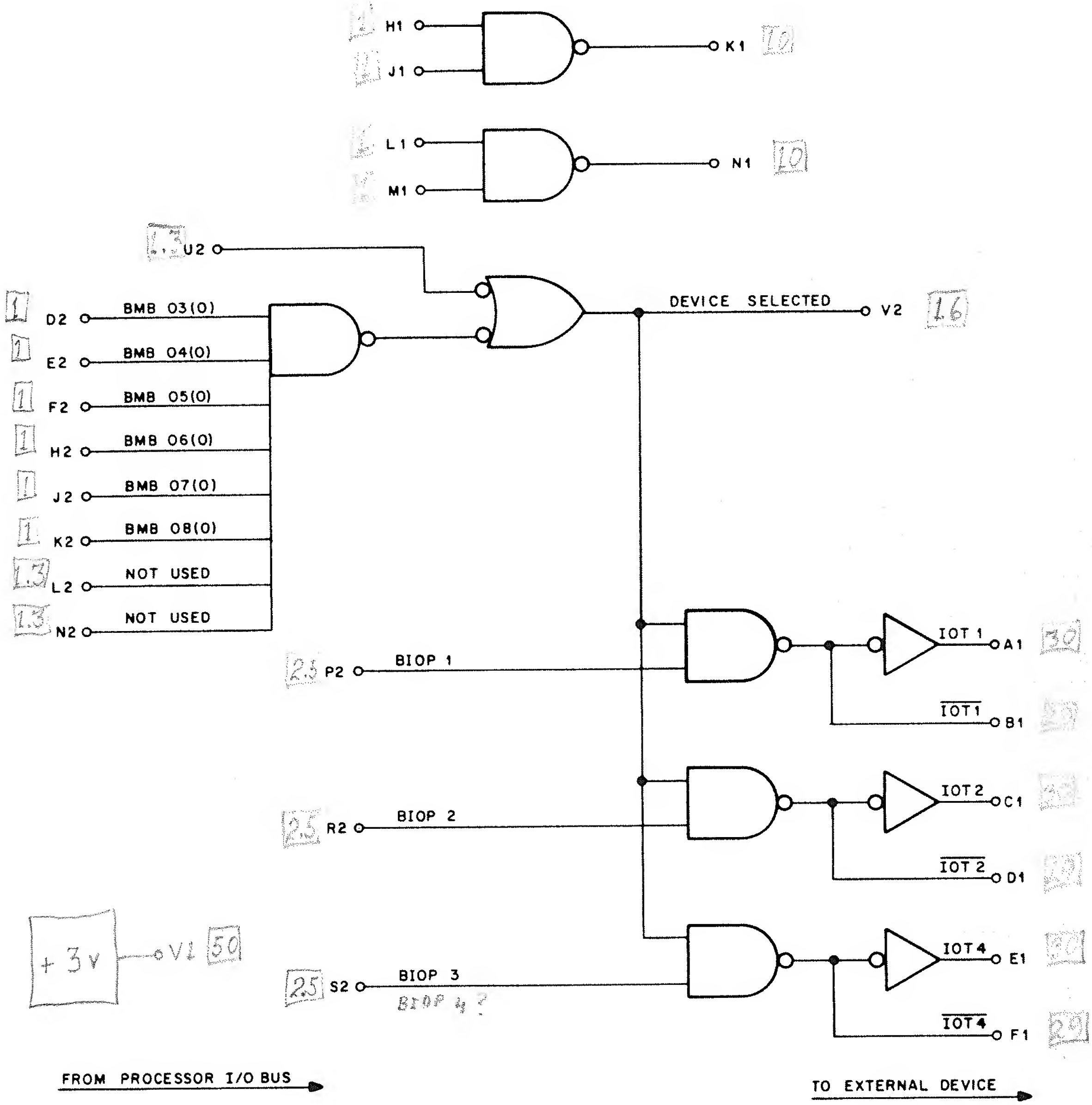


Figure 5-27. M103 Device Selector Logic Circuit

5.4.1.7 **M101 Bus Data Interface (See Figure 5-28)** – 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-12 I/O bus. The M101 module includes diode protection clamps on input lines so that it may be used directly on the PDP-12 positive bus.

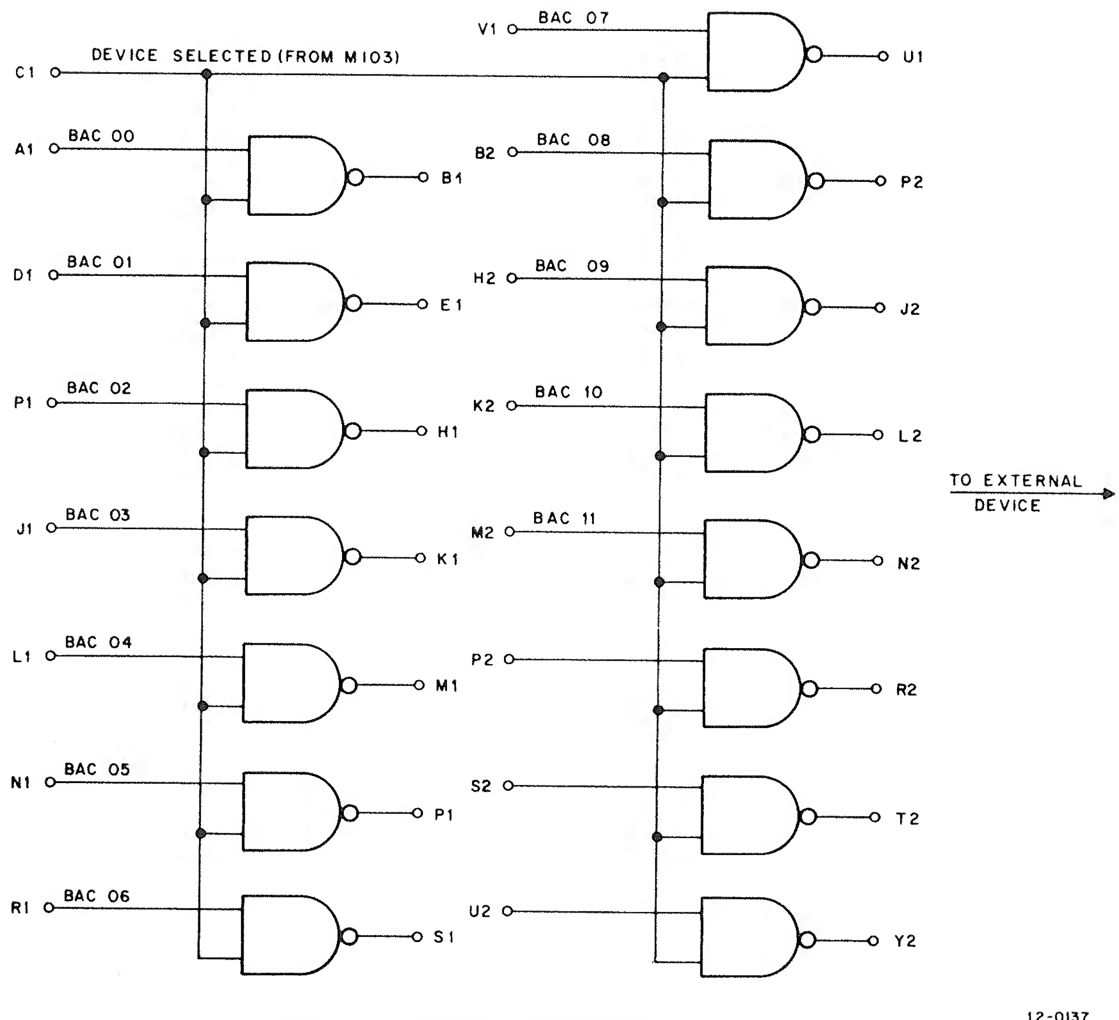


Figure 5-28. M101 Bus Data Interface Logic Circuit

5.4.2 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.

Table 5-2. M Series Module Summary

Type	Function	Description
M002	15 Loads	Fifteen +3 volt sources each capable of driving ten unit loads. Can be used for tying 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	50 ma Indicator and Relay 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, but for use with PDP-8/I and PDP-12 options. Output pulses are not regenerated but only buffered.
M111	Inverter	Sixteen inverter circuits with a fan-in of one unit load and fan-out of ten unit loads.
M112	NOR Gate	Ten positive NOR gates with a fan-in of one unit load and fan-out of ten unit loads.
M113	10 2-Input NAND Gates	Ten two-input positive NAND gates with a fan-in of one unit load and fan-out of ten unit loads.
M115	8 3-Input 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	6 4-Input 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	3 8-Input 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.

Table 5-2. M Series Module Summary (cont)

Type	Function	Description
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	Gate Module	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 M161s. 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 that 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 of one unit load and a fan-out of ten unit loads.
M204	Counter-Buffer	Four J-K flip-flops which can be interconnected as a ripple or synchronous counter or used as general control elements.
M206	Six Flip-Flops	Six D-type flip-flops which can be used in shift registers counters, buffer registers, and general purpose control functions.
M207	Flip-Flops	Six single-input J-K type flip-flops for use in shift register, ripple counters, and general purpose control functions.
M208	Buffer Shift Register	An internally connected 8-bit buffer or shift register. Provisions are made for gated single-ended parallel load, bipolar parallel output, and serial input.

Table 5-2. M Series Module Summary (cont)

Type	Function	Description
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	6-Bit L-R 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	One Shot Delay	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	Clock	A gateable RC clock with both positive and negative pulse outputs. The output frequency is adjustable from 10 MHz to below 100 Hz.
M405	Crystal Clock	Stable system clock frequencies from 5 kHz to 10 mHz are available with this module. Frequency drift at either the positive or negative pulse output is less than 0.01% of the specified frequency.
M410	Reed Clock	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	Variable Clock	Provides square wave output of 880 Hz, 440 Hz, and 220 Hz necessary for clocking the M706 and M707 in a 110-baud teletype system.

Table 5-2. M Series Module Summary (cont)

Type	Function	Description
M501	Schmitt Trigger	Provides 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 -3 volt 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 -3 volt systems to M Series levels. Each of the six circuits on this module provides a low impedance output for driving unterminated long lines.
M507	Bus Converter	Six inverting level shifters which accept -3 and GND, as inputs and have an open collecter NPN transistor at the output. Output rise is delayed by 100 nsec for pulse spreading.
M516	Positive Bus Receiver	Six four-input NOR 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 Generator	The two pulse amplifiers in this module provide standard 50-nsec or 110-nsec pulses for M Series systems.
M617	6-4 Input NOR Buffers	Six four-input positive NOR gates with a fan-in of one unit load and a fan-out of 30 unit loads.
M627	Power Amplifier Module	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 pulses (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	Positive Level 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.

Table 5-2. M Series Module Summary (cont)

Type	Function	Description
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 Connector	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 Terminator	Double-sided 36-pin terminator module with 100 Ω terminations on signal leads. Alternate grounds 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.

5.4.3 Construction of Interfaces

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

5.4.3.1 **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.

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

To implement several devices, the cables parallel-connect each peripheral in a serial type form (see Figure 5-29). 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.

5.4.3.2 **Module Layout** – In general, module layout is 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.

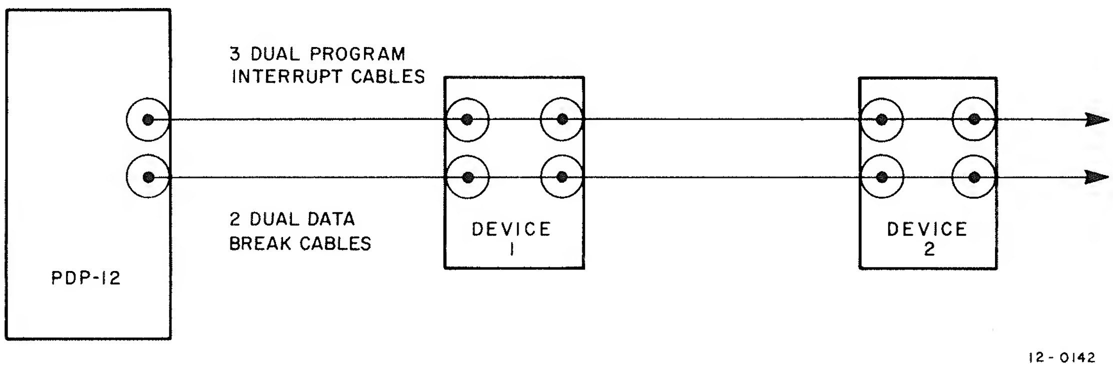


Figure 5-29. I/O Bus Configuration

Figure 5-30 shows the I/O cable connections in an option mounting panel. Module slot locations 1 through 3 (looking at the wiring pin side) 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.

5.4.3.3 **Cable Selection** – Two types of cables are recommended for I/O interface connections.

The first is 9-conductor coaxial 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:

$Z = 95 \pm 5\ \Omega$
$C = 13.75$ pF/foot approx. (unterminated)
$L = 124$ nH/foot approx.
$R = 0.095\ \Omega$/foot nominal
$Y = 79\%$ of velocity of light, approx. (1.5 nsec/ft.)

The second type is a 19-conductor (9 signals and 10 grounds), #30 gauge flat copper Flexprint.

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

5.4.3.4 **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.

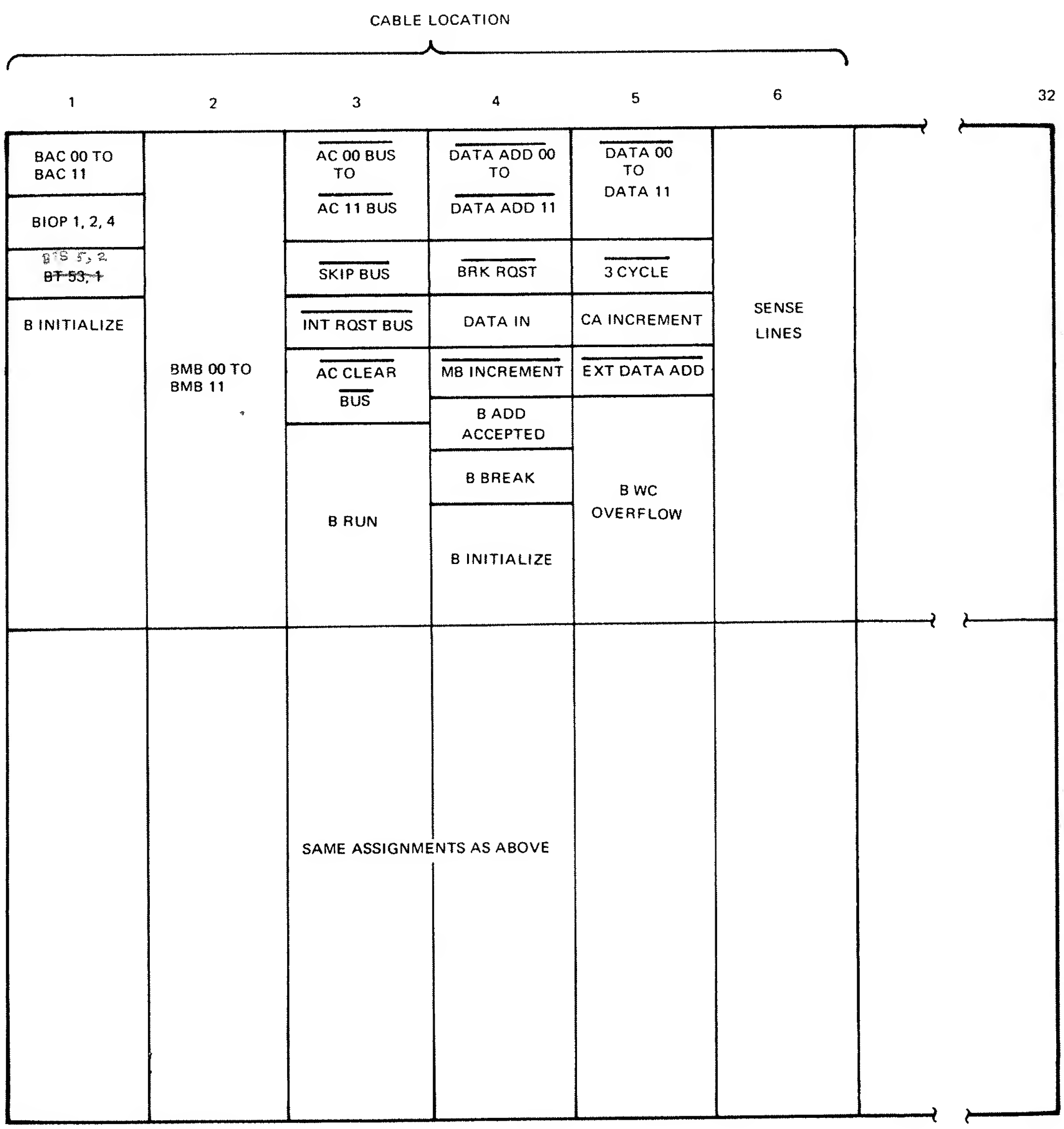


Figure 5-30. I/O Cable Connections

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, A INIT
V2

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 fast, neat 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 interconnect chassis ground with Pins C2 and T1 grounds. Run these wires from the uppermost mounting panel to the bottom panel. On the first and last blocks of the mounting panel, connect 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 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 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 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 must be grounded, except when all modules are removed from the mounting panel. This procedure must be followed because, even with tools isolated from the ac power line, such as those operated by batteries or compressed air, static charges may build to sufficient amplitudes so that damage to semiconductors may result.

Cooling – The low power consumption of M Series modules results in a total of about 15 watts dissipation in a typical H911 mounting panel containing 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.

5.4.4 IOT Allocations

IOT	Option
00	Interrupt (Incl. API)
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~~ API maintenance
06	Displays, Types VC8/I and KV8/I
07	Displays, Types VC8/I, and Light Pen Type 370
10	Power Fail Option KP12
11	Teletype System Type PT08
12	Teletype System Type PT08
13	Real Time Clock Type KW12
14	Mode Change (IOT 6141)
15	Tape Maintenance
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 DP12
41	Teletype System Type DP12
42	Teletype System Type PT08
43	Teletype System Type PT08
44	Teletype System Type PT08
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, AM03A 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 FPP
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~~ RK08 DISK CARTRIDGE
75	
76	DECtape Control TC01 API
77	DECtape Control TC01 API

5.4.5 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 progress alphabetically 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 use the suffix number 1 to designate the top side of a module and the suffix number 2 to designate the bottom side.

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

Receptacle	Use
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. Terminals A2 and B2 are not used. 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 receives the computer connection from the computer itself or the previous device, and one passes 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 +3.6 volts.
sink 1.6 ma. max ... one unit load ... supply 40 µa mly

The following table presents 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., IOO 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).

Table 5-3. Cable Connections to the PDP-12 I/O Bus

Signal	Connection	Signal	Connection
IOB XL 00 H	N13B1	IOB XL 11 H	N13D2
IOB XL 01 H	N13D1	IOB XL 12 H	N13E2
IOB XL 02 H	N13E1	IOB XL 13 H	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 10 H	N13S1	NOT USED Buffered A	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
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

Table 5-3. Cable Connections to the PDP-12 I/O Bus (cont)

Signal	Connection	Signal	Connection
EXT IO BUS 00 L	N16B1	EXT IO BUS 09L	N16D2
EXT IO BUS 01 L	N16D1	EXT IO BUS 10 L	N16E2
EXT IO BUS 02 L	N16E1	EXT IO BUS 11 L	N16H2
EXT IO BUS 03 L	N16H1	EXT SKIP BUS L	N16K2
EXT IO BUS 04 L	N16J1	EXT INT RQST BUS L	N16M2
EXT IO BUS 05 L	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 MEA BRK SYNC CLKB H	N16T2
EXT IO BUS 08L	N16S1	NOT USED EXT ENAB INT PAUSE H	N16V2
EXT DATA ADD 00 L	N17B1	EXT DATA ADD 09 L	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 100 Ω	N17S2
EXT DATA ADD 07 L	N17P1	EXT INCREMENT MB L	N17T2
EXT DATA ADD 08 L	N17S1	IOO BB INITIALIZE H 100 Ω	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 current address	N18K2
EXT DATA 04 L	N18J1	IOB CA INCREMENT H	N18M2
EXT DATA 05 L	N18L1	IOO WC OVERFLOW (0) H 22 Ω	N18P2
EXT DATA 06 L	N18M1	EXT EXTEND DATA ADD 02 L	N18S2
EXT DATA 07 L	N18P1	EXT EXTEND DATA ADD 01 L	N18T2
EXT DATA 08 L	N18S1	EXT EXTEND DATA ADD 00 L	N18V2

This is an external signal. IOB only has [illegible]

CHAPTER 6
PERIPHERAL DEVICES

INTRODUCTION

This chapter contains descriptions of all the standard prewired I/O bus options which are available with the PDP-12. It describes the peripheral logic expander, BA12, and options contained within the panel, as well as the most commonly used PDP-8 and PDP-12 family of I/O bus options. In general, most PDP-8 family of options can be operated without modification on the PDP-12 I/O bus. The reader, therefore, should refer to the DEC Small Computer Handbook (1970) for additional information.

Prewired options and options contained in the BA12 Peripheral Expander panel derive their power from the PDP-12 power supply. The Peripheral Expander contains the necessary buffering to provide the isolation and current driving requirements for I/O devices. The control logic for these options is contained in plug-in modules; therefore, when one of these options is added, wiring changes or additions are not needed. Separate power supplies are normally included with all the other options.

Option Groupings	PG	Option Type Number
a. Prewired		
Teletype Model 33 ASR	6-3	
Additional Teletype or Dataphone	6-9	DP12-A,B
Real-Time Interface	6-18	KW12-A
Fixed Interval Clocks	6-25	KW12-B,C
LINCtape to DECtape format converter		TC12-F
Digital Plotter and Control	6-68	XY12
Power Fail	6-74	KP12
b. Peripheral Expander Type BA12		
4-Station TTY Control		DC02-D,E
Line Printer	6-59	LP12
High-speed Paper Tape Reader/Punch	6-71	PC12, PP12, and PR12
Standard or Mark Sense Card Reader	6-63, 6-66	CR12, CM12
Data Buffers	6-73	DB12-P, N

Option Groupings		Option Type Number
c. I/O Bus Stand Alone Peripherals		
32 Station TTY Control		DC02-F,G
1- and 2-Station TTY Control		PT08
Line Printer		LP08
Magnetic Tape Control	-45	TC58
Magnetic Tape Transport		TU20
Disk Fixed Head, 32K	6-27	DF32, DS32
Disk Fixed Head, 256K	6-30	RF08, RS08
Disk Movable Head 800K	6-38	RK8, RK01
A-D Converter		AF01A
D-A Converter		AA01A

The above groupings represent the physical organization of the PDP-12; however, the options will be described in the following order:

6.1 TELETYPE

6.1.1 Model 33 ASR

The Teletype Model 33 ASR is the standard Teletype device offered with the PDP-12. It may 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 ten characters per second. Signals transferred between the Model 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 current in the Teletype, and to zeros and ones in the teletype 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 occupies two units. 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). Bits are numbered from right to left, from 1 through 8, with bit 1 having the least significance.

Figure 6-1 illustrates the relationship between paper tape information and the AC.

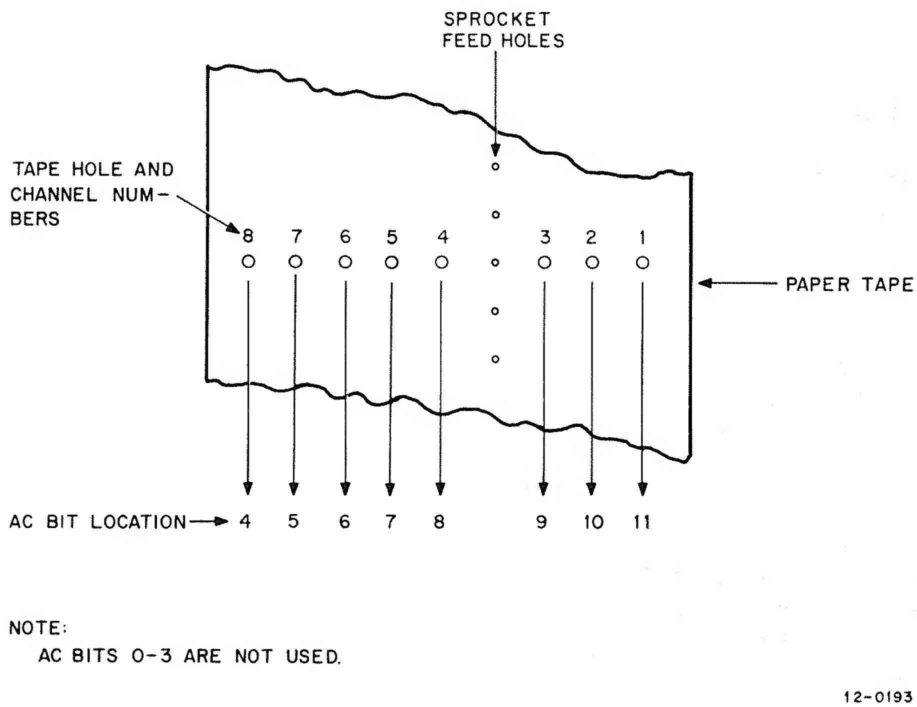


Figure 6-1. Relationship Between Paper Tape and Accumulator

The character (number) four (4) as it would be punched on paper tape is shown in Figure 6-2.

The Model 33 ASR set generates 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 detects 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 F lists the character codes for the Teletype.

6.1.2 Model 33 KSR

This Teletype model is similar to the 33 ASR, except that it does not have either a paper tape reader or punch. The control logic, however, is the same as that used with the 33 ASR.

SPROCKET FEED HOLE

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	

12-0193

Figure 6-2. Punched Paper Tape Format for the Number 4

6.1.3 **Model 35 KSR**

This unit is functionally the same as the 33 KSR. It is designed for heavy duty use and extended reliability. The control logic, however, is the same as that used in the 33 ASR.

6.1.4 **Model 37 KSR**

This Teletype is offered as part of the LT37-AD, AE (50 Hz) option. It has an expanded character set (i.e., upper and lower case) and control functions, and operates at 15 characters per second, both transmitting and receiving. The LT37-AD option has a front panel switch which can effectively convert the unit to operate as a 33 ASR. Only upper case characters would then be received or transmitted. The LT37 option provides the following programmed operations:

a. Horizontal tab set and clear
b. Motor control; on and off
c. Vertical tab; set and clear at full line increments
d. Ribbon color shift
e. Reverse linefeed; full and half line increments

The LT37 is useful for the preparation of formal reports, business forms and graphical plots.

Table F-3 in Appendix F provides the character and control codes for each mode of operation.

6.1.5 **Teletype Controls**

The basic programmed operation of the following devices is similar:

Console Teletype
Prewired Dataphone Option, Type DP12-A, B
Add-on Single and Dual TTY Control, Type PT08-B, C

They all transmit and receive asynchronous, full-duplex, bit-word information. These devices use control modules M706 (Receiver) and M707 (Transmitter), which are positive logic modules, or W706 (Receiver) and W707

(Transmitter), which are the negative equivalent logic control modules. These modules are fully described in the Digital Logic Handbook. The main differences in these options being controlled are:

a. Source of data (e.g., Dataphone, keyboard/display terminal, Teletype, etc.)

b. Speed of operation
Slow speed devices such as Teletype are driven by the stabilized RC oscillator clock module (M452). Higher speed requires a high stability of the selected frequency; therefore, a crystal-controlled clock module (M405) is used. The frequency of operation must be specified for each separate device.

c. Voltage level of inputs
Typically, DEC equipment will interface directly with EIA RS-232-B industry standard devices or the standard 0, 20 mA Teletype current loop (sometimes referred to as 0, +3V logic level). As the console Teletype is typical of the three different controls discussed in this section, it will be described in detail. The differences which are noteworthy in the PT08 and DP12 will be further discussed.

6.1.5.1 **PDP-12 Console Teletype Control** – The Teletype control uses the standard M706 receiver, M707 transmitter, and M452 clock modules for basic logic. It will drive any one of the previously discussed Teletype models (the KSR-37 requires a slight adjustment of the clock to operate at 15 characters per second).

Serial information read or written by the Teletype unit is assembled or disassembled by the Teletype control for parallel transfer to the accumulator (AC). The control also provides the program flags that cause a program interrupt or an instruction skip depending on the availability of the Teletype and the processor.

In all programmed operation, the Teletype unit and control are considered as a Teletype in (TTI) for input data from the keyboard or the perforated-tape reader, and as a Teletype out (TTO) for computer output information to be printed and/or punched on tape. Therefore, two device select codes are used. Select code 03 initiates operations associated with the keyboard/reader (TTI) and select code 04 performs operations associated with the teleprinter/punch (TTO). Parallel input and output functions are performed by corresponding IOT pulses produced by the two device selectors. Pulses produced by the IOP1 pulse trigger skip gates; pulses produced by the IOP2 pulse clear the control flags and/or the accumulator; and pulses produced by IOP4 initiate data transfers to and from the control.

6.1.5.2 **Keyboard Reader** – The keyboard and tape reader control contains an 8-bit shift register (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 register TTI. The code of a Teletype character is loaded into the TTI so that spaces correspond to binary zeros and holes (marks) correspond to binary ones. Upon program command, the contents of the TTI are 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 is set 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 instruction, and, if the flag is set, issue a KRB instruction which clears the AC, clears the keyboard flag, transfers the contents of the TTI into the AC, and enables advance of the tape feed mechanism. Program interrupt can be controlled by the LINC mode instruction ESF (0004) (refer to Paragraph 3.3.16). This instruction either enables or inhibits interrupts when either the TTI or TTO flag is set.

6.1.5.3 **Instructions** – Instructions for use in supplying data to the computer from the Teletype are as follows:

KSF Skip on Keyboard Flag

Octal code:	6031
Event time:	1
Execution time:	4.25 μs
Operation:	If the keyboard flag is set, the contents of the PC are 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 μs
Operation:	The AC and the keyboard flag are cleared. If there is tape in the reader and the reader is on, the character over the read head is loaded into the TTI and the tape advanced one frame. If there is no tape or the reader is off (STOP or FREE) the character struck on the keyboard is assembled into the TTI. In either case, when the character is completely assembled in the TTI, the hardware sets the keyboard flag.
Symbol:	0 → AC 0 → Keyboard flag allowing the hardware to cause: Keyboard/Tape Character → TTI 1 → Keyboard flag (approximately 100 ms after issuing the instruction)

KRS Read Keyboard Buffer Static

Octal code:	6034
Event time:	3
Execution time:	4.25 μs
Operation:	The contents of the TTI are transferred into AC_{4-11}. This is a static command in that neither the AC nor the keyboard flag is cleared. KRS can be microprogrammed with KCC.
Symbol:	TTI V $AC_{4-11} \rightarrow AC_{4-11}$ 0 → Keyboard Flag

NOTE

The KRS instruction has been redefined in a later version of the PDP-12 to permit clearing of the TTI flag without advancing the paper tape and assembling the next character.

KRB Read Keyboard Buffer Dynamic

Octal code:	6036
Event time:	2, 3
Execution Time:	4.25 μs
Operation:	This instruction combines the functions of the KCC and KRS. The AC and keyboard flag are both cleared and the contents of the TTI are transferred into AC_{4-11}. Clearing the keyboard flag allows the hardware to begin assembling the next input character into the TTI (as described for KCC). When the character is completely assembled in the TTI, the hardware causes the flag to be set, indicating that TTI again has a character ready for transfer.
Symbol:	0 → AC C(TTI) V $C(AC_{4-11}) \rightarrow AC_{4-11}$ 0 → Keyboard Flag allowing the hardware to cause: Tape Reader to advance 1 character Keyboard/Tape Character → TTI 1 → Keyboard Flag when down (approximately 100 ms after issuing instruction)

KST Key Struck (LINC mode)

Form:	KST I
Octal code:	0415 + 20I
Execution time:	1.6 μs
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.)

The program example shown below will read 1 character from the keyboard.

```
*200
INPUT,      KCC             /CLEAR KEYBOARD FLAG
            JMS LISN
            DCA STORE
            HLT
LISN,       0               /SKIP ON KEYBOARD FLAG
            KSF
            JMP.-1
            KRB             /READ KEYBOARD BUFFER
            JMP I LISN
STORE,      0
$
```

The main program begins with KCC. In general, the main program should begin by clearing the flags of all devices to be used later in the program. If the above program is started at location 200, it will proceed to the KSF, JMP.–1 loop, and stay in this loop endlessly until a key on the Teletype unit is pressed or a paper tape is loaded into the reader. When the ASCII code for the character is assembled in the keyboard/reader buffer register, the flag will be set to a 1 and the program will skip out of the loop. The contents of the buffer will be transferred into the accumulator, and the buffer and flag will be cleared.

6.1.5.4 **Teleprinter/Punch** – On program command, a character is transferred from the accumulator (AC) to the output shift register (TTO) for transmission to the teleprinter/punch unit. The teleprinter control generates the start space, shifts the eight character bits serially into the printer selector magnets of the teletype unit, and then generates two stop marks. Bit transfer rate from the TTO to the teleprinter/punch unit is at the normal Teletype rate of 110 baud. A character transfer requires 100 milliseconds for completion. The teleprinter flag is set when the last bit of the character code is sent to the teleprinter/punch, indicating that the TTO is ready to receive a new character from the AC. The flag activates either the program interrupt synchronization element or the instruction skip element. When using instruction skip, the program checks the flag by means of TSF instruction. If the flag is set, the program must issue a TLS instruction which clears the flag and sends a new character from the AC to the TTO. AC to TTO transfer time is short compared to the print/punch time, so the program must wait for the flag to set before issuing another TLS. Instructions for use in outputting data to the teleprinter/punch are as follows:

TSF Skip on Teleprinter Flag

Octal code:	6041
Event time:	1
Execution time:	4.25 μs
Operation:	If the teleprinter flag is set, the contents of the PC are 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 μs
Operation:	The teleprinter flag is cleared. Can be microprogrammed with TPC.
Symbol:	0 → Teleprinter Flag

TPC Load Teleprinter and Print

Octal Code:	6044
Event time:	3
Execution time:	4.25 μs
Operation:	AC_{4-11} are parallel transferred to the TTO, then the hardware starts shifting the character out to the printer/punch unit. When the transfer is complete (approximately 100 ns), the TTO flag is set.
Symbol:	$C(AC_{4-11})$ → TTO causing: C(TTO) → printed and (if punch is on) punched

TLS Load Teleprinter Sequence

Octal Code:	6046
Event time:	2, 3
Execution time:	4.25 μs
Operation:	This is an instruction that combines TCF and TPC. The teleprinter flag is cleared, then the contents of AC_{4-11} are parallel transferred to the TTO, where the hardware serially shifts the character-bits out to the printer/punch unit. When the printer/punch has finished outputting the character, the hardware sets the teleprinter flag. The whole operation, from the time TLS clears the flag and TPC starts character transfer until the time the hardware finishes with the character and again sets the flag, requires 100 ms.
Symbol:	0 → Teleprinter flag I doubt it $C(AC_{4-11})$ → TTO causing: C(TTO) → Printed and (if punch on) punched 1 → Teleprinter flag when done (approximately 100 ns after issuing instruction)

Shown below are several programming examples illustrating the use of the TTO control.

```
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 100 ms spent waiting for the character to be output and the flag to be set is used to continue the main program, making more efficient use of program time.

```
TYPE,          0
               TSF              /TEST FLAG TO SEE IF TELEPRINTER FREE, SKIP IF NOT
               JMP .-1          /WAIT TILL IT IS BY TESTING AGAIN AND AGAIN
               TLS              /OUTPUT CHARACTER
               CLA CLL          /CLEAR CHARACTER FROM AC
               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 continues to run the user's program instead of waiting while the teleprinter (a much slower device) is off typing/punching the last character. The user must initialize the control by setting the teleprinter flag to a one. Otherwise, the subroutine will "hang-up" the first time through (in the TSF, JMP .-1 loop). The initialization can be accomplished by issuing a TLS or TPC instruction at the beginning of the mainline program.

Format Routines

Input and output routines are very often written in the form of subroutines, like the TYPE subroutine in the previous example. The example below is a carriage return/line feed subroutine that calls the TYPE subroutine to execute a carriage return and line feed on the printer, thus advancing to a new line for the printing of information.

Carriage Return/Line Feed Subroutine:

```
CRLF,          0           CLA
               TAD K212
               JMS TYPE        WRONG ORDER
               TAD K215
               JMS TYPE
               JMP I CRLF
K212,          212              /ASCII FOR CARRIAGE RETURN
K215,          215              /ASCII CODE FOR A LINE FEED
TYPE,          0
               TSF
               JMP .-1
               TLS
               CLA CLL
               JMP I TYPE
```

Subroutines similar to the one above could be written to tab space the carriage a given number of spaces, or to ring the bell of Teletype Model 33 ASR by using the respective codes for these nonprinting control characters.

6.1.5.5 **Single Teletype Control, Type DP12-A (prewired)** – This internal option provides an interface which is programmed similar to the standard console Teletype. The device select codes are IOTs 40 and 41. Interrupts caused by either the transmitter or receiver flags can be disabled by the LINC mode instruction ESF (refer to Paragraph 3.3.16).

6.1.5.6 **Dataphone Control, Type DP12-B** – This option is a modification of the DP12-A, which permits communication to most standard Dataphone sets. An extra crystal-controlled clock permits the user to specify a baud rate from 110 to 100K baud in order to provide the necessary frequency stability required at higher baud rates. In addition, the option is supplied with a 25-ft cable, Type BC01A-25, which will connect (via a 25-pin connector) to a Dataphone set. This cable has a card connector (M850), which converts the EIA standard RS-232-B signals to DEC logic levels of 0 and +3V

6.1.5.7 **Single and Dual TTY Control Type PT08** – The PT08 is a serial-to-parallel, parallel-to-serial converter which provides full-duplex communication between an asynchronous channel and a PDP-12 computer. Two basic configurations are offered: PT08-B (one full-duplex channel) and PT08-C (two full-duplex channels). Systems may be expanded up to five duplex channels by stacking PT08 units.

The PT08-B and C are designed to supply transmit and receive keying current that is intended for use with 20 mA, dc-keyed devices. Digital Equipment Corporation's Model 33 or 35 teleprinter units have been modified to be compatible with the PT08. Devices equivalent to the modified teleprinter units are also compatible with the PT08.

The PT08-B and C are negative bus options (0, -3V) and, therefore, a DW08-A negative-to-positive I/O bus converter must be used when connecting the PT08 to the PDP-12.

The PT08-F option provides EIA standard RS-232-B level conversion as well as a 25-ft cable designed to connect to a Dataphone set. Another option, the PT08-X, can be installed in any channel for customer selection of character format and speed. With the PT08-F and PT08-X options combined, the bit rate can be increased to 100K baud for driving medium-to-high-speed asynchronous modems. This combination can be used for an economical intercomputer communication channel, or for interfacing to special equipment with unique asynchronous speeds and character formats.

Specifications

Performance specifications are summarized in Table 6-1.

Table 6-1. PT08 Specifications

	Specifications	
Speed	110 baud is standard; up to 100K (software limited) with PT08-X option.	
Character Format	Standard:	1-unit start; 8 character bits; 2-unit stop.
	PT08-X Option:	5 or 8 character bits, 1- or 1.5-unit stop element at user's request.
Operating Mode	Full duplex.	
Interface	Standard:	Supplies transmit and receive keying current that is intended for use with 20 mA, dc-keyed devices.
	PT08-F Options:	Provides interface that conforms to EIA RS-232-B devices.
Transmission Distance	1500-ft maximum (environment dependent) for local terminals.	
	EIA interface transmission distance is limited only by characteristics of modem and associated communication facility. A 25-ft cable to the modem is supplied.	

Figure 6-3 illustrates the various PT08 equipment configurations for both the standard system expansion and interface provisions.

Programming

IOT instructions test for character-ready conditions and transfer assembled characters to and from the computer's accumulator. The same basic commands are used for all channels, with individual channels assigned different device selection codes. For PT08 channel 1, the devices codes are 40 and 41, etc. (See Table 6-2 for complete listings of PT08 device codes.) It will be noted that channel 1 IOTs are the same as assigned to the internal prewired option DP12-A or B. When channel 1 is implemented, the IOT for channel 1 must be revised to an unused device select code. This normally would be one of the device codes 30 through 37. The basic mnemonic plus the PT-number designator identifies the mnemonic for the specific channel.

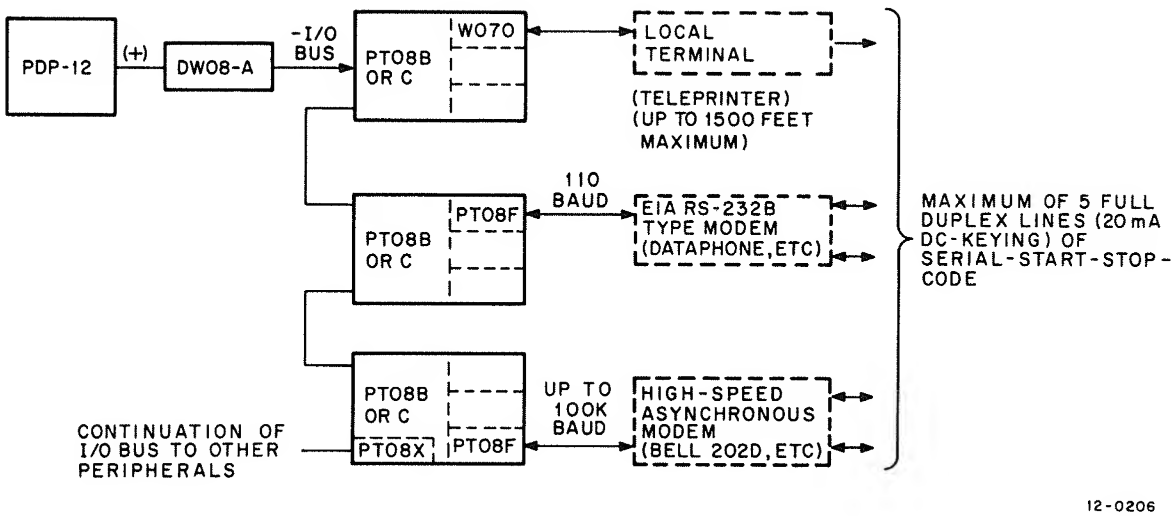


Figure 6-3. PT08 Equipment Configurations

Table 6-2. PT08 Device Codes

Basic Mnemonic	Channel Number				
	1	2	3	4	5
KSF	6421 KSFPT2	6441 KSFPT3	6461 KSFPT4	6111 KSFPT5	6301 KSFPT1
KCC	6422 KCCPT2	6442 KCCPT3	6462 KCCPT4	6112 KCCPT5	6302 KCCPT1
KRS	6424 KRSPT2	6444 KRSPT3	6464 KRSPT4	6114 KRSPT5	6304 KRSPT1
KRB	6426 KRBPT2	6446 KRBPT3	6466 KRBPT4	6116 KRBPT5	6306 KRBPT1
TSF	6431 TSFPT2	6451 TSFPT3	6471 TSFPT4	6121 TSFPT5	6311 TSFPT1
TCF	6432 TCFPT2	6452 TCFPT3	6472 TCFPT4	6122 TCFPT5	6312 TCFPT1
TPC	6434 TPCPT2	6454 TPCPT3	6474 TPCPT4	6124 TPCPT5	6314 TPCPT1
TLS	6436 TLSPT2	6456 TLSPT3	6476 TLSPT4	6126 TLSPT5	6316 TLSPT1

Instructions

The following instructions are used with the PT08:

KSF Skip on Receive Flag

Event time:	1
Indicators:	IOT, Fetch, Pause
Execution time:	4.25 μs
Operation:	Causes the program to skip the next instruction if the receive flag is set, indicating that an assembled character is ready.
Symbol:	If receive flag = 1, PC + 1 → PC

KCC Clear Receive Flag and AC

Event time: 2
Indicators: IOT, Fetch, Pause
Execution time: 4.25 μs
Operation: Clears the accumulator and the receive flag.
Symbol: $0 \rightarrow AC, 0 \rightarrow RF$

KRS Read Receive Buffer (Static)

Event time: 3
Indicators: IOT, Fetch, Pause
Execution time: 4.25 μs
Operation Transfers an assembled character from the receive buffer to $AC_{4\text{-}11}$. Does not reset AC or receive flag.
Symbol: $RB \rightarrow AC_{4\text{-}11}$

KRB Read Receive Buffer (Dynamic)

Event time: 2, 3
Indicators: IOT, Fetch Pause
Execution time: 4.25 μs
Operation: Performs the functions of KCC and KRS together, so that the receive flag and AC are cleared before data is transferred from the receive buffer to the AC.
Symbol: $0 \rightarrow AC, 0 \rightarrow RF$
$RB \rightarrow AC_{4\text{-}11}$

TSF Skip on Transmit Flag

Event time: 1
Indicators: IOT, Fetch, Pause
Execution time: 4.25 μs
Operation: Causes the program to skip the next instruction if the transmit flag is set, indicating that the transmit flag = 1, PC + 1 $\rightarrow$ PC

TCF Clear Transmit Flag

Event time: 2
Indicators: IOT, Fetch, Pause
Execution time: 4.25 μs
Operation: Resets the transmit flag.
Symbol: $0 \rightarrow TF$

TPC Load Transmit Character

Event time: 3
Indicators: IOT, Fetch, Pause
Execution time: 4.25 μs
Operation: Loads the transmit buffer from $AC_{4\text{-}11}$ and initiates transmission of a character.
Symbol: $AC_{4\text{-}11} \rightarrow TB$

TLS Load Transmit Sequence

Event time:	2, 3
Indicators:	IOT, Fetch, Pause
Execution time:	4.25 μs
Operation:	Performs the functions of TCF and TPC together.
Symbol:	$0 \rightarrow TF$
	$AC_{4-11} \rightarrow TB$

Maximum Data Rates

In transmitting the PT08 provides a full character cycle for the program to deliver new data. In receiving, one bit time is required to read in necessary data. However, for maximum data transfer rates, the time at which data transfer can occur is limited to an aperture equal to the stop bit time plus half a bit time. This response time is measured from the beginning of a stop bit (the time at which the transmit or receive flag is reset), and the midpoint of the next character's start bit. If the program fails to respond within this time, a character is lost. Timing is illustrated in Figure 6-4.

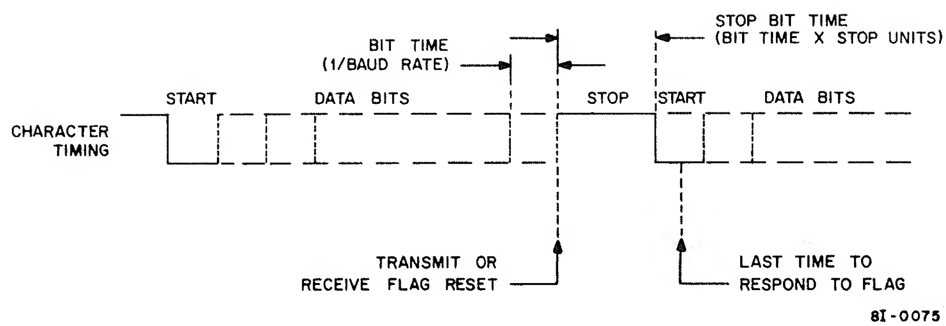


Figure 6-4. PT08 Program Response Time

For example, at 110 baud (9.09 ms bit time), response time is:

Stop bit time + half a data bit time

$= 2 \times 9.09 + 9.09/2$

$= 22.725$ ms

Note that the number of bits per character need not be considered.

6.1.5.8 **Multiple Teletype Control, Type DC02-E (see Table 6-3)** – The DC02-E option, which is prewired in the BA12 Peripheral Expander, allows the user to add up to four serial-to-parallel, parallel-to-serial asynchronous data channels. It consists of a DC02-E multiple station control and from one to four DC02-D station interfaces (each full-duplex). A Type BC01A-25 cable is available that will connect to most Dataphones via a 25-pin connector and will convert EIA standard RS-232-B signals to DEC logic levels of 0 + 3 volts.

The control will handle teletypes at their normal baud rate of 110. In addition, higher speed devices may be operated by using crystal clocks. In this case the baud rate must be specified by the user. Up to two clocks may be used and may be connected by a jumper module in any combination to control the stations' transmitters and receivers.

Table 6-3. DC02-E Specifications

Characteristics	Specifications
Speed	110 baud is standard; up to 100K baud crystal controlled.
Character Format	Standard: 1-unit start; 8 character bits; 2-unit stop.
	Option: 5 or 8 character bits 1- or 1.5-unit stop element at user's request.
Operating Mode	Full duplex.
Interface	Standard: Supplies transmit and receiver keying current that is intended for use with 20 mA, dc-keyed devices.
Transmission Distance	1500 ft maximum (environment dependent) EIA interface transmission distance is limited only by characteristics of modem and associated communication facility. A 25-ft cable to the modem is supplied. (BC01A-25).

DC02-E Control Instructions

The following instructions are used with the DC02E control:

MTPF Multiple Teleprinter Flag

Octal code: 6113
Event time: 1, 2
Execution time: 4.25 μs
Operation: Transfer status of teleprinter flags to AC $_{0-3}$.
Symbol: Teleprinter flags $\rightarrow AC_{0-3}$.
$0 \rightarrow AC_{4-11}$.

MINT Multiple Interrupt

Octal code: 6115
Event time: 3
Execution time: 4.25 μs
Operation: Interrupt on if AC_{11} is set (interrupt request if any flags).
Symbol: $AC_{11} \rightarrow$ Interrupt Enable

MTON Multiple Station Select

Octal code: 6117
Event time: 1, 2, 3
Execution time: 4.25 μs
Operation: Transfer AC_{0-3} to selection register (select stations when bit is set).
Symbol: $AC_{0-3} \rightarrow SR$

MTKF Multiple Keyboard Flag

Octal code: 6123
Event time: 1, 2
Execution time: 4.25 μs
Operation: Transfer status of keyboard flags to AC_{0-3}.
Symbol: $KF \rightarrow AC_{0-3}$
$0 \rightarrow AC_{4-11}$

MINS   Multiple Interrupt and Skip

Octal code: 6125
Event time: 1, 4
Execution time: 4.25 μs
Operation: Skip if the interrupt request is active (if interrupt is on and any flag is raised).
Symbol: If interrupt request is active, PC + 1 → PC

MTRS   Read Station Select Status

Octal code: 6127
Event time: 1, 2, 3
Execution time: 4.25 μs
Operation: Transfer the status of the selection register to AC_{0-3}.
Symbol: SR → AC_{0-3}
0 → AC_{4-11}

DC02-D Instructions

Decoder instructions for the DC02-D are in two basic groups: receiver and transmitter. The receiver instructions are 6111, 6112, 6114, and 6116; the transmitter instructions are 6121, 6122, 6124, and 6126. The "Station Select" flip-flop is gated with each device select code; therefore, these IOTs are effective only when a particular station is selected.

MKSF   Skip On Keyboard Flag

Octal code: 6111
Event time: 1
Execution time: 4.25 μs
Operation: Skip if the selected keyboard flag is set.
Symbol: If KF = 1, PC + 1 → PC

**MKCC   Clear Keyboard Flags*

Octal code: 6112
Event time: 2
Execution time: 4.25 μs
Operation: Clear the keyboard flag of the selected station; clear AC. If the paper tape reader is on, the tape will advance 1 character and the receiver flag will be set approximately 100 ns after issuing the instruction.
Symbol: 0 → KF, 0 → RF, 0 → AC

MKRS   Read Keyboard Register Static

Octal code: 6114
Event time: 3
Execution time: 4.25 μs
Operation: Transfer the keyboard register contents to AC_{4-11}.
Symbol: AC_{0-11} V KR to AC_{0-11}.

---

* On some systems a logic revision has been added which inhibits tape advance and consequent setting of the receiver flag.

MKRB Load Keyboard Sequence

Octal code: 6116
Event time: 2,3
Execution time: 4.25 μs
Operation: Clear the keyboard and reader flags, clear AC; transfer the keyboard register contents to $AC_{4\text{-}11}$ (MKCC and MKRS combined).
Symbol: $0 \rightarrow KF$
$0 \rightarrow RF$
$0 \rightarrow AC$
$KR \rightarrow AC_{4\text{-}11}$

MTSF Skip on Teleprinter Flag

Octal code: 6121
Event time: 1
Execution time: 4.25 μs
Operation: Skip if the selected teleprinter flag is set.
Symbol: If $TF = 1$, $PC + 1 \rightarrow PC$

MTCF Clear Teleprinter Flag

Octal code: 6122
Event time: 2
Execution time: 4.25 μs
Operation: Clear the teleprinter flag of the selected keyboard station.
Symbol: $0 \rightarrow TF$

MTPC Multiple Load Teleprinter Register and Print

Octal code: 6124
Event time: 3
Execution time: 4.25 μs
Operation: Load $AC_{4\text{-}11}$ into the teleprinted register (begin print/punch).
Symbol: $AC_{4\text{-}11} \rightarrow TR$

MTLS Load Teleprinter Sequence

Octal code: 6126
Event time: 2, 3
Execution time: 4.25 μs
Operation: Clear the teleprinter flag and load $AC_{4\text{-}11}$ into the teleprinter register (MTCF and MTPC combined).
Symbol: $0 \rightarrow TF$
$AC_{4\text{-}11} \rightarrow TR$

Multiple Teletype Control, Type DC02-F

This control is similar to the DC02-E in operation and design. It consists of a double row of logic (10-1/2" x 19" nominal rack space) complete with power supplies. This unit will handle up to eight stations. The DC02-F is designed so that four controls may be connected together in such a way that 32 stations total may be operated. A jumper module is used to enable selected signals when the system is expanded.

As in the four-station control DC02-E, the high order bits of the AC are used to select a station. $AC_{0\text{-}7}$ are used to select the stations within one control and $AC_{8\text{-}11}$ are used to select any one of up to four DC02-F controls. Up to three separate clocks may be specified. In addition, two frequency divide registers are available with outputs at divide-by-8 and divide-by-128. The resultant timing pulses may be connected in any combination with jumper modules to provide timing for the transmitter and receiver modules of the stations.

The specifications for this control are the same as those of the DC02-E. The program instructions are likewise identical except for the following.

MTPF   Multiple Teleprinter Flag

Octal code:	6113
Event time:	1, 2
Execution time:	4.25 μs
Operation:	Transfer status of teleprinter flags AC_{0-7}. AC_{8-11} selects one of four controls.
Symbol:	Teleprinter flags → AC_{0-7}
	0 → AC_{8-11}

MTKF   Multiple Keyboard Flag

Octal code:	6123
Event time:	1, 2
Execution time:	4.25 μs
Operation:	Transfer status of keyboard flags to AC_{0-7}
Symbol:	KF → AC_{0-7}
	0 → AC_{8-11}

MTON   Multiple Teleprinter

Octal code:	6117
Event Time:	1, 2, 3
Execution time:	4.25μs
Operation:	Transfer AC_{0-7} to selection register select stations when bits are set. AC_{8-11} selects one of four controls.
Symbol:	AC_{0-7} → SR
	0 → AC_{8-11}

MINS   Multiple Interrupt and SKP

Octal code:	6125
Execution time:	4.25 μs
Operation:	Skip if the interrupt request is active (i.e., if interrupt is on and any flag is raised) AC_{8-11} select one of four controls.
Symbol:	0 → AC_{8-11}

MTRS   Multiple Teleprinter Read Status

Octal code:	6127
Event time:	1, 2, 3
Execution time:	4.25 μs
Operation:	Transfer the status of the selection register to AC_{0-7}. AC_{8-11} selects one of four controls. Transfer status of interrupt enable flip-flop to AC_{11}. If AC_{11} = 1, the interrupt enable is set.
Symbol:	SR → AC_{0-7}
	Interrupt Enable → AC_{11}
	0 → AC_{8-10}

6.2 REAL TIME CLOCKS

6.2.1 Real Time Interface, Type KW12-A

The KW12-A (see Figures 6-5 and 6-6) is a prewired PDP-12 option with an input control panel, which mounts behind the vertical door on the left front of the PDP-12. The Real Time Interface can be used to synchronize the central processor to external events, count external events, measure intervals of time between events, or provide program interrupts at programmable intervals from 2.5 μs to over 40 seconds. Some of the above mentioned operations can be done simultaneously.

(yeah, but not over 41)

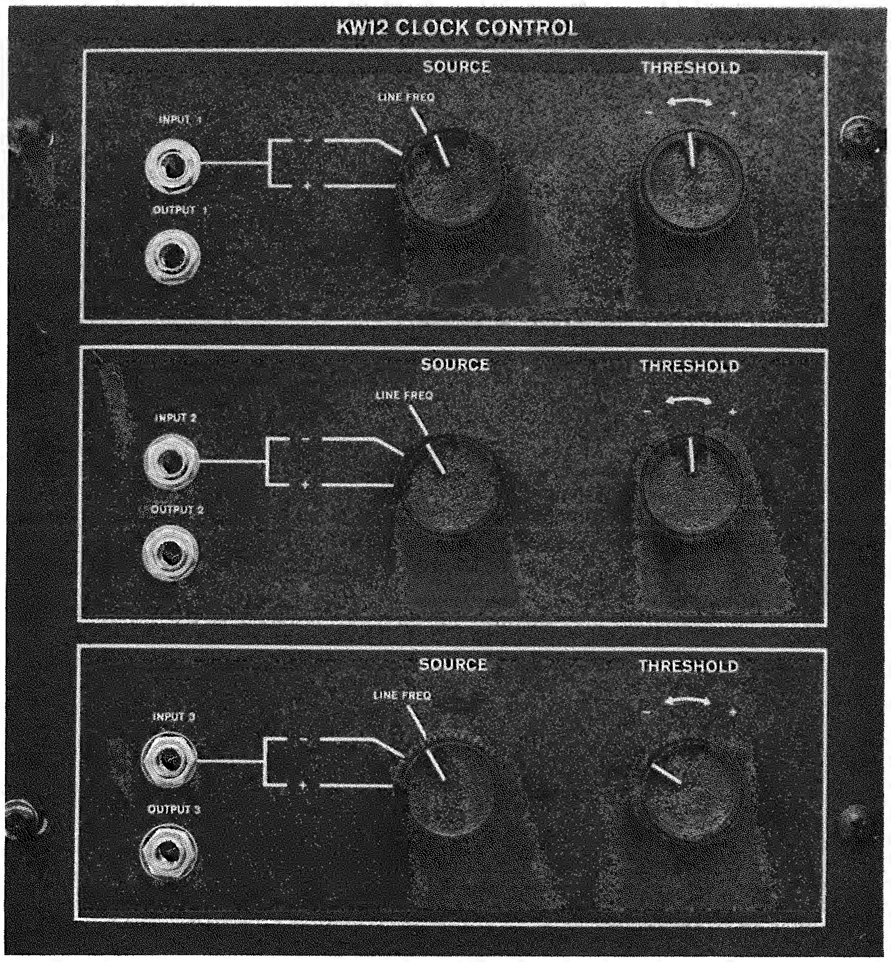


Figure 6-5. KW12-A Real Time Interface

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	1 kHz
100 kHz	100 Hz
10 kHz	

Input channel 1 may be used to enable an external source to drive the counter. This external source may be switch-selected to be either the power line waveform or an actual signal from a laboratory instrument. The programmable selection of the rate is accomplished with the three rate-bits of the clock control register.

Input Synchronizers

Three input channels are used to convert external events into a synchronized 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 Schmitt trigger and pulse generator converts the preselected voltage threshold crossing by an external signal into a single event (pulse). This Schmitt trigger has level and slope selection controls available on the front panel. They provide selection of any threshold between ±6V and either positive or negative-going slope. The Schmitt trigger has a hysteresis of 0.3V.

cf ¶ 6.2.1.2 pg 6-22

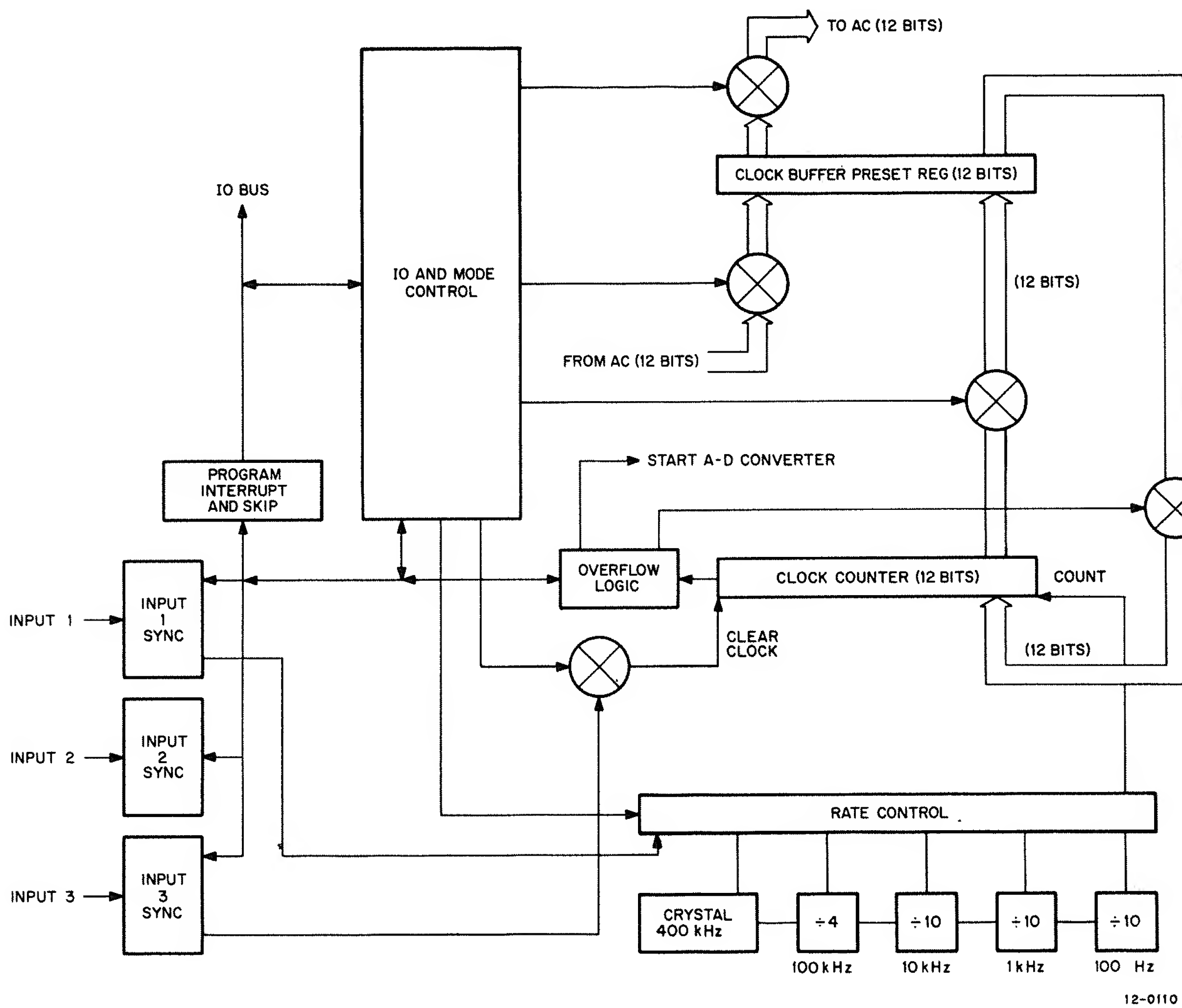


Figure 6-6. KW12-A Organization

The five Input Circuits flip-flops are: INPUT ENABLE, INPUT, PRE-EVENT, EVENT, and ENABLE EVENT INTERRUPT (see Figure 6-7). Figure 6-8 shows a basic timing of the Input Synchronizer.

Not shown in Figure 6-7 is the logic which clears the EVENT and PRE-EVENT flip-flops when the CLSA instruction is given. This logic ensures that events occurring during the execution of the present CLSA instruction are indicated when the next CLSA instruction is given.

Input Enable Flip-Flop – Gates on and off input signals to the clock. It is set and cleared under program control.

Input Flip-Flop – Set by an external signal from the Schmitt 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 – Strobed into the EVENT flip-flop by the strobe 1 signal. This clears the INPUT flip-flop if EVENT is clear.

Event Flip-Flop – Loaded with the PRE-EVENT flip-flop on the next strobe 1 and set to a 1 by the second strobe 1 following the setting of the INPUT flip-flop. Subsequent to EVENT being loaded, PRE-EVENT is cleared.

Event Enable Interrupt Flip-Flop – Permits external events to cause program interrupts. It is set and cleared by the CLEN instruction.

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.

The status of the EVENT and PRE-EVENT flip-flops is 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 the EVENT flip-flop is cleared, then both the PRE-EVENT and EVENT flip-flops will remain set, indicating an error.

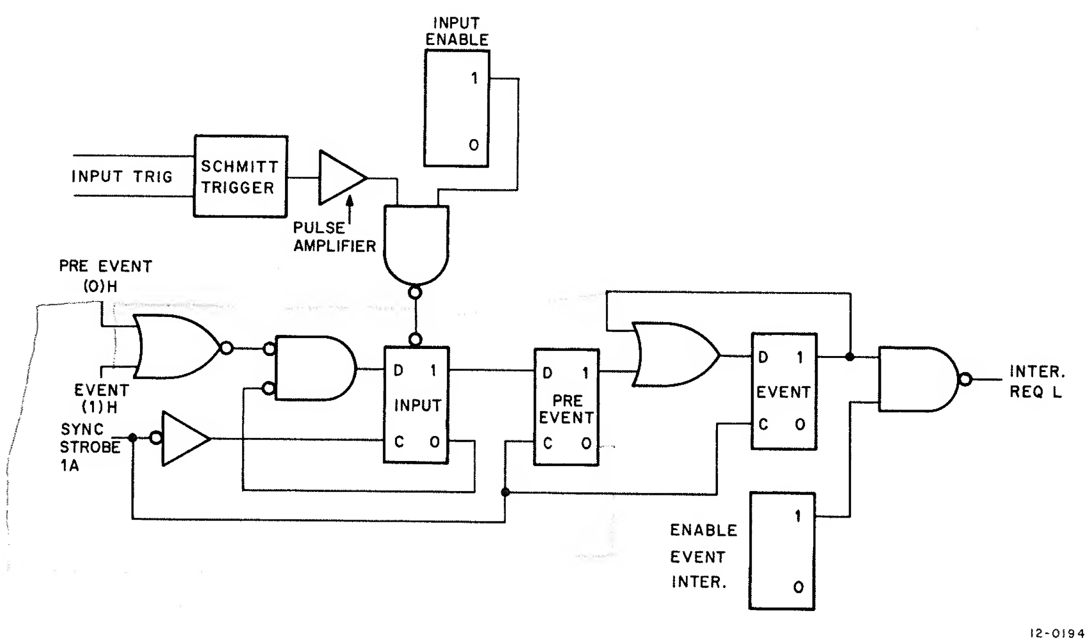
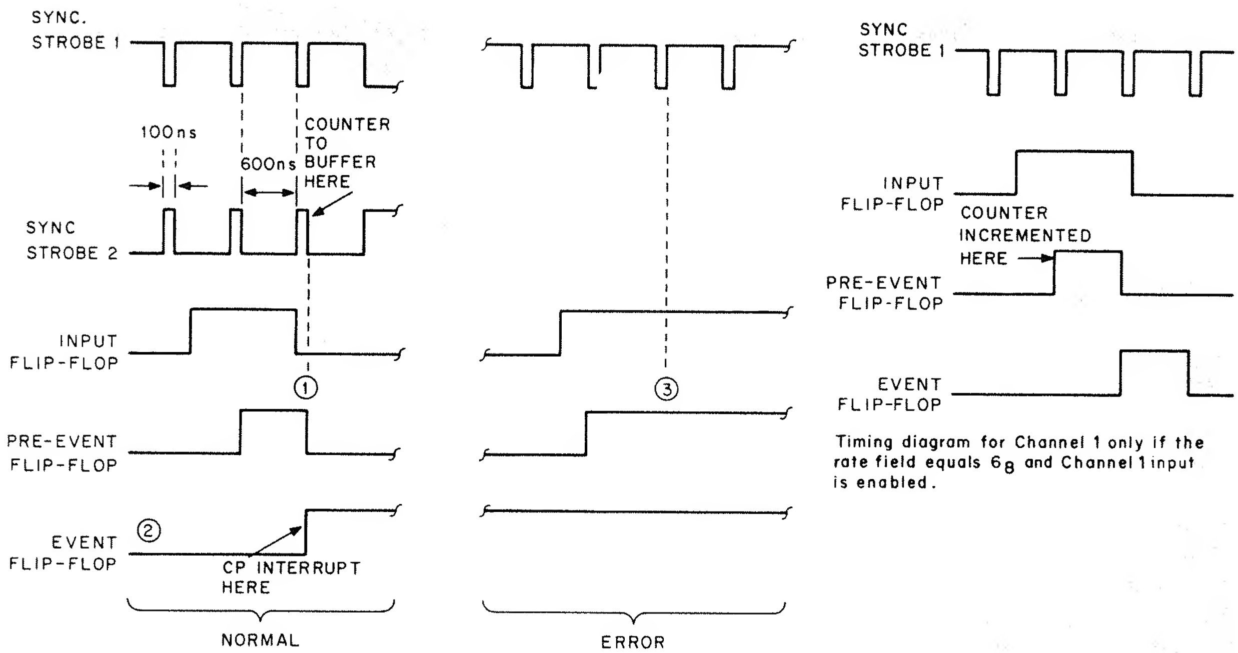


Figure 6-7. Simplified Input Synchronizer Logic Diagram



NOTES:

1. Normally PRE-EVENT is reset. If however, EVENT was already set, PRE-EVENT remains set as an error flag; indicating that more than one input occured between clearing of the EVENT flip-flop.
2. EVENT is cleared only by the CPU using the 6135 (CLSA) instruction with the following exception: Inputs to Channel 1 may be used to increment the counter. For this case the timing diagram is shown below.
3. Error indication that 6135 (CLSA) instruction was not issued soon enough following an EVENT.

12-0205

Figure 6-8. KW12-A Timing Diagram

Counter Register

The counter register is a 12-bit counter that is loaded from the buffer-preset register or can be transferred into the buffer-preset register.

The counter may be used to count events, measure intervals of time between events, or provide processor interrupts at program selected intervals from 2.5 μs to over 40 seconds.

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

Buffer-Preset Register – 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.

Refers to Mode Control Register Bit ØØ of which is bit Ø3 of the Clock Control Register.

6.2.1.1 **Use of Interface 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 triggered only by the clock. When a SAM instruction is given, the result of the last conversion is transferred to the AC, and a new analog channel is selected for the next conversion to be performed when the clock overflows. If the SAM instruction is given while a conversion triggered by the clock is in progress, the processor waits in timestate TS5 until the conversion is complete.

6.2.1.2 **KW12-A Input Panel** – The input panel for the clock is located behind the door on the left side of the front of the PDP-12. External signals (from up to three instrument/sources) are connected to the Input jacks on the front panel (see Figure 6-5) via standard 3-conductor telephone plugs. Each Input and Output jack is wired in parallel to permit convenient connection of the external input signals to the KW12-A and, if desired, the Analog-to-Digital Converter. The input is differential, ±5V range, input resistance greater than 10,000 ohms, and protected against inputs up to ±50V. A level control and a slope control are associated with each input. The level control selects the threshold voltage at which the trigger pulse is generated. The slope determines the polarity of the input signal causing a trigger pulse. The trigger pulse sets the associated input flip-flop of the clock if that input channel is enabled.

6.2.1.3 **KW12-A Real Time Interface Instructions** – The KW12-A is controlled by PDP-12 IOT instructions. These instructions can be used from either 8 or LINC mode. Execution time for the IOTs is 4.25 μs when in 8 mode and 5.9 μs when in LINC mode (using IOB).

CLSK Skip On Clock Interrupt

Octal code: 6131
Event time: 1
Execution time: 4.25 μs
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 Event 3 (1).
d. Enable Overflow Interrupt (1) and Overflow (1).

CLLR Load Clock Control Register

Octal code: 6132
Event time: 2
Execution time: 4.25 μs
Operation: The contents of the AC are 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	Function
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
07	Simulate Input to Channel 1
08	Not used
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	Period of Count Pulses	Clock Control Register
000	Stop	∞	ØXXX
001	400 kHz	2.5 μsec	1XXX
010	100 kHz	10 μsec	2XXX
011	10 kHz	100 μsec	3XXX
100	1 kHz	1 msec	4XXX
101	100 Hz	10 msec	5XXX
110	Rate of Input Channel 1		6XXX
111	Stop	∞	7XXX

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. Also the clock counter is advanced one count each time an IO PRESET is performed either manually or under program control.

The contents of the mode control register determine the method by which the clock system operates.

Program Example

The following program rings the teleprinter bell once for every ~~10~~ 60_{10} inputs (60 Hz) on external channel ~~three~~ one. If the source knob is turned to line frequency the Teletype bell will ring once/per second on computers connected to 60 Hz power lines.

```
            PMODE           /PSEUDO TO LAP6 DIAL ASSEMBLER
BEGIN,      CLA             /CLEAR AC
            CLLR            /CLEAR ALL MODES
            CLEN            /CLEAR ENABLES
            TAD K74         /NO. OF COUNTS
            CIA             /FORM 25 COMP   FORM 2'S COMP
            CLAB            /LOAD PRESET REG
            CLA             /CLEAR AC
            TAD K0100       /SET AC05=1
            CLLR            /GENERATE CLEAR COUNTER
            CLSA            /CLEAR STATUS AND POSSIBLE OVERFLOW
            CLA             /CLEAR AC
            TAD K0320       /LOAD AC
            CLEN            /LOAD BUFFER INTO COUNTER, ENABLE INTERRUPT ON OVERFLOW,
                             AND ENABLE INPUT CHAN 1
PUT CHAN 1
            CLA             /CLEAR AC
            TAD K6100       /LOAD AC
            CLLR            /START CLOCK BY ENABLING RATE
LOOP,       CLSK            /SKIP ON CLOCK FLAG
            JMP LOOP        /WAIT
            CLSA            /CLEAR STATUS
            CLA             /CLEAR AC
            TAD K207        /LOAD AC
            TLS             /RING TELEPRINTER BELL
            JMP LOOP        /RETURN TO WAIT LOOP
K74,        74              /CONSTANTS
K0100,      100
K0320,      320
K6100,      6100
K207,       207
```

Mode Control Register

000 — Free-run
Counter runs selected rate. Overflow occurs every 4096 counts. The overflow flag remains set until cleared with CLSA instruction.

001 — Preset Time
Counter runs at selected rate. When overflow occurs, the contents of the Clock Buffer-Preset register are transferred automatically to the Counter which continues. The Overflow flag remains set until cleared with a CLSA instruction. When Mode is changed from X00 to X01, the clock counter is zeroed.

010 — Time Base (measures intervals between events)
Counter runs at selected rate. On the occurrence of an Input Event, the contents of the counter are transferred automatically to the Buffer Preset register, and the counter continues to count.

011 — Time Base (measures intervals between events)
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. Events 1 and 2 remain only to cause transfer from the clock counter to the Buffer Preset register.

100
101
110
111 — When Mode bit 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 conversions to take place under the automatic timing control of the clock. In this mode, A-D conversions are triggered only by the clock counter overflow. The SAM instruction reads the result of the previous conversion and sets the channel number for the next conversion. For details of the Analog-to-Digital converter, see Paragraph 6.11. The remaining two Mode Control bits are decoded exactly as above.

wrong A-D hardware

CLAB AC to Buffer Preset Register

Octal code: 6133
Event time: 2 1.2
Execution time: 4.25 μs
Operation: Transfer AC to Buffer Preset register. The previous contents of the Buffer Preset registers are lost and the AC is unchanged.

CLEN Load Clock Enable Register

Octal code: 6134
Event time: 3
Execution time: 4.25 μs The AC is unchanged.
Operation: The contents of the AC are transferred to the Clock Enable register. The function of each bit is given below:

Bit	Function
00	Not used
01	Not used
02	Not used
03	Not used
04	Inclusive OR Clock Buffer into counter if mode = X01 and overflow = 0
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 on Event 3
11	Enable Input Channel 3

Buffer Preset Register?

Interrupt on line frequency occurs @ ca. 135° (ie. ca. 2 msec before down-going zero-crossing)

CLSA   Clock Status to AC

Octal code: 6135
Event time: 1,3
Execution time: 4.25 μs
Operation: This instruction interrogates the CLOCK INPUT and OVERFLOW STATUS flip-flops. The clock status information is inclusive ORed into the AC. Then the AC status bits which are set are cleared. from the CLOCK INPUT and OVERFLOW STATUS flip-flops. This ensures that only one occurrence of an EVENT will be transferred to the program.

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

If any pair is set, >1 input at that channel has occurred.

CLBA   Buffer Preset Register to AC

Octal code: 6136
Event time: 2,3
Execution time: 4.25 μs
Operation: The AC 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 μs
Operation: The AC is cleared and the contents of the Clock Counter are transferred to the BUFFER-PRESET register. Then the contents are transferred into the AC.

6.2.2 KW12-B and KW12-C Fixed-Interval Clocks

The KW12-B provides a means of interrupting the CP at intervals determined by a variable RC Oscillator. The KW12-C is similar to the KW12-B except that the RC oscillator is replaced by a crystal oscillator with a single fixed frequency. The KW12-B or KW12-C may be turned on or off under program control. However, variations in frequency require physically altering or changing the oscillator.

Instruction Set

The KW12-B and KW12-C are controlled by IOT instructions. These instructions can be used from either mode. Execution time for the Simple Clock IOTs is 4.25 μs when in 8 mode and 5.9 μs when in LINC mode (using IOB).

CSOF Skip on Clock Flag

Octal code: 6131
Execution time: 4.25 μs
Operation: Skip if clock flag is set.

CTOC Turn Off Clock

Octal code: 6132
Execution time: 4.25 μs
Operation: Turn off the clock, clear the clock flag, and disable the clock interrupt.

CTON Turn On Clock
Octal code: 6134
Execution time: 4.25 μs
Operation: Turn the clock on and clear the flag.

CRUN Clock Running

Octal code: 6135
Execution time: 4.25 μs
Operation: Turn on the clock, enable the clock interrupt, clear the clock flag, skip if the clock flag was set when the instruction was issued.

Frequency Range

KW12-B The KW12-B uses the M401 variable clock in slot F18 for a time base. The frequency may be varied with a range by adjusting a potentiometer on the module. Five frequency ranges are available. The KW12-B is nominally set to the 1.75 kHz-to-17.5 kHz frequency range by jumpering F18N2 to F18T2. Other frequency ranges may be achieved by removing the jumper from F18T2 and attaching as shown below. The exact frequency may be checked by observing the signal on F18D2 with an oscilloscope.

? Looks like three

Frequency Range	Interconnections Required
17.5 kHz to 50 kHz	F18N2 – F18S2
1.75 kHz to 17.5 kHz	F18N2 – F18T2 (nominal setting)
175 Hz to 1.75 kHz	F18N2 – F18P2

KW12-C The KW12-C uses the M405 crystal clock in slot F18 for a time base. The frequencies are fixed by a series resonant crystal oscillator to obtain a frequency stability of .01 percent of the specified value between 0°C and +55°C. The frequencies available are in the range of 5 kHz to 50 kHz and must be specified in advance by the customer.

6.3 DISK STORAGE

6.3.1 Random Access Disk File, Types DF/DS32 and DF/DS32-D

The DF32 and DF32-D operate through the three-cycle data break facility to provide 32,768 13-bit words (12 bits plus 1 parity bit) of storage, and are economically expandable to 131,072 13-bit words using the respective expander disk, DS32 or DS32-D.

The DF32 and DF32-D comprise two basic assemblies: the storage unit containing the read/write circuits and the computer interface logic. The storage unit contains a nickel cobalt plated disk driven by a hysteresis synchronous motor. Data is recorded on a single disk surface by 16 read/write heads in fixed positions. The disk motor, read/write data heads, and photocell assembly (on DF32) are all mounted on a rack assembly that permits sliding the unit in and out of a DEC 19-inch Standard Cabinet.

The following table summarizes the DF32 and DF32-D:

	DF32		DF32-D	
	60Hz	**50Hz**	**60Hz**	**50Hz**
Data Transfer rate (microseconds per word)	66	80	32	39
Average Access Time (milliseconds)	16.67	20	Same	Same
Timing and Address Track "Start & Finish" Sensing	Photo-electric reflective marker on disk outer perimeter		Special Start and Finish coding and decoding on timing track	
Type of Logic	Negative		DF32DP – Positive DF32DN – Negative	

Instructions

The DF32 and DF32-D disk systems share the following instruction set:

DCMA Clear Disk Memory Address Register

Octal code:	6601
Event time:	1
Execution time:	4.25 μs
Operation:	Clears Memory Address Register, parity error, and completion flags. This instruction clears the disk memory request flag and interrupt flags.
Symbol:	$0 \rightarrow$ completion flag $0 \rightarrow$ error flag

DMAR Load Disk Memory Address Register and Read

Octal code:	6603
Event time:	1,2
Execution time:	4.25 μs
Operation:	The contents of the AC are loaded into the disk memory address register and the AC is cleared. Begins to read information from the disk into the specified core location. Clears parity error and completion flags. Clears interrupt flags.
Symbol:	$AC_{0-11} \rightarrow DMA_{0-11}$ $0 \rightarrow$ completion flag $0 \rightarrow$ error flag

DMAW Load Disk Memory Address Register and Write

Octal code:	6605
Event time:	1,3
Execution time:	4.25 μs
Operation:	The contents of the AC are loaded into the disk memory address register and the AC is cleared. Begins to write information onto the disk from the specified core location. Clears parity error and completion flags. Clears interrupt flags. Data break must be allowed to occur within 66 microseconds after issuing this instruction.
Symbol:	$AC_{0-11} \rightarrow DMA_{0-11}$ $0 \rightarrow$ completion flag $0 \rightarrow$ error flag

DCEA Clear Disk Extended Address Register

Octal code:	6611
Event time:	1
Execution time:	4.25 μs
Operation:	Clears the Disk Extended Address and Memory Address Extension register.
Symbol:	$0 \rightarrow$ Disk Extended Address Register $0 \rightarrow$ Memory Address Extension Register

DSAC Skip on Address Confirmed Flag

Octal code:	6612
Event time:	2
Execution time:	4.25 μs
Operation:	Skips next instruction if address confirmed Flag is set. Flag is set for 16 μs (AC is cleared).
Symbol:	If address confirmed flag = 1, then $PC + 1 \rightarrow PC$

DEAL Load Disk Extended Address

Octal code:	6615
Event time:	1,3
Execution time:	4.25 μs
Operation:	The disk Extended Address and Memory Address Extension registers are cleared and loaded with the track address data in the AC.
Symbol:	$AC_{6-8} \rightarrow$ Core Memory Extension $AC_{1-5} \rightarrow$ Disk Address Extension 32K, 64K, 96K, 128K $AC_{0\ 9-11}$ used in DEAC instruction (See NOTE on opposite page)

DEAC   Read Disk Extended Address Register

Octal code: 6616
Event time: 2,3
Execution time: 4.25 μs
Operation: Clear the AC then load the contents of the disk Extended Address register into the AC allowing program evaluation. Skip next instruction if address confirmed flag is set.
Symbol: Disk Address Extension 32K, 64K, 96K, 128K $\rightarrow AC_{1-5}$
Core Memory Extension $\rightarrow AC_{6-8}$
Photo-cell sync mark $\rightarrow AC_0$ (Available 200 μs)
Data Request Late flag $\rightarrow AC_9$
Non-Existent or Write Lock switch on $\rightarrow AC_{10}$
Parity Errors $\rightarrow AC_{11}$

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 μs
Operation: Skips next instruction if parity error, data request late, and write lock switch flag are clear. Indicates no errors.
Symbol: If Parity Error flag = 1
and Data Request Late flag = 1,
and Write Lock Switch flag = 1, then PC + 1 $\rightarrow$ PC

DFSC   Skip on Data Completion Flag

Octal code: 6622
Event time: 2
Execution time: 4.25 μs
Operation: Skips next instruction if the completion flag is set. Indicates data transfer is complete.
Symbol: If Completion flag = 1, PC + 1 $\rightarrow$ PC

DMAC Read Disk Memory Address Register

Octal code:	6626
Event time:	2,3
Execution time:	4.25 μs
Operation:	Clears the AC then loads contents of the Disk Memory Address register into the AC allowing program evaluation. During read, the final address is the last one transferred.
Symbol:	$DMA_{0-11} \rightarrow AC_{0-11}$

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 two'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 the disk module contains write instructions. The nonexistent disk condition will appear following the completion of a data transfer during read, where the address acknowledged was the last address of a disk and the next word to be addressed falls within a nonexistent disk. The completion flag for the data transfer is set by the nonexistent disk condition 16 microseconds following the data transfer.

6.3.2 Disk Memory System Type RF/RS08

The RF08 control and RS08 disk combine as a fast, low-cost, random access bulk storage package for the PDP-12A. 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 data break system of the PDP-12A.

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 mounting in a standard 19-inch DEC cabinet.

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 (Table 6-4).

Instructions

The RF/RS08 has the following DF32 Programming Compatibility:

DCMA Clear Disk Memory Address Register

Octal code:	6601
Event time:	1
Execution time:	4.25 μs
Operation:	Clear the Disk Memory Address register, and all other disk and maintenance flags except interrupt enable.

DMAR   Load Disk Memory Address Register and Read

Octal code:	6603
Event time:	1,2
Execution time	4.25 μs
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.

During Read, the final address status is the last address transferred + 1.

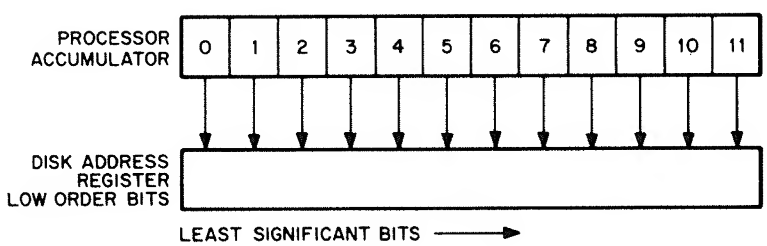
When reading the last address of the last available disk the nonexistent disk flag is raised in coincidence with the completion flag.

DMAW   Disk Memory Address Register and Write

Octal code:	6605
Event time:	1,3
Execution time:	4.25 μs
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.



The DF32 maintenance instruction IOT 663X is not assigned to the RF08 system.

DCIM   Clear Interrupt Enable and Core MA Extension Registers

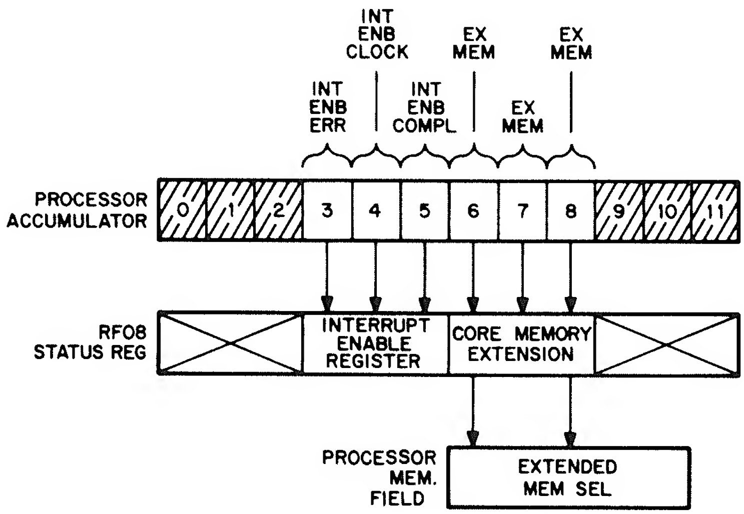
Octal code:	6611
Event time:	1
Execution time:	4.25 μs
Operation:	Clears the disk interrupt enable and core memory address extension registers.

DSAC   Skip on Confirmed Flag

Octal code:	6612
Event time:	2
Execution time:	4.25 μs
Operation:	Maintenance Instruction Skip next instruction if the Address Confirmed flag is a 1. (AC is cleared.)

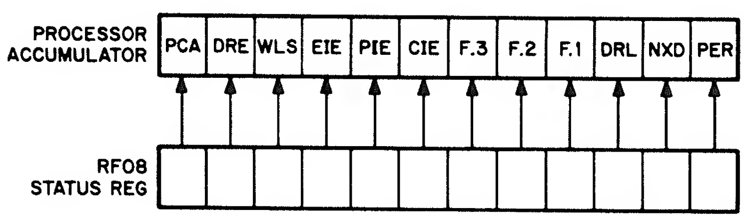
DIML Clear Interrupt Enable and MA Extension Register and Load

Octal code: 6615
Event time: 1,3
Execution time: 4.25 μs
Operation: Clears the interrupt enable, and memory address extension register. Then loads the interrupt enable and memory address extension registers with data held in the accumulator. Then clears AC.



DIMA Clear AC and Load from Status

Octal code: 6616
Event time: 2,3
Execution time: 4.25 μs
Operation: Clears the accumulator. Then loads the contents of the status into the accumulator to allow program evaluation



DFSE Skip on Zero Error Flag

Octal code: 6621
Event time: 1
Execution time: 4.25 μs
Operation: Skips the next instruction if there is a Parity Error, Data Request Late, Write Lock Status, or Nonexistent Disk flag set.

DFSC Skip on Data Completion Flag

Octal code: 6622
Event time: 2
Execution time: 4.25 μs
Operation: Skips next instruction if the Completion flag is a 1 (data transfer is complete).

DISK Skip on Error or Completion Flag

Octal code: 6623
Event time: 1,2
Execution time: 4.25 μs
Operation: Skips next instruction if either the Error or Completion flags or both are set.

DMAC Read Disk Memory Address Register

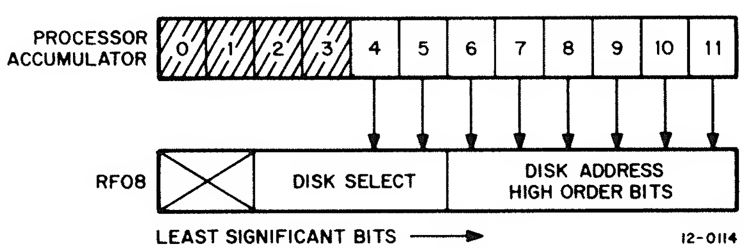
Octal code: 6626
Event time: 2,3
Execution time: 4.25 μs
Operation: Clears the accumulator. Then loads the contents of the Disk Memory address register into the accumulator to allow program evaluation. This instruction must be issued when the completion flag is set.

DCXA Clear Address Register

Octal code: 6641
Execution time: 4.25 μs
Operation: Clears the high order 8-bit disk address register.

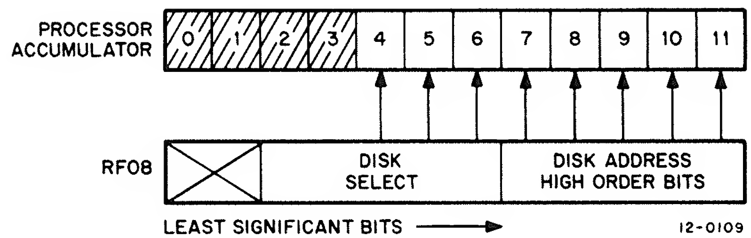
DXAL Clear Address Register and Load from AC

Octal code: 6643
Execution time: 4.25 μs
Operation: Clears the High Order 8 bits of the disk address register. Then loads the disk address register from the data held in the accumulator. Then clears the AC.



DXAC Clear AC and Load

Octal code: 6645
Execution time: 4.25 μs
Operation: Clears the accumulator, then loads the contents of the High Order 8-bit disk address register into the accumulator.



DMMT (Maintenance Instruction)

Octal code: 6646

Execution time: 4.25 μs

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's 646 and associated AC bits. AC is cleared. The maintenance register is initiated by issuing an IOT 601 command.

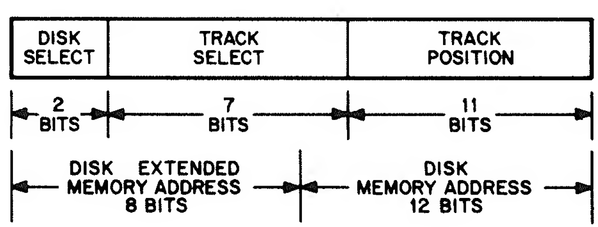
$AC_{11}(1)$	Track *A* Pulse
$AC_{10}(1)$	Track *B* Pulse
$AC_9(1)$	Track *C* Pulse
$AC_7(1)$	DATA PULSE (DATA HEAD #0)
$AC_6(1)$	+1 Photocell
$AC_0(1)$	+1 DBR

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.

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.

Table 6-4. DF32 Instructions Compared with RF/RS08 Instructions

DF32 Mnemonic	Octal Code	RF08 Mnemonic	RF08 to DF32 Comparison
DCMA	6601	same	Identical functions.
DMAR	6603	same	Identical functions.
DMAW	6605	same	Identical functions.
DCEA	6611	DCIM	Clears interrupt enable, does not clear EMA. On both units, clears memory address extension.
DSAC	6612	same	Identical functions.
DEAL	6615	DIML	Similar, except functions transmitted from the AC are different. EMA information not transmitted. See DXAL.
DEAC	6616	DIMA	Similar, except that functions transmitted to the AC are different. See DXAC.
DFSE	6621	same	Instruction is skip on error, rather than skip-no error. NXD added as an error.
DFSC	6622	same	Identical function.
(none)	6623	DISK	New instruction. Skips on error or data completion, or both. (DFSE and DFSC combined.) Skip enabled at IOP 2.
DMAC	6626	same	Identical functions.
(none)	6641	DCXA	Clears EMA.
(none)	6643	DXAL	Clear and load EMA with information in the accumulator.
(none)	6645	DXAC	Clear accumulator and load address in EMA into the accumulator.
(none)	6646	DMMT	Maintenance instruction. See description.

```
                /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 WRITE 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
                DIML
                ISZ WRT
                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 TO DMA IN DISK;
                                    /START WRITE OPERATION
                                    /WRITE OPERATION
                DISK                /FOR ERROR OR COMPLETE
                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
```

NOTE

This coding assumes no live interrupts.

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 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 I WRT) loads the AC with the contents of SUB+1 or 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 operates under 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 are detected, control jumps to an error or diagnostic (not shown) routine. If no errors are found, 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 parity bit).	
Disks	Four RS08s may be controlled by one RF08 for 1,048,576 words.	
Data Transfer Path	Three-Cycle Break	Address Locations 7750 Word Count 7751 Memory Address
Data Transfer Rate	60 Hz Power 16.2 μs per word	50 Hz Power 20 μs per word
Minimum Access Time	258 μs	320 μs
Average Access Time	16.9 ms	20.3 ms
Maximum Access Time	33.6 ms	40.3 ms
Program Interrupt	33 ms 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 address 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 during 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 or shipboard mounting.	
Heat Dissipation	RF08: 150 watts RS08: 500 watts	
AC Power Requirements	115 ± 10 Vac, single phase, 50 ± 2 or 60 ± 2 Hz	
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 nonrecoverable error in 2 x 10^9 bits transferred. A recoverable error is defined as an error that occurs only once in four successive reads. All other errors are nonrecoverable. 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 H950 cabinet is designed to accommodate one RF08, up to two RS08s, and power supplies. Two additional RS08s can be mounted in a second H950. Other equipment should not be mounted in disk cabinets.

6.3.3 RK8 Disk Cartridge Memory System

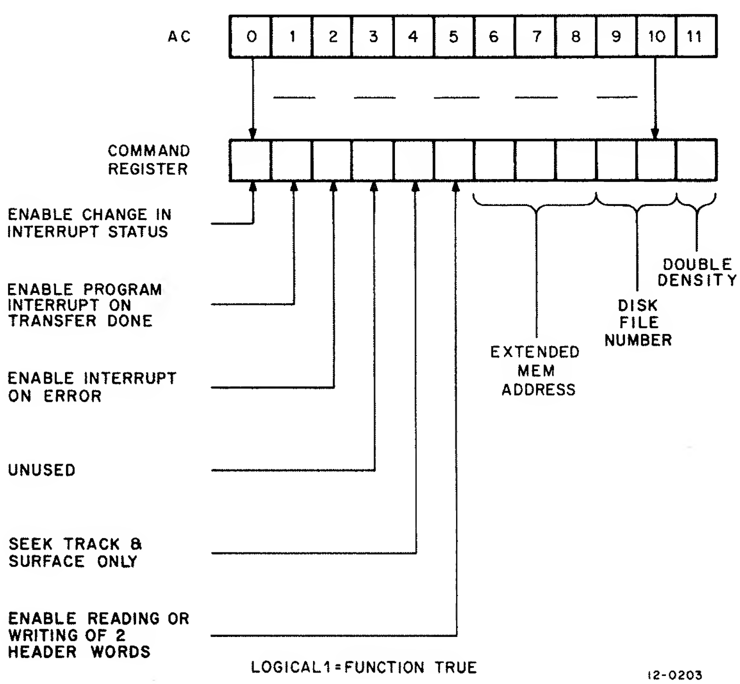
The RK08 Control and RK01 Disk Drive (RK8) are low cost, random access, removable disk storage devices. One RK08/RK01 provides 831,488 13-bit (12 bits plus parity) words of storage. Up to four RK01 disks can be operated by the RK08 Control for a total of 3,325,952 words of storage.

The removable cartridge utilized is an IBM 2315 disk pack or equivalent with a single aluminimum disk platter coated on both sides with magnetic oxide. Average access time is 154 milliseconds with the read/write head at random positions. Data transfer rate is 16.7 microseconds via the PDP-12 single-cycle data break facility. The RK8-P will operate off the positive I/O bus. An RK8-N is also available for connection to a negative I/O bus.

Instructions

DLDC Load Command Register

Octal code: 6732
Operation: Loads the Command Register from the AC, clears the AC.



**DLDR   Load Disk Address and Read*

Octal code: 6733

Operation: Loads track, surface, and sector address from AC, then clears AC and starts to read data from disk if Command Register bit 4 = 0.

**DLDW   Load Disk Address and Write*

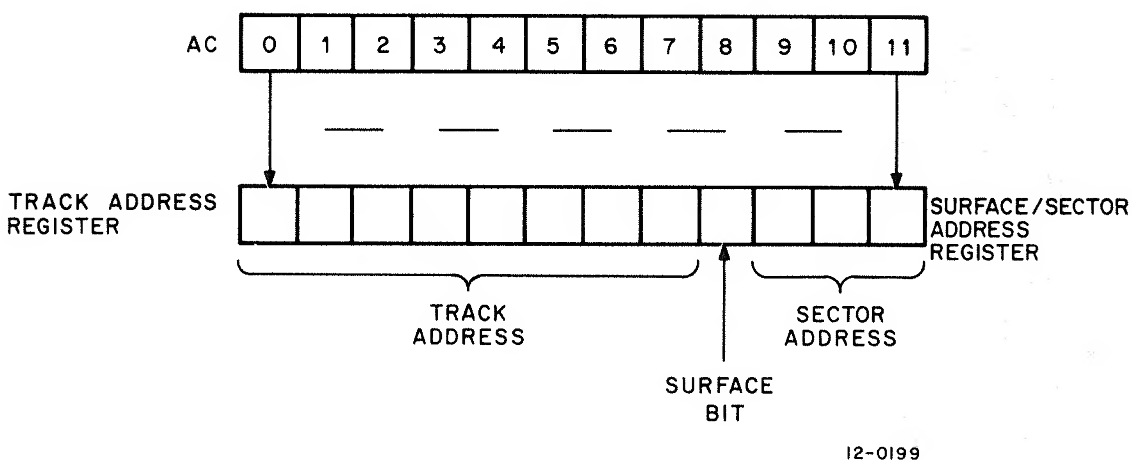
Octal code: 6735

Operation: Loads track, surface, and sector address from AC, then clears AC. Starts to write on disk if Command Register bit 4 = 0.

**DCHP   Load Disk Address and Check Parity*

Octal code: 6737

Operation: Loads track, surface, and sector address from AC, then clears AC. Reads data and checks parity if Command Register bit 4 = 0.



*If command register bit 4 = 1, instruction will be executed only to seek track and surface. These three instructions start all disk operations.

DRDA Read Disk Address

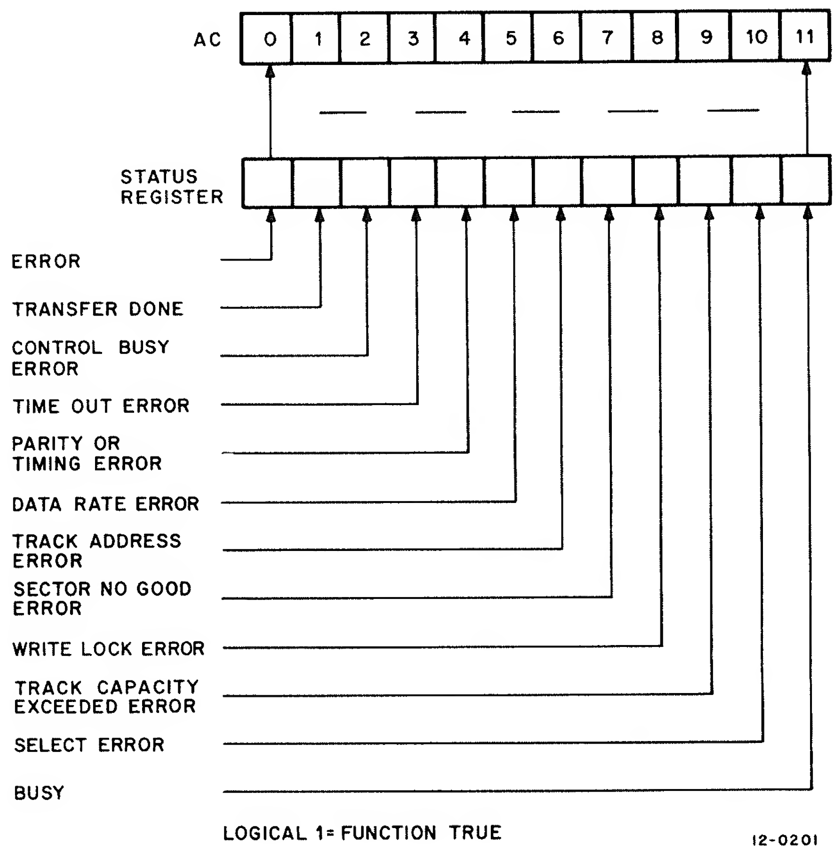
Octal code: 6734
Operation: Clears AC and then reads Track Address Counter and surface/sector counter into AC.

DRDC Read Disk Command Register

Octal code: 6736
Operation: Clears AC then reads Command Register into the AC.

DRDS Read Disk Status Register

Octal code: 6741
Operation: Clears the AC and then reads the Status Register into the AC.

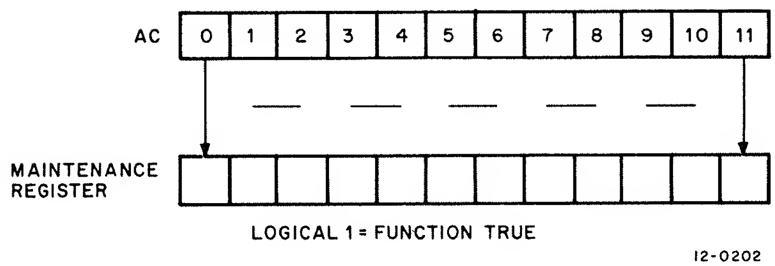


DCLS Clear Status Register

Octal code: 6742
Operation: Clears the Status Register.

DMNT   Load Maintenance Register

Octal Code:	6743
Operation:	Loads Maintenance register from AC and carries out the operation specified. Bits will remain set until DMNT is issued with AC bits = 0.



AC Bit	
0	Transfer contents of Track address Register to Track counter Register.
1	Transfer Data Register to Serial Register.
2	Transfer Serial Register to Data Register.
3	Clear AC and Read Data Register into AC.
4	Shift a "1" into the Serial Register.
5	Shift a "0" into the Serial Register.
6	Unformatted Disk
7	Sector Pulse.
8	Index Pulse.
9-11	Not used.

DLDA   Load Disk Address (Maintenance Only)

Octal code:	6731
Operation:	Loads the disk address register with the content of AC.

DSKD   Skip on Transfer Done Flag = 1

Octal code:	6745
Operation:	Skips next instruction if the Transfer Done Flag is a 1.

DSKE   Skip on Error Flag = 1

Octal code:	6747
Operation:	Skip when Error flag is equal to 1.

DCLA   Clear All

Octal code:	6751
Operation:	Clears selected Disk to Track 000. Then clears all control registers and flags except disk selection. Transfer done set when disk positioned on Track 000.

DRWC Read Word Count Register

Octal Code: 6752
Operation: Clears AC, then reads the contents of the Word Count register into the AC.

DLWC Load Word Count Register

Octal code: 6753
Operation: Loads Word Count register from AC, then clears the AC.

DLCA Load Current Address Register

Octal code: 6755
Operation: Load Current Address register from AC; then clears the AC.

DRCA Read Current Address Register

Octal code: 6757
Operation: Clears the AC then reads the contents of the Current Address register into the AC.

Example of I/O Subroutine

```
DISKIO,     0               /SUBROUTINE ENTRY POINT
            TAD CNT         /FETCH # OF WORDS TO BE READ
            DLWC            /LOAD WORD COUNT REGISTER
            TAD CUR         /FETCH ADD OF MEMORY BUFFER
            DLCA            /LOAD CUR ADDR. REGISTER
            TAD 0000        /PUT COMMAND IN ACCUMULATOR
            DLDC            /LOAD COMMAND REGISTER
            TAD ADR         /FETCH TRACK, SURFACE, & SECTOR
            DLDR            /LOAD DISK ADR AND READ DATA
                            /DLDW WOULD HAVE WRITTEN
            DSKD            /DONE WITH TRANSFER?
            JMP .-1         /NO – GO BACK 1
            DSKE            /YES – IS THERE AN ERROR?
            JMP I DISKIO    /NO ERROR – RETURN
            JMP ERR         /GO TO ERROR SUBROUTINE

CNT         0               /# WORDS TO BE READ
CUR         0               /BEG. ADDR. OF MEMORY BUFFER
ADR         0               /DISK ADDRESS
```

Models *

RK8-P	Positive Bus System; includes RK01 and RK08-P
RK8-N	Negative Bus System; includes RK01 and RK08-N
RK01	Consists of disk drive, interface and removable cartridge (up to 4 per RK8 system)
RK08-P	Positive Bus disk control
RK08-N	Negative Bus disk control

* See RK8 Manual for details.

Specifications

Disks	Four RK01s per RK08 for 3,325,952 words.
Storage Capacity	Each RK01 stores 831,488, 12-bit words.
Data Transfer Rate	16.7 μs per word
Transfer Path	Single-cycle Data Break
Minimum Access Time	2.35 ms
Average Access Time	154 ms
Maximum Access Time	477 ms
Settle Time	37 ms
Program Interrupt	Transfer Done Flag
Write lock	Two words at the beginning of each track contain status information that is automatically checked for write lock of sectors. Also total disk can be write-locked.
Data Tracks	200 plus 3 spare
Words per Track	4096 (2048 on each of two sides)
Sectors	16 per track
Words per Sector	256
Min. Block Size	256 words
Max. Block Size	4096 words
Recording Method	Double Frequency-Time plus Data
Density	1026 bits/inch max.
Speed	1500 rpm

Environmental Requirements

Operating Temp	+65°F to 90°
Operating Humidity	20% to 80% excluding all conditions which would cause condensation.
Heat Dissipation	RK08 – 150 watts RK01 per drive; 700 watts, 1 KW, surge
AC Power Requirements	115 ±10 VAC, 60 ±½ Hertz 230 ±20 VAC, 60 ±½ Hertz

Mechanical Package	One standard DEC cabinet will accommodate the RK08 and one RK01. A second cabinet will house two RK01 units. A third cabinet will house a fourth RK01.

6.3.4 Disk Software

Mass Storage LAP6-DIAL-MS, commonly referred to as DIAL-MS or DISK-DIAL, is a modified version of LAP6-DIAL that may use a disk for the LAP6-DIAL System and/or user programs. The minimum hardware configuration required is 8K of core storage and two LINCtape transports. LAP6-DIAL-MS will support one of the following:

a. One or two DF32 or FD32D Disks

b. One to four RF08 Disks

c. One RK8 Disk.

LAP6-DIAL-MS will use only one of the above; for example, the RK8 rather than a DF32, or a RK8 rather than a RF08, if more than one are installed in the system. It is not possible to operate a DF32 and a RF08 in the same system as the IOT instructions are similar.

A brief description of LAP6-DIAL-MS is provided in Paragraph 7.1.1.

6.4 MAGNETIC TAPE

6.4.1 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 with the PDP-12 three-cycle data break facility, which it uses for data transfer directly between system core memory and magnetic tape. The tape transports offer industry compatibility (or IBM compatibility) 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 memory word count (WC) and current address (CA) registers 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 two'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 instruction required for the TYPE TC58 is minimized by transferring all necessary control data (unit number, function, mode, direction, etc) from the PDP-12 accumulator (AC) to the control using IOT instructions. Similarly, all status information (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 instructions 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 μs
Operation:	The status of the error flag (EF) and the magnetic tape flag (MTF) are sampled. If either or both are set, the contents of the PC are incremented by one, skipping the next instruction.
Symbol:	If MTF or EF = 1, PC + 1 → PC

MTCR Skip on Tape Control Ready

Octal code:	6711
Event time:	1
Execution time:	4.25 μs
Operation:	If the tape control is ready to receive a command, the PC is incremented by one, skipping the next 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 μs
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:	4.25 μs
Operation:	Clears the status and command registers, and the EF and MTF if tape control is ready. If tape control is 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

MTRC Inclusive OR Contents of Command Register

Octal code:	6724
Event time:	3
Execution time:	4.25 μs
Operation:	Inclusively OR the contents of the command register into AC_{0-11}.
Symbol:	AC V command register → AC_{0-11}

MTCM Inclusive OR Contents of Accumulator

Octal code:	6714
Event time:	3
Execution time:	4.25 μs
Operation:	Inclusively OR the contents of $AC_{0-5,9-11}$ 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 μs
Operation:	Load the contents of AC_{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 μs
Operation:	Inclusively OR the contents of the status register into AC_{0-11}
Symbol:	Status Register V AC → AC_{0-11}

MTRS Read Status Register

Octal code:	6706
Event time:	2,3
Execution time:	4.25 μs
Operation:	Read the contents of the status register into AC_{0-11}
Symbol:	Status Register → AC_{0-11}

MTGO Mag Tape GO

Octal code:	6722
Event time:	2
Execution time:	4.25 μs
Operation:	Set GO bit to execute command in the command register if command is legal.
Symbol:	None

MCLA Clear the AC

Octal code:	6702
Event time:	2
Execution time:	4.25 μs
Operation:	Clear the accumulator.
Symbol:	0 → AC

Although any number of tapes may be rewinding simultaneously, 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 is 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 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 instruction 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 instructions for spacing records, SPACE FORWARD and SPACE REVERSE. The number of records to be spaced (two's complement) is loaded into the WC. CA does not need to 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 instruction, 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 instruction is given when a transport is set at BOT, the instruction is ignored, the illegal command error and BOT bits are set, and an interrupt occurs.

Read Data – Records may be read into memory in the forward mode only. Both CA and WC must be set; CA to the initial core address minus one, WC to the two'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 in such a way 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 instruction must be issued before the shut-down delay terminates. If an error occurs, the EF will be set when the MTF is set.

Write EOF – The Write EOF instruction transfers a single character (17_8) record to magnetic tape and follows it with the 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 instruction.

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 instruction does not require setting of the CA or WC. Density and parity settings are also ignored. The REWIND instruction rewinds the tape to loadpoint (BOT) and stops. Another unit may be selected after the instruction 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 instruction. The selected unit's status can be read to determine or verify that REWIND is in progress.

Continued Operation

a. To continue operating the same mode, the MTGO instruction is given before tape motion stops. The order of instructions required for continued operation are as follows:

(1) MTLC, if the instruction is to be changed.

(2) MTAF, clears only MTF and EF flags, since tape control will be in a Not Ready state.

(3) MTGO, if MTLC requested an illegal condition; the EF will be set at this time.

b. To change modes of operation, either in the same or opposite direction, the MTLC instruction is given to change the mode and a MTGO instruction 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.

c. If the WRITE function is being performed, the only forward change in command that can be given is WRITE EOF.

d. If no MTGO instruction is given, the transport will shut down in the inter-record gap.

NOTE

Flags will not be set when the control or the transport becomes ready, except if the REWIND instruction is present in the command register and the selected drive reaches BOT and is ready for a new instruction.

e. If a WRITE (odd parity) instruction 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 $\neq$ 0 (short)
9	Data request late
10	Bad tape
11	Magnetic tape flag (MTF) or job done

The register bits are equivalent in position to the AC bits (i.e., $SR_0 = AC_0$, etc).

MTF (SR11) – The MTF flag is set under the following conditions:

a. Whenever the tape control has completed an operation (after the EOR mark passes the read head).

b. 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 instruction 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) deflection 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 instruction is given, an illegal command error bit is set, causing an EF with BOT set. An interrupt then occurs.

*All references to interrupts assume the tape flags have been enabled to the interrupt (command register bit 9 = 1) and the unit is selected.

Illegal Command Error (SR3) – The illegal command error bit is set under the following conditions:

a. A command is issued to the tape control with the control not ready.

b. A MTGO instruction is issued to a tape unit which is not ready when the tape control is ready.

c. Any instruction which the tape control, although ready, cannot perform; e.g.:

(1) WRITE with WRITE LOCK condition

(2) 9-channel tape and incorrect density

(3) 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).

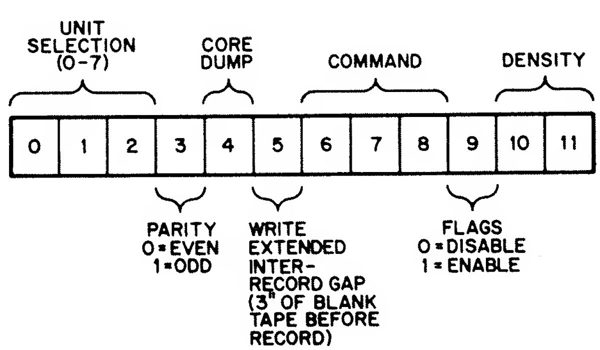
Bad Tape (SR10) – A BAD TAPE ERROR indicates detection of a bad spot on the tape. Bad tape is defined as three or more consecutively missing characters followed by data, within the period defined by the READ SHUTDOWN DELAY. The error bit is set by the tape control when this occurs. MTF and interrupt do not occur until the end of the record in which the error was detected.

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 occurs, and data transmission ceases. 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 an 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

	Bits	
Density	**10**	**11**
200 bpi	0	0
556 bpi	0	1
800 bpi	1	0
800 bpi 9 channel	1	1

Command Selection

	Bits		
Command	**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 of 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 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

6.4.2 **Magnetic Tape Transports**

The following paragraphs describe some of the Magnetic Tape Transports that are available for the PDP-12A. These machines are also compatible with the automatic Tape Control Type TC58.

6.4.2.1 **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 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 0.5 in., length 2400 ft. (1.5 mil.). Reels are 10.5 in. in diameter, IBM-compatible, with file protect (WRITE LOCK) ring.

Head: Write-read gap 0.300 in., Dynamic and skew is less than 14 microseconds.

Tape Specifications: 45 ips speed. Start time is less than 5 milliseconds. Start distance is 0.080 in. (+0.035, -0.025 in.). Stop time is less than 1.5 milliseconds. Stop distance is 0.045 in. (±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 22-1/4 in., depth 27-1/6 in., height 69-1/8 in., weight 600 lbs.

Read (Read/Compare) Shutdown Delay: 3.6 milliseconds.

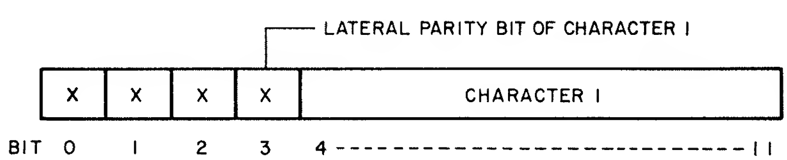
Write Shutdown Delay: Approximately 4.5 milliseconds.

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 the following format:



X- THESE BITS ARE IGNORED

Read – A word is read into memory from tape with the following format:

LATERAL PARITY BIT OF CHARACTER 1

BIT 0	1	2	3	4 – 11
X	X	X	P1	CHARACTER 1

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:

BIT 0 – 5	6 – 11
CHARACTER 1	CHARACTER 2

12-0143

This is accomplished by utilizing only 7 of the 9 tracks on the tape.

Tape written in CORE DUMP MODE must be READ (READ/COMPARE) in this same mode. These operations are the same as for a 7-track transport.

6.4.2.2 **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 one parity bit. End and loadpoint sensing compatible with IBM.

Tape: Width 0.5 in., length 2400 ft. (1.5 mil.). Reels are 10.5 in. in diameter, IBM-compatible, with file protect (WRITE LOCK) ring.

Heads: Write-read gap of 0.150 in. Dynamic and static skew is less than 14 microseconds.

Tape Specifications: 45 ips speed. Rewind time is less than 5 milliseconds. Start distance is 0.080 in. (+0.035, -0.025 in.). Stop time is less than 1.5 milliseconds. 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 22-1/4 in., depth 27-1/6 in., height 69-1/8 in. Weight 600 lbs.

Read (READ/COMPARE) Shutdown Delay: 3.6 milliseconds.

Write Shutdown Delay: Approximately 4.5 milliseconds.

6.4.2.3 Magnetic Tape Transport, Type TU20C

General Description

The TU20C connects to the PDP-12 via the TC58 Automatic Magnetic Tape Control. IOT instructions, issued by the PDP-12, control the operation of the tape transport; these instructions are decoded within the tape control unit to specify the type of tape transport operation. To write on tape, the tape control unit sends a motion-forward instruction and a write-enable instruction, and, for each character to be recorded, a record data pulse. For spacing and reading tape, the tape control unit sends the required motion instruction. Therefore, regardless of the operation, the tape is always being read and the data is always being transmitted to the tape control unit; this information is used for detecting end-of-record to terminate operation. The TU20C operates at 45 inches per second, and has the capability of recording in three densities, 200, 556, and 800 bpi; maximum transfer rate is, therefore, 800 x 45 = 36,000 six-bit characters per second. Standard seven-channel IBM-compatible tape format is used.

A nine-channel IBM-compatible version of the TU20C, designated TU20B, is also available; remaining characteristics of the TU20B are essentially the same as those for the TU20C.

Operation

Either version of the Tape Transport (TU20C, TU20B) may be operated in the local mode from the control panel or in the remote mode from the PDP-12. Local operation is selected by depressing the front panel OFF LINE control; this action causes the FORWARD, RESET, REVERSE, and REWIND switches to become operable. If either the FORWARD, REVERSE, or REWIND control is depressed, the tape moves in the specified direction until RESET is depressed, the beginning of the tape is detected for rewind, or the end of the tape is detected for FORWARD. Activating the front panel ON-LINE control places the Tape Transport in the remote mode, wherein it is controlled by the PDP-12; the Tape Transport must, however, be selected by the TC58 Tape Control.

Switches

The following switches are located on the control panel:

Switch	Function
POWER	Applies power to the tape transport.
ON LINE	Selects programmed (remote) operation by the computer.
OFF LINE	Selects local operation by the control panel.
FWD	In local operation, spaces the tape in the forward direction.
REV	In local operation, spaces the tape in the reverse direction.
REWIND	In local operation, rewinds the tape at high speed.
RESET	In local operation, terminates the space forward, space reverse, or rewind operation.
LOAD	Loads the tape into the tape column.
Unit Select Switch	Selects the tape transport unit by number. This number is used in the program to select the tape transport.

Indicators

The following indicators are provided on the control panel:

Indicators	Meaning When Illuminated
SELECT	The tape transport is selected by the tape control (or program).
READY	The tape transport is ready (vacuum on, and settle-down delay complete), no motion.
LOAD POINT	The tape is at load point.
END POINT	The tape is at end point (end of tape).
WRITE LOCK	The write lock-out ring is missing from the tape reel, which prevents the write function.
WRITE STATUS	The program has enabled the write function in the tape transport.
9	This transport is a 9-track type.
7	This transport is a 7-track type.
REWIND	The tape transport is in the rewind operation.
LOAD	The vacuum is on and the tape is loaded into the vacuum columns.
REV	Reverse operation is specified.
RESET	No motion (forward, reverse, or rewind) is specified.
FWD	Forward motion of the tape is specified.
LOCAL	Local operation by the control panel.
REMOTE	Remote operation by the computer.
POWER	Power is applied to the tape transport.

6.4.3 LINCtape Option TC12-F

The TC12-F option (prewired) in conjunction with the PRTC12-F program extends the TC12 LINCtape control to read and write DECtapes formatted on the PDP-8, PDP-9, PDP-10, and PDP-15 computers. It will operate on a standard PDP-12A system. Tape units 0 through 7 are selectable under program control. Description document DC-12-YIYA-D explains the differences between the LINCtape and DECtape formats and describes the PRTC12-F program.

Data is transferred between the tapes and the computer on the I/O bus. Two status flags enable the program to identify the information on tape. The BLOCK flag is set, indicating that a block mark was encountered on tape and the Block Number (BN) is in the tape accumulator. This number is then read into the PDP-12 AC and is processed to determine if the desired block has been found, or to initiate the necessary adjustments to access the desired block.

The WORD flag is set after four lines of data have been assembled and are ready to store in memory when reading from the tape. When writing data on tapes, the flag indicates that a 12-bit word has been written on tape and the control is ready to accept another word from the computer.

The system program PRTC12-F no. DC-12-YIYA-PB for this option is contained on the standard LAP6-DIAL system tape.

NOTE

> This option does <u>not</u> permit the user to execute tape operations using DEC tape instruction subroutines.

Instructions

IOT 6152

Octal code:	6152
Event time:	2
Execution time:	4.25 μs
Operation:	As shown below:

AC Bit	Function
4 ~~5~~(1)	Clear Block FF
5 (1)	Set Backward
6 (1)	Select unit 1
7 (1)	Set forward
9 (1)	Set motion
10 (1)	Select DECtape & AC_{11} write
SWD 0457	Skip on word flag
STB 0414	Skip on block flag
TAPE PRESET	0 → DECtape write
TAPE PRESET	0 → Motion
TAPE PRESET	Deselect DECtape

6.5 LINE PRINTERS

6.5.1 Line Printer and Control, Type LP12

The Line Printer and Control Type LP12 allows the PDP-12 computer to output data at up to 600 lines per minute (depending upon which printer is used). Either the Mohawk Data Sciences Model 4000 or 5000 series line printers can be used.

Model	Lines Per Minute	Columns Per Line
4000	up to 300	132
5000	up to 600	132

Total number of characters available: 64

A 6-bit line printer code is loaded into the LP12 buffer from AC_{6-11} with the instruction Load Printer Buffer (LLB). The Print (LPR) instruction causes the lineprinter to print the characters in the lineprinter buffer. After the buffer is filled the LPR instruction should be given to print the contents of the buffer. Two status flags indicate the line printer conditions. The Error Flag is set when a printer instruction is given if an error status exists, such as power off, paper supply low, or control circuits not reset. The Done Flag is set by a BUFF AVAILABLE pulse, generated by the line printer, after the following IOT instructions are given: Load Printer Buffer (LLB) 6654, Load Format Register and Print (LPR) 6664 and Clear Printer Buffer (LCB) 6662. The BUFF AVAILABLE pulse indicates that the line printer buffer is ready to receive another character or the lineprinter is ready to print a line. The Done Flag is cleared by the IOT instruction Clear Line Printer Flags (LCF) 6652 or the status of the Done Flag can be checked by the IOT instructions. Skip on Line Printer Done Flag (LSD) 6661.

A three-bit format register in the Printer is loaded from AC_{9-11} during a print command. This register selects one of eight channels of a perforated tape in the printer to control spacing of the paper.

The Print Flip-Flop is set when the print instruction is given and, when the Print Flip-Flop and the Done Flag are both set, a program interrupt request is generated, indicating that the printer has finished printing a line. The Print Flip-Flop can be cleared by an IOT instruction Clear Printer Buffer (LCB). AC_8 is loaded into the Space Flip-Flop during a print instruction. If the Space Flip-Flop is set, the paper is advanced according to the selected channel of the format tape in the line printer. If the Space Flip-Flop is clear, the paper advance is inhibited.

LSE Skip on Line Printer Error Flag

Octal code:	6651
Event time:	1
Execution time:	4.25 μs
Operation:	The status of the Printer Error Flag is sensed, and, if it is set, the contents of the PC are incremented by one, skipping the next sequential instruction.
Symbol:	If Printer Error Flag = 1, then PC + 1 $\rightarrow$ PC

LCF Clear Line Printer Flags

Octal code:	6652
Event time:	2
Execution time:	4.25 μs
Operation:	The Line Printer Error and Done Flags are cleared.
Symbol:	0 $\rightarrow$ Line Printer Error Flag 0 $\rightarrow$ Done Flag

LLB Load Printer Buffer

Octal code: 6654
Event time: 4
Execution time: 4.25 μs
Operation: The contents of AC_{6-11} are loaded into the Printer Buffer.
Symbol: $AC_{6-11} \rightarrow$ Printer Buffer
$AC_8 \rightarrow$ Space Flip Flop
$0 \rightarrow AC$

LSD Skip on Line Printer Done Flag

Octal code: 6661
Event time: 1
Execution time: 4.25 μs
Operation: The status of the Line Printer Done Flag is sensed, and, if it is set, the contents of the PC are incremented by one, skipping the next sequential instruction.
Symbol: If Line Printer Flag = 1, then $PC + 1 \rightarrow PC$

LCB Clear Printer Buffer

Octal code: 6662
Event time: 2
Execution time: 4.25 μs
Operation: The Printer Buffer is cleared.
Symbol: $0 \rightarrow$ Printer Buffer

LPR Load Format Register and Print Character

Octal code: 6664
Event time: 4
Execution time: 4.25 μs
Operation: The contents of AC_{9-11} are loaded into the line printer format register and the line contained in the printer buffer is printed. Paper is advanced in accordance with the selected channel of the format tape.
Symbol: $C(AC_{9-11}) \rightarrow$ Format Register
$AC_8 \rightarrow$ Space Flip-Flop
Print Line
$0 \rightarrow AC$

6.5.2 **Line Printer and Control, Type LP08**

The LP08 Line Printer interface allows control of the four models of Data Products Line Printers by the PDP-12 or positive bus PDP-8 family of computers through the use of IOT instructions. The printers are available in 80-column and 132-column printing widths with either a 64 or 96 visible character set. The device select code for the LP08 is 66_8.

The 80-column printer contains a 20-character (24-character in 132 column) buffer and prints 20 characters simultaneously when either the character buffer is full or a format control character (carriage return, paper feed, form feed) instruction is given. A paper feed or form feed command must be given to move the paper at the end of a line or overprinting will result. Characters and commands are given by loading the AC with the appropriate setting derived from the LP08 Character Code Chart (Appendix F-4) and issuing the IOT 6664 (LPC).

The three format commands in addition to initiating the print cycle are:

a. *Paper Feed* – Advances the paper one line. The perforation between pages is automatic.

b. *Carriage Return* – Returns the print position to the leftmost edge of the paper.

c. *Form Feed* – Advances the paper to a new page. Nonvisible characters are decoded as spaces if they are not control characters.

Instructions

LSF Skip on Demand Character Flag

Octal code:	6661
Operation:	If printer is ready for the next character, this flag accounts for timing required to store character in buffer, print from buffer, and advance paper.
Symbol:	If Character Flag = 1, then PC + 1 → PC

LCF Clear Character Flag

Octal code:	6662
Operation:	The line printer character flag is cleared.
Symbol:	0 → Character Flag.

LSR Skip on Not Ready

Octal code:	6663
Operation:	The next instruction is skipped if the error status flag is set. Such errors as out-of-paper, drum gate open, paper jam, etc. will set error status flag.
Symbol:	If Error Flag = 1, then PC + 1 → PC

LLC Load Buffer from AC

Octal code:	6664
Operation:	Load the character into the Print Buffer and print if buffer is full or character was a control function. This instruction does not clear the AC.
Symbol:	AC → Print Buffer

LIN Set Program Interrupts Enable

Octal code:	6665
Operation:	Sets the device interrupt flip-flop to initiate program interrupt when the demand character flag is set or an error condition exists.
Symbol:	1 → Int Enable

LPC Load Buffer from AC and Clear Flag

Octal code:	6666
Operation:	The print buffer is loaded from the AC and the character flag is cleared. Microprogram combination of LCF and LLC.
Symbol:	AC → Print Buffer, 0 → Character Flag.

LCP Clear Program Interrupt Enable

Octal code:	6667
Operation:	The program interrupt enable is cleared.
Symbol:	0 → Int Enable

Print rate

80 column model	
64 character	356 Lines/minute, columns 1-80 460 Lines/minute, columns 1-60 650 Lines/minute, columns 1-40 1110 Lines/minute, columns 1-20
96 character	253 Lines/minute, columns 1-80 330 Lines/minute, columns 1-60 478 Lines/minute, columns 1-40 843 Lines/minute, columns 1-20
132 column model	
64 character	245 Lines/minute, columns 1-132 290 Lines/minute, columns 1-110 356 Lines/minute, columns 1-88 460 Lines/minute, columns 1-66 650 Lines/minute, columns 1-44 1110 Lines/minute, columns 1-22
96 character	173 Lines/minute, columns 1-132 205 Lines/minute, columns 1-110 253 Lines/minute, columns 1-88 330 Lines/minute, columns 1-66 478 Lines/minute, columns 1-44 843 Lines/minute, columns 1-22
Format	Top-of-form control, single line advance, and perforation step over.
Paper Feed	One pair of pin-feed tractors for 1/2 inch hole center, edge-punched paper. Adjustable for any paper width from 4 inches to 9-7/8 inches on the 80 column model; or a maximum width of 14-7/8 inches for the 132 column model.
Paper slew speed	13 inches per second

6.6 CARD READERS

6.6.1 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 by a photoelectric process. 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. Column information can be read in one of two program selectable modes, alphanumeric or binary. In the alphanumeric mode, the 12 information bits in one column are automatically decoded and transferred into the least significant half of the AC as a 6-bit Hollerith code. In the binary mode, the 12 bits of a column are transferred directly into the AC so that the top row (12) is transferred into AC_{00} and the bottom row (9) is transferred into AC_{11}. A punched hole is interpreted as a binary 1, and the absence of a hole is interpreted as a binary 0.

Three program flags indicate card reader conditions. The data ready flag sets and a program interrupt is requested when a column of information is ready to be transferred into the AC. A read alphanumeric or read binary instruction must be issued within 1.4 milliseconds after the data ready flag is set to prevent data loss. The card done flag is set and a program interrupt is requested when the card leaves the read station. A new select instruction must be issued immediately after this flag is set 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 a check for data loss. The reader-not-ready flag can be sensed by a skip instruction to provide an 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 the flag is set, the reader cannot be selected and select instructions are ignored. The reader-not-ready flag is not connected to the program interrupt facility, and cannot be cleared under program control. Manual operation is required to clear the reader-not-ready flag. Instructions for the CR12 are:

RCSF Skip on Data Ready

Octal code:	6631
Event time:	1
Execution time:	4.25 μs
Operation:	The status of the data ready flag is sensed, and if it is set (indicating information for one card column is ready to be read) the contents of the PC are incremented by one skipping the next sequential instruction.
Symbol:	If Data Ready Flag = 1, then PC + 1 $\rightarrow$ PC

RCRA Read Alphanumeric

Octal code:	6632
Event time:	2
Execution time:	4.25 μs
Operation:	The 6-bit Hollerith code for the 12 bits of a card column is transferred into AC_{6-11} and the data ready flag is cleared.
Symbol:	AC_{6-11} V Hollerith $\rightarrow AC_{6-11}$ 0 $\rightarrow$ Data Ready Flag

RCRB Read Binary

Octal code:	6634
Event time:	3
Execution time:	4.25 μs
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 rows 12, 11, and 0 enter AC_{0-2} and card rows 1 through 9 enter AC_{3-11} respectively.
Symbol:	0 $\rightarrow$ Data Ready Flag

RCSD Skip on Card Done Flag

Octal code:	6671
Event time:	1
Execution time:	4.25 μs
Operation:	The status of the card done flag is sensed, and if it is set (indicating that the card has passed the read station) the contents of the PC are incremented, skipping the next 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 μs
Operation:	The status of the card reader flag is sensed and if the reader is ready, the PC is incremented skipping 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 Ready Flag = 1, then PC + 1 → PC 0 → Card Done Flag

RCRD Clear Card Done Flag

Octal code:	6674
Event time:	3
Execution time:	4.25 μs
Operation:	The card done flag is cleared. This instruction allows a program to stop reading at any point in the card deck.
Symbol:	0 → Card Done Flag

A logical instruction sequence to read cards is:

```
START,    RCSE          /START CARD MOTION AND SKIP IF READY
          JMP 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       /YES. READ ONE CHARACTER OR ONE
          RCRB          /COLUMN AND CLEAR DATA READY FLAG
          DCA I 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
```

The CR12 does not perform validity checking, although a programmed validity check can be made by reading each card column in both the alphanumeric and binary mode (within the 1.4 millisecond time limitation), then performing a comparison check.

The following discussion and controls and indicators deal with the General Design Industries (GDI) Model 100 card reader. Other card readers can be used and will have similar controls and indicators.

Before commencing a card reading program, load the input hopper with cards and press Motor Start and Read Start pushbuttons. The functions of the manual controls and indicators are as follows (as they appear from left to right):

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 illuminated. 4. Internal power supply is not operational.
C – READ START	Momentary action pushbutton, with separate indicator. Causes ready line to go high, enabling card reading under control of the external read instructions. If read command is open or high, card reading begins immediately at full rated speed.
D – READ STOP	Momentary action pushbutton with indicator. Stops card reading if depressed without stopping drive motor. However, READ STOP light can indicate a stopped motor or a ready line low condition.
E – INDICATORS	Several detection circuits are incorporated in the card reader. Whenever any red indicator lights, the drive motor is stopped after 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 immediately inhibited.
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	When approximately 400 cards are in the stacker hopper, indicator light lights.

6. SYNC FAIL Indicator

SYNC FAIL is indicated if the sync signal is lost. Internal timing signals are derived from an oscillator which is synced to the track speed.

7. LIGHT CHECK INDICATOR

The photocells must always be illuminated except during the time a card is being read. The LIGHT CHECK detector is inhibited each time a card enters the read station until position (count of) 84 is reached. If a card fails to leave the read station by this time, a LIGHT CHECK is indicated.

6.6.2 **Optical Mark Card Reader Type CM12**

The GDI Model 100-MS is an optical-mark card reader that reads (12-row Hollerith) reflective data cards of various format designs at a rate of up to 200 cards per minute by a photoelectric process. Cards are read column by column, beginning with column 1. A single select instruction will cause the reader to feed and read a card. Once a card is in motion, all columns are read. Column information is read in one of two program-selectable modes, alphanumeric or binary. In the alphanumeric mode, the 12 information bits in one column are automatically decoded and transferred into the least significant half of the AC as a 6-bit Hollerith code. In binary mode, all 12 bits of a column are transferred directly into the AC so that the top row (12) is transferred into AC_{00} and the bottom row (9) is transferred into AC_{11}. A punched hole or a nonreflective spot (either nonreflective ink or No. 2 pencil) is interpreted as a binary 1, and the absence of a hole or reflective spot is interpreted as a binary 0.

Instructions

The instruction set associated with the optical mark reader is identical to that of the CR12 in Paragraph 6.6.1.

Characteristics

Size: The complete unit is 14 in. wide, 18 in. deep, and 18 in. high. The card deck is tilted back at a 45-degree angle.

Weight: Complete unit weighs 47 lbs.

Card Rate: 200 per minute.

Input Power: 115 VAC ±10 VAC, 60 ±5 Hz single phase, at 300 VA maximum.

Card Specification: The card reader is designed to read 7-3/8 in. x 3-1/4 in. optical mark cards conforming to the material and size requirements of EIA Standard RS-292 Media I. Format and printing requirements are specified in the DEC Mark Sense Card Specification.

Card Capacity: Both input hopper and output stacker hold 450 cards. Cards may be added or removed during reader operation.

Environment:	Operating:	32° to 120°F ambient 15% to 80% relative humidity
	Storage:	30° to 150°F ambient 0% to 100% relative humidity

Controls and Indicators

Power Switch: An alternate action rocker-type switch used to apply AC power to the Card Reader.

Start Switch: A momentary pushbutton switch used to condition the unit to read cards. When it is depressed, the drive motor will start, and any error indicators will be reset if the error has been cleared. When it is released and the motor has reached operating speed, the reader may accept a "read command' and process cards.

Stop Switch: A momentary pushbutton switch used to stop the reader. When it is depressed, the motor will stop; if it is depressed during a time a card is in process, the card cycle will be completed before the motor stops.

Power: A green indicator to verify AC power on; mounted next to the POWER switch.

On Line: A green indicator to verify that the START switch has been depressed and the unit is ready to operate. Light will remain on until the STOP switch is depressed or an error condition is sensed; mounted next to START switch.

Cards: A red indicator to identify an input hopper empty or an output stacker full condition.

Feed Error: A red indicator to identify when a card has not been fed from the input hopper to the read station at the end of a feed cycle.

Stacker Error: A red indicator to identify when a card has not been properly delivered to the output stacker.

Motion Error: A red indicator to indicate a card jam in the read station.

L.D. Error: A red indicator to identify when a Light or Dark Check of the read station was not met.

6.7 INCREMENTAL PLOTTERS

6.7.1 Incremental Plotter and Control, Type XY12

Calcomp (California Computer Products) Models 563 and 565 available in three step-size models and four models of the Complot (Houston Instruments) digital plotters can be operated from a Digital Equipment Corporation Type XY12 Incremental Plotter Control. The characteristics of the recorders are summarized:

Name	Model	Paper Width (inches)	Speed (step/minute)	Step Size
Calcomp	563	30	12,000 200 Hz	.01-in. .005-in. .1-mm
	565	12	18,000 300 Hz	.01-in. .005-in. .1-mm
Complot	DP-1-1	12	18,000	.01-in.
	DP-1-5	12	18,000	.005-in.
	DP-1-M2	12	18,000	.25-mm
	DP-1-M1	12	18,000	.1-mm

The principles of operation are basically the same for each of the 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:

PLSF Skip on Plotter Flag

Octal code:	6501
Event time:	1
Execution time:	4.25 μs
Operation:	If the Plotter Flag is set, the contents of the PC are incremented by one so that the next sequential instruction is skipped.
Symbol:	If Plotter Flag = 1, then PC + 1 $\rightarrow$ PC

PLCF Clear Plotter Flag

Octal code:	6502
Event time:	2
Execution time:	4.25 μs
Operation:	Clear the AC and Plotter Flag
Symbol:	0 $\rightarrow$ AC 0 $\rightarrow$ Plotter Flag

PLPU   Pen Up

Octal code:	6504
Event time:	3
Execution time:	4.25 μs
Operation:	Raise the plotter pen from the paper surface.
Symbol:	None

PLPR   Pen Right

Octal code:	6511 (May be microprogrammed with PLDU or PLDD.)
Event time:	1
Execution time:	4.25 μs
Operation:	Move the plotter pen to the right in either the raised or lowered position.
Symbol:	None

PLDU   Drum Up

Octal code:	6512 (May be microprogrammed with PLPR.)
Event time:	2
Execution time:	4.25 μs
Operation:	Move the plotter paper drum upward. This instruction can be combined with the PLPR and PLPD commands.
Symbol:	None

PLDD   Drum Down

Octal code:	6514 (May be microprogrammed with PLPR.)
Event time:	3
Execution time:	4.25 μs
Operation:	Move the plotter paper drum downward.
Symbol:	None

PLPL   Pen Left

PLPD ?

Octal code:	6521 (May be microprogrammed with PLUD, PLAD, or both.)
Event time:	1
Execution time:	4.25 μs
Operation:	Move the plotter pen to the left in either the raised or lowered position.
Symbol:	None

PLUD   Drum Up

Octal code:	6522 (May be microprogrammed with PLPL, PLPD, or both.)
Event time:	2
Execution time:	4.25 μs
Operation:	Move the plotter paper drum upward. This instruction is similar to the 6512 instruction except that it can be combined with the PLPL or PLPD instructions.
Symbol:	None

PLPD Pen Down

Octal code:	6524 (May be microprogrammed with PLPL, PLUD, or both.)
Event time:	3
Execution time:	4.25 μs
Operation:	Lower plotter pen to the surface of the paper.
Symbol:	None

Program sequence assumes 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 the 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.

6.8 PAPER TAPE

6.8.1 High-Speed Paper-Tape Punch and Reader, Type PC12

The High-Speed Paper-Tape Punch and Reader Type PC12 provides the user with a faster means of inputting or outputting information from paper tape to the PDP-12 than that provided by the standard Teletype tape reader/punch. The PC12 is functionally a PP12, High Speed Paper Tape Punch, and a PR12, High Speed Paper Tape Reader, configured mechanically in the same unit. The operating characteristics are discussed in the following paragraphs.

6.8.2 High-Speed Paper-Tape Punch, Type PP12

The High-Speed Paper-Tape Punch option Type PP12 consists of a PC05-P paper-tape punch that perforates 8-hole fanfold paper-tape at a rate of 50 characters per second. Information to be punched in tape is loaded in an 8-bit punch buffer (PB) from AC_{4-11}. The punch flag is set at the completion of the punching action, signaling that new information may be transferred into the punch buffer and punching may be initiated. The control circuitry for this device is located in the BA12 Peripheral Expander. 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 μs
Operation:	The punch flag is sensed, and, if it is set, the contents of the PC are incremented by one, skipping the next sequential instruction.
Symbol:	If Punch Flag = 1, the PC + 1 → PC

PCF Clear Punch Flag

Octal code:	6022
Event time:	2
Execution time:	4.25 μs
Operation:	The punch flag and the punch buffer are both 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 μs
Operation:	An 8-bit character is transferred from AC_{4-11} into the punch buffer, and then the character is punched. This instruction does not clear the punch flag or the punch buffer.
Symbol:	AC_{4-11} → PB → PB

PLS Load Punch Buffer Sequence

Octal code:	6026
Event time:	2,3
Execution time:	4.25 μs
Operation:	The punch flag and punch buffer are both cleared, the contents AC_{4-11} are 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. Combines PCF and PPC.
Symbol:	0 → Punch Flag, PB AC_{4-11} → PB 1 → Punch Flag when done

A program sequence loop to punch characters 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
          JMP FREE
```

6.8.3 **High-Speed Paper-Tape Reader, Type PR12**

The High-Speed Paper-Tape Reader (PC05-R) option Type PR12 senses data in a 8-hole perforated-paper tape (unoiled) 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 into the Reader Buffer from the perforated tape. The Reader Buffer is transferred into AC_{4-11} under program control. The Reader Flag is connected to the program interrupt and instruction skip facilities, and is cleared by IOT pulses. Control circuitry for this device is located in the BA12 Peripheral Expander. Computer instructions for the reader are:

RSF Skip on Reader Flag

Octal code:	6011
Event time:	1
Execution time:	4.25 μs
Operation:	The status of the Reader Flag is sensed, and, if it is set, the contents of the PC are incremented by one, skipping the next sequential instruction.
Symbol:	If Reader Flag = 1, then PC + 1 → PC

RRB Read Reader Buffer

Octal code:	6012
Event time:	2
Execution time:	4.25 μs
Operation:	The contents of the reader buffer are transferred into AC_{4-11} and the Reader Flag is cleared. This instruction does not clear the AC.
Symbol:	RB V $AC_{4-11} \rightarrow AC_{4-11}$ 0 → Reader Flag

RFC Reader Fetch Character

Octal code:	6014
Event time:	3
Execution time:	4.25 μs
Operation:	The Reader Flag and the reader buffer are both cleared. A 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 characters from perforated tape can be written as follows:

```
LOOK,     RFC                 /FETCH CHARACTER FROM TAPE
          RSF                 /SKIP WHEN RB FULL
          JMP LOOK
          CLA
          RRB                 /LOAD AC FROM RB
          JMP LOOK
```

6.9 DATA BUFFERS

6.9.1 Data Buffer Type DB12-P, N

The DB12 option is an input/output transfer register, consisting of a 12-bit input bus driver and a 12-bit output buffer register. The basic logic for this option is contained on an M735 module, which is described in the Logic Handbook (1970). Three DB12 options are prewired in the BA12 Peripheral Expander. The device select gating and data lines provide the capability for transferring data into or out of the PDP-12 accumulator. The user selects a device code by inserting jumpers in the M921 Device Code Select Jumper module located in row A, slot 19, of the BA12 Peripheral Expander. Two I/O cables (Flexprint®) terminating in M903 connector modules are provided to connect the DB12 to the external I/O device. The output register is buffered to provide either positive (0, +3V) or negative (0, -3V) logic levels. The option designation is DB12-P (positive) or DB12-N (negative).

® Flexprint is a Registered Trademark of Sanders Associates Inc.

6.10 POWER FAIL/RESTART

6.10.1 Power Failure Option, Type KP12

General

The KP12 Power Failure Option protects an operating program upon failure or interruption of the computer's primary power source. In the event of a power abnormality, a program interrupt is initiated by the KP12 and enables continued operation of the central processor for 1 millisecond. During this interval, the interrupt routine identifies the power low condition as the initiator of the interrupt. The interrupt routine then stores the contents of active registers (AC, L, MQ, etc) and the program counter in known core memory locations. Upon restoration of power, the power low flag is cleared and a routine in the 8 mode beginning at address 0000_8 starts automatically, restoring the contents of the active registers and the program counter, and then continues the interrupted program.

Operation

A manual RESTART switch enables or disables the automatic restart operation upon restoration of primary power. When it is ON (down), the program counter is cleared and a signal which simulates the console START key (RCL START PC) is produced 200 milliseconds after power conditions become satisfactory. Operation is restarted (always in the 8 mode) by executing the instruction contained in address 0000_8; this instruction is a JMP to the starting address of a subroutine which restores the contents of the active registers and the program counter to the conditions that existed prior to the power low interrupt. The 200-millisecond delay assures that slow mechanical devices, such as Teletype equipment, have completed any previous operation before the program is resumed.

When the RESTART switch is OFF (up), the power low flag is cleared upon the return of normal power conditions, but the program must be manually restarted, possibly after resetting peripheral equipment.

Programming

SPL Skip on Power Low

Octal code:	6102
Event time:	2
Execution time:	4.25 μs
Operation:	The condition of the power low flag is sampled; if set (indicating a power failure has occurred), the contents of the PC are incremented by one, skipping the next sequential instruction.
Symbol:	If Power Low Flag = 1, then PC + 1 $\rightarrow$ PC

Figures 6-9 and 6-10 illustrate the Automatic Restart Program Events and Typical Power Failure Program Service Routine, respectively.

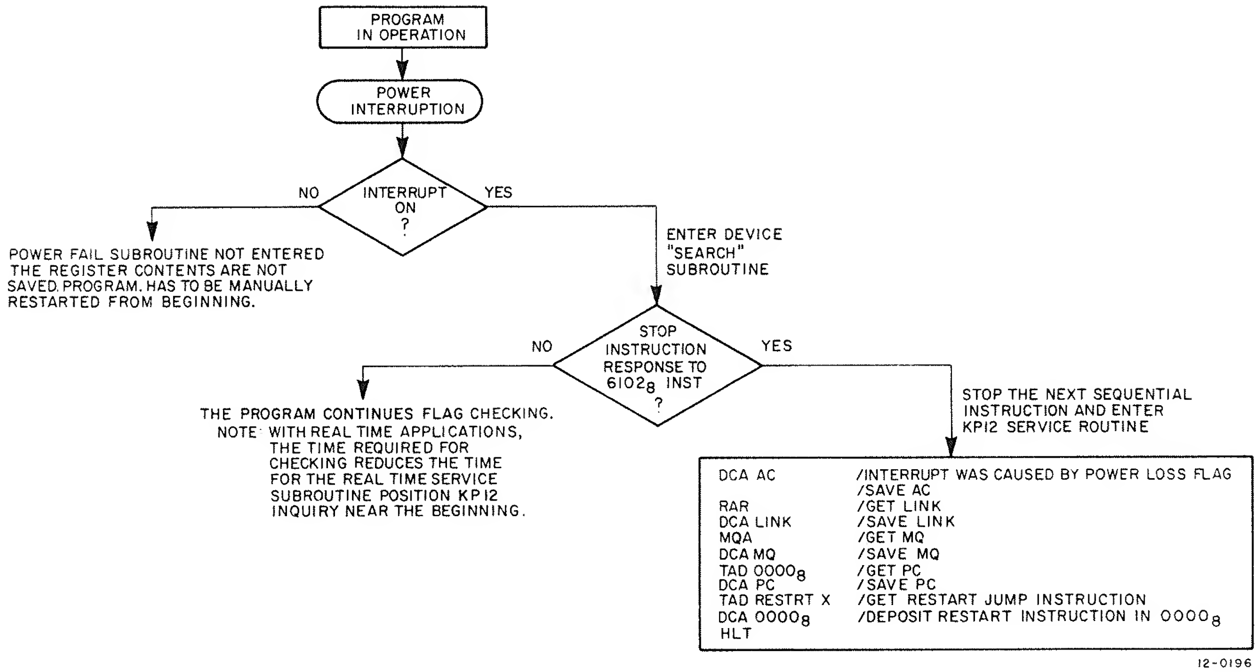


Figure 6-9. Automatic Restart Program Events

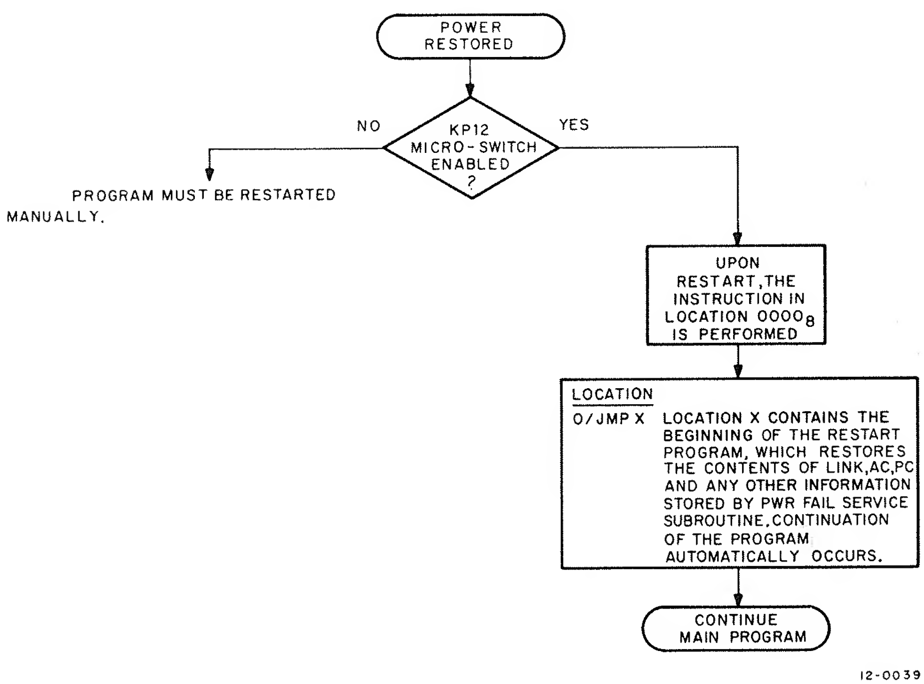


Figure 6-10. Typical Power Failure Program Service Routine

6.11 ANALOG-TO-DIGITAL

6.11.1 General Purpose Multiplexed Analog-to-Digital Converter System, Type AF01-A

The Type AF01-A General-Purpose Multiplexed Analog-to-Digital Converter combines a versatile, multipurpose converter with a multiplexer to provide a fast, automatic, multichannel scanning and conversion capability. It is intended for use in systems that sample and process analog data from sensors or other external signal sources at high rates of speed. The AF01-A is used when greater accuracy than that provided by the standard AD12 A-D converter is needed. The Type AF01-A 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 percent, with an additional quantization error of half the analog value of the least significant bit (LSB).

A/D Converter Specifications

The Type AF01-A 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 6-6 (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 6-5).
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 two'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.

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 ns. The Type A400 may also be used to scale the signal input to accept ±10V, 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 6-4.) ← Table 6-5?

pg 6-35!!

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-switching 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 AF01-A System. Specifications (measured at input connector) are as follows:

---

*Conversion rate = [(35 + 2) (10 - 6) (64)] -1 = 420 cycles/sec.
= [(9 + 2) (10 - 6) (64)] - 1 = 1420 cycles/sec.

Input Operating Signal Voltages:	+10V to -10V
Current:	1 mA
On Resistance	500 ohm (max)
Voltage Offset	1 μV
"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.

Table 6-5. Input Signal Scaling

Configuration	Gain	Input Signal	Input Impedance (ohms)	Input Output	Option Designation
Standard		0 -5 -10	10 meg. 10 meg. 10 meg.	4000_8 0000_8 3777_8	STD
Sample & Hold	-1 -1 -1	+5 0 -5	10K 10K 10K	3777_8 0000_8 4000_8	AHO2
Sample & Hold	-1/2 -1/2 -1/2	+10 0 -10	10K 10K 10K	3777_8 0000_8 4000_8	AHO2
Amplifier	+1 +1 +1	+5 0 -5	>100 meg. >100 meg. >100 meg.	4000_8 0000_8 3777_8	AHO3
Amplifier	+1/2 +1/2 +1/2	+10 0 -10	>100 meg. >100 meg. >100 meg.	4000_8 0000_8 3777_8	AHO3
Amplifier and Sample & Hold	-1 or -1/2	+5 +10 0 or 0 -5 -10	>100 meg. >100 meg. >100 meg.	3777_8 0000_8 4000_8	AHO3 & AHO2

NOTE

Unipolar signals (0 to +5, or 0 to +10v) may also be specified with either the AHO3 or AHO2 option.

System 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 instruction is received. After indexing through the maximum number of channels implemented, the address is returned to 0.

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. A separate A/D Convert I/O Transfer instruction may also initiate one or more conversions on a currently selected channel.

A/D conversion times are increased by 2 microseconds 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 microseconds when AH03 is used. These items are added to the conversion times shown in Table 6-6 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 microseconds. (See A400 specifications.)

A/D CONVERTER/MULTIPLEXER CONTROLS

Designation	Function
WORD LENGTH:	Rotary switch selects digital word length or conversion accuracy. Refer to Table 6-6 for corresponding conversion times.
POWER ON/OFF:	Applies 117 Vac power to internal power supplies.
CLR:	Clears 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.

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, 53_8 and 54_8, depending upon whether conversion or multiplexing function is being selected. The converter/multiplexer interprets the device selection code to enable execution of the IOP command pulse generated by the IOT instruction.

Table 6-6. System Conversion Characteristics

Word Length (No. of Bits)	Max Switching Point Error*	Selected Channel (A/D) Conversion Time (μs)**	Random or Sequential (MPX & A/D) Conversion Time (μs)**	AH03 MPX A/D Conversion Time (μs)**	AH02 MPX A/D Conversion Time (μs)**	AH02 AH03 MPX & A/D Conversion Time (μs)**
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

*±1/2 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.

ADSF Skip on A-D Flag

Octal code: 6531
Event time: 1
Execution time: 4.25 μs
Operation: The A-D converter flag is sensed, and if it is set (indicating that the conversion is complete) the contents of the PC are incremented by one, skipping the next instruction.
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 6-6.
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. The number of binary bits in the digital-value word and the accuracy of the word are 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 μs
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 instruction 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 μs
Operation:	The channel address register (CAR) of the multiplexer is cleared in preparation for setting of a new channel.
Symbol:	$0 \rightarrow CAR$

ADSC Set Multiplexer Channel

Octal code:	6542
Event time:	2
Execution time:	4.25 μs
Operation:	The channel address register of the multiplexer is set to the channel specified by AC_{6-11}. 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 μs
Operation:	The contents of the channel address register of the multiplexer are incremented by one. If the maximum address is contained in the register when this instruction is given, the minimum address (00) is selected.
Symbol:	$CAR + 1 \rightarrow CAR$

The converter/multiplexer may be operated by the 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
CLA           /CLEAR AC
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 AF01-A

The AH03 consists of a DEC amplifier (part #1505379-10) 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 MHz
Full output voltage	1 MHz
Slewing rate	100V/μs
Overload recovery	50 μs
Input voltage offset	Adjustable to 0
Avg vs temp	20 μv/°C
Vs supply voltage	15 μv/%
Vs time	10 μv/day
Input current	50 pA max
Avg vs temp	doubles every 10°C
Vs supply voltage	10 pA/%
Input impedance (ohms)	
Between inputs	10^{10} (5 pF shunt c)
Common mode	10^{10} (5 pF shunt c)
Input Voltage	±15V
Max common mode	±10V
Common mode rejection	50,000V
Power	
Voltage	±15 to 16V
Current at rated load	40 ma
A400 (standard gain options)	
Acquisition time to 0.01% (full-scale step)	<12 μs
Aperture time	<150 ns
Hold inaccuracy (droop)	<1 mv/ms
Temperature coefficient	0.1 mv/ms/°C
Gain (negative)	1.0 or 0.5
Input range (volts)	±5.0 ±10.0
Output Impedance	<1.0 ohm

6.12 DIGITAL-TO-ANALOG

6.12.1 Digital-to-Analog Converter, Type AA01-A

The general-purpose Digital-to-Analog Converter Type AA01-A converts 12-bit binary numbers into analog voltages. The basic option consists of three channels, each containing a 12-bit digital buffer register and a digital-to-analog converter (DAC). A common digital input to all three registers is provided by a 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 contents of the associated register, providing an analog output voltage.

Type AA01-A 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 is operated by a single IOT command. Select codes of 55, 56, and 57 are assigned to the AA01-A, making it possible to operate nine single-buffered channels or various configurations of double-buffered channels. A typical instruction for the AA01-A is:

DALI Load Digital-to-Analog Convert 1

Octal code:	6551
Event time:	1
Execution time:	4.25 μs
Operation:	The contents of the accumulator are loaded into the digital buffer register of channel 1.
Symbol:	AC → DAC1

CHAPTER 7
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 equipped with the necessary 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. This chapter is divided into four sections: PDP-12 Programs, PDP-8 Programs, DECUS Programs, and Diagnostic Programs. This list is representative and subject to change. The Digital Program Library and the DECUS Program Library maintain the current program lists.

7.1 PDP-12 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, and eliminating many commonly experienced initial programming delays.

7.1.1 LAP6-DIAL Display Interactive Assembly Language (DEC-12-SE2D-D)

LAP6-DIAL provides the PDP-12 user with an operating system that includes editing, assembling, and data handling capabilities. An interactive display permits quick user response; a File Index and Peripheral device Interchange Program (PIP) facilitate file manipulation. The minimum hardware configuration for using LAP6-DIAL is a 4K PDP-12B system. (The character editing facility is, however, designed primarily around the use of the AD12 Analog-to-Digital Converter and Multiplexer for the PDP-12A.) In addition, the system will utilize an additional 4K of memory to improve its efficiency significantly.

The LAP6-DIAL system is provided to the user on LINCtape. Each tape contains a reserved area occupied by LAP6-DIAL, a working area for temporary storage of user programs, and a file area for permanent storage of user programs. The LAP6-DIAL area of the tape contains the Editor, Assembly and Utility Programs, and a File Index. The Index stores the name, starting block number, and length of each stored file. User programs are saved as named files in the file area of the system tape. The scope is used as a moving window to view source programs in the working area. Up to 17 lines with up to 40 characters per line can be displayed at a time on the scope (maximum of 256). In the edit mode, any portion of a program in the Working Area can be displayed by an appropriate locate request.

A LINCtape containing LAP6-DIAL is designated as the system tape, and is assigned to tape unit 0. Some operations may be performed with only one LINCtape containing LAP6-DIAL, but many procedures, such as assembling programs, require two tapes. Most efficient operation is achieved when both tapes contain LAP6-DIAL.

On startup the system automatically enters the edit mode. A source program may be typed in via the TTY keyboard. The program will reside in the Working Area and will be displayed on the CRT Display character-by-character as it is entered. The LAP6-DIAL Editor may be used at this time to add, modify, or delete characters, lines, or large sections of the program. A command may also be issued via the TTY keyboard to the Monitor. When called, the Monitor writes out its buffer pointers and is replaced by the called program. When the system program operation is completed, the Monitor is automatically called back into core and it retrieves its buffer pointers.

The Monitor Commands are summarized in the following table. Items in parentheses are optional; if they are omitted, the user's program that was most recently manipulated is used.

Commands			Functions
AS	(N, U)		Assemble (U = 0, 1)
LO	(N, U)		Load Binary
LI	(L, L,)	(N, U)	Assemble & List (U - 0, 1)
QL	(L, L,)	(N, U)	Assemble & Quick List (U = 0, 1)
PS	(L(,L),)	(N, U)	Print Source
SB	N, U (,M[A])		Save Binary
SP	N, U		Save Program (Source)
AP	(L, L,) N, U or B, U		Add Program (Source)
DX	(, U)		Display Index
PX	(, U)		Print Index
CL			Clear Working Area
PI			Peripheral Interchange Program
EX			Exit
MC	X(Y), U		User's Monitor Command (READS IN USER PROGRAM FROM FREE AREA)

Legend:

N = File Name	M = Mode (L for LINC or P for PDP-8)
U = Tape Unit	A = Address (5 digits-used only if mode is specified)
L = Line Number	B = Tape Block Number

The assembler processes both LINC mode and 8 mode statements, assembling programs up to 8K in length. Six-character alphanumeric symbols can be defined; source listings, assembly listings, or abbreviated assembly listings may be optionally obtained on the Teletype or, if provided, the Line Printer. To facilitate the preparation of system programs and large programs, conditional assembly pseudo-operations are provided along with a facility to preserve the tag table definitions from one assembly to another assembly (or to several assemblies).

Upon loading, programs may be started automatically in either LINC or 8 mode at any memory location.

A second version of LAP6-DIAL, LAP6-DIAL-MS, provides the user of the PDP-12 (with 8K of memory and a disk) with a fully-integrated tape-disk system. The additional facilities include:

1. Mass storage capabilities to support DF32, RS08, or RK8 disks (or LINCtape).

2. Monitor commands to clear the Binary Working Area and to merge binary files.

3. I/O routines to read, write, or move data.

4. An automatic system built to specialize the system tape for the user's present configuration.

5. Increased assembler facilities for processing large programs.

7.1.2 Peripheral Interchange Program (PIP)

The Peripheral Interchange Program provides a flexible means of transferring data among peripheral devices such as LINCtape, Teletype, High Speed Paper-Tape Reader/Punch, Line Printer, Disk, and Card Reader. Symbolic and binary files, as well as absolute data, are processed in response to scope-directed operator requests.

7.1.3 QUANDA (DEC-12-FISA-D)

QANDA is a subroutine which allows a user to display textual information on the CRT Display, ask questions of the viewer, allow editing of the input, and receive answers.

7.1.4 DATAM* (DEC-L8-FDAA-D)

The DATAM program retrieves, displays, and stores individual data blocks from LINCtape 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.

7.1.5 GRAPHA* (DEC-L8-UGAA-D)

The GRAPHA program allows data to be retrieved from LINCtape and displayed, and allows 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 LINCtape for future reference.

7.1.6 FRQANA* (DEC-L8-FANA-D)

The FRQANA 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.

7.1.7 MAGSPY (DEC-12-UZSA-D)

The MAGSPY program provides a moving window for scanning data stored on LINCtape. The data is displayed on the scope and can be scanned at a rate determined by a potentiometer setting. The data is interpreted either as a binary point plot or as packed ASCII characters depending on a switch setting.

7.1.8 COMPAR* (DEC-L8-EUCA-D)

The COMPAR program compares contents of specified LINCtape blocks on a word by word basis.

7.1.9 SEARCH* (DEC-L8-EUSA-D)

The SEARCH program performs a search of blocks of LINCtape for a specific word. The user may specify the word to be searched and a mask of bits for comparison.

7.1.10 CONVERT (DEC-12-ESYB-D)

The CONVERT program is used to convert symbolic text from LAP-4 and LAP-6 manuscripts for compatibility with the LAP6-DIAL Assembler.

*LINC-8 Programs.

7.1.11 MARK 12 (DEC-12-YITB-D)

The MARK 12 program is used to format tapes to be used with the PDP-12. Several format options are available and, by using the subroutines within MARK 12, the user can generate a tape of arbitrary format.

7.1.12 L8SIM (DEC-12-SI1B-D)

The LINC-8 Simulator Trap Processor handles Teletype input and output for LINC-8 and classic LINC programs so they can be run on the PDP-12 without modifications. It must be loaded into the PDP-12 core memory with any LINC-8 or classic LINC program which uses the keyboard or the Teleprinter, in order for that program to be run on the PDP-12.

The trap processor operates by using the PDP-12 Instruction Trap Facility to detect execution of either of the two LINC-8 Teletype input/output instructions by the user's program. It responds to user's execution of a Teletype instruction by executing coding to simulate the instruction's LINC-8 or classic LINC effect. After simulation of the instruction, the trap processor returns control to the user program.

Users may easily adapt the LINC-8 Simulator Trap Processor to handle other devices or for their own purposes. Explicit instructions for a number of useful adaptations are provided in this document, along with enough information on the internal operation of the program to permit users to easily implement adaptations of their own invention.

An important limitation of the trap processor is that it is not interruptible. It may not be operated when the PDP-12 Program Interrupt facility is enabled.

7.1.13 FRED (DEC-12-FZFA-D)

The File Replacement, Entry, and Deletion subroutine processes the LAP6-DIAL Index for the user, freeing him from the clerical function of maintaining the file entries. A second version Mildred (DEC-12-FZDA-D) performs these operations in a disk based system.

7.1.14 PRTC 12-F (DEC-12-YIYA-D)

PRTC12-F is a PDP-12 program to utilize the TC12-F tape hardware option in transferring data between LINCtape and DECtape. The DECtape may have been formatted on a PDP-8, PDP-8/I, PDP-9, PDP-10, and PDP-15. For a complete description of the TC12-F hardware see Section 6.4.3.

7.1.15 SIGAVG/SINPRE (DEC-12-UZ1A-D/DEC-12-UW4A-D)

SIGAVG is a multisweep signal averager that allows the user to enhance signals with a low signal/noise ratio and display them on the CRT Display. SIGAVG will sample at rates ranging from 55 to 4095 μs per point per instrument, supporting a maximum of five instruments. It can take up to 4096 sweeps, and can output averaged results to LINCtape. SINPRE converts the output of SIGAVG (two-word) to the commonly used one-word format.

7.1.16 CATACAL (DEC-12-UW1A-D)

CATACAL is a "boxcar" averager and data manipulation program that acquires data from an external instrument at rates that range from .2 ms to 35 seconds per point. CATACAL is capable of reading and writing on LINCtape; it can output one or two data files (spectra) to either the CRT display or an X-Y recorder. It can also differentiate, integrate, edit (strip), and compare data files (spectra), and display the results on the scope. It has the capability of curve-fitting and deconvolution, using Lorentzian or Gaussian equations.

7.1.17 NMRSIM (Nuclear Magnetic Resonance Simulation) (DEC-12-UW5A-D)

NMRSIM allows the user to calculate theoretical spectra of a wide variety of compounds. The user inputs the appropriate parameters from the keyboard (such as spin, chemical shifts, and coupling constants), and calculated line-spectra are then displayed on the scope. NMRSIM can output spectra to LINCtape and can also read, merge, and display a series of spectra from LINCtape which effectively simulate large spin systems or mixtures of compounds.

7.1.18 ADTAPE/ADCON (DEC-12-UW2A-D)

ADTAPE is a data acquisition program that allows the user to: sample simultaneously from 1 to 16 A/D channels at sampling rates ranging from 1000 points per second to 40 seconds per point; display the output of any two channels on the CRT display; and output all results to LINCtape in real time. ADTAPE has a setup mode that allows the user to define a wide variety of sampling schemes using either the keyboard, CRT display or LINCtape. The program ADCON is utilized upon completion of ADTAPE and allows the user to sort the ADTAPE LINCtape output for a given channel onto contiguous tape blocks for further processing.

7.1.19 TISA (DEC-12-UW3A-D)

TISA can acquire asynchronous (time-independent) or synchronous data simultaneously from up to five instruments at rates that do not exceed 2 milliseconds per point. TISA stores data on LINCtape and supports up to 32K of core. Data is displayed on the CRT display via a moving window and cursor with X-Y decimal readout. TISA has a setup mode that allows the user to define a wide variety of parameters using either the keyboard/CRT display of LINCtape. TISA is capable of acquiring data from instruments that are interfaced via shaft encoders or potentiometers, or both. With the ability to call any LAP6-DIAL program, TISA is able to interact with all PDP-12 software.

7.1.20 FOCAL 12

FOCAL-12 is an adaptation of the conversational FOCAL(T) (FOrmula CALculator) language designed specifically to optimize use of the PDP-12 computer and its standard peripheral devices. The LINCtape, console switches, external sense lines, and VR12 display can all be utilized effectively through FOCAL-12. Simple data acquisition and reduction tasks may be quickly programmed using FOCAL-12, and the language can be used to analyze previously generated data stored on LINCtape. FOCAL-12 requires 8K of memory.

7.2 PDP-8 PROGRAMS

The PDP-8 programs available to the user are listed in this section. The programs described in these abstracts come from two sources, past programming efforts on the PDP-5, 8, 8/S, 8/I, 8/L, and present and continuing programming effort on these 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 already in the field.

Although all utility and functional program documentation is issued in a newer format than that introduced with the Family-of-8 computers, in many cases PDP-12 programs originated from previous 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 PDP-12 users. Consequently, users of all PDP-12 computers can take full advantage of continuing program developments.

7.2.1 System Programs

DEC-08-AJAD-D FOCAL – FOCAL (for FOrmula CALculator) 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 in standard notation for the most part. No previous programming experience is needed either to understand the 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 Programming Languages Handbook.

DEC-D8-SDAB-D Disk Monitor System (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, including a FORTRAN Compiler, Program Assembly Language (PAL-D), Edit program (EDITOR), Peripheral Interchange Program (PIP), and Dynamic Debugging Technique (DDT-0) 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, 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 disk (referred to as the system device) for storage and retrieval of his programs. By typing appropriate commands into 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 disk by the Assembler, and assign it core locations), 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. See Introduction To Programming for further details.

DEC-08-AFC0-D, 4K FORTRAN

The 4K FORTRAN Compiler lets the user express problems in a mixture of English words and mathematical statements. It reduces the time needed for program preparation and enables users with little or no knowledge of the computer's organization and operating language to write effective programs.

The 4K FORTRAN language consists of four general types of statements; arithmetic, logic, control, and input/output. FORTRAN functions include addition, subtraction, multiplication, division, sine, cosine, arctangent, square root, natural logarithm, and exponentiation.

DEC-08-KFXB-D, 8K FORTRAN

The 8K FORTRAN system translates a source program into relocatable binary code. The relocatable binary code is output on paper tape and then loaded into the computer for program execution. The 8K FORTRAN system features: USA Standard FORTRAN syntax; subroutines; two levels of subscripting; function subprograms; input/output supervisors; relocatable output loaded by the 8K Linking Loader; COMMON statements; I, F, E, A, X, and H format specifications; and arithmetic and trigonometric library subroutines.

The 8K FORTRAN system consists of a one-pass compiler, the 8K SABR Assembler, 8K Linking Loader, and a comprehensive Library of subprograms. The system requires a PDP-8 Family Computer with at least 8K words of core memory, an ASR33 Teletype, and a high-speed paper tape reader and punch. 8K FORTRAN utilizes all available core to 32K. 8K FORTRAN is a modified version of USASI Basic FORTRAN and is further described in Programming Languages Handbook.

DEC-08-CDDB-D, DDT

The Dynamic Debugging Technique 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. This and it's octal counterpart ODT-8 (DEC-08-COCO-D) are described in Introduction To Programming.

DEC-08-YQYB-D, Floating-Point System

A Basic System (DEC-08-YQ1B-PB)
B Interpreter, I/O, I/O Controller (-YQ2B-)
C Interpreter, I/O Functions (-YQ3B-)
D Interpreter, I/O, I/O Controller, Functions (-YQ4B-)

As described in Programming Languages Handbook this 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 six digits is maintained.

DEC-08-AFA2-PB, FORTRAN SYMBOL PRINT

Loaded after 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.

7.2.2 Elementary Function Routines

The following routines are described in the Program Library Math Routines Manual (DEC-08-FFAC-D) and in the Programming Languages Handbook.

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 $-4<X<4$

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 fraction $-1 < \mathrm{Sin}(X) < 1$.

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.

Arithmetic Shift Subroutines

Four basic subroutines, arithmetic shift right and arithmetic shift left at both single and double precision, are presented as a package.

Logical Shift Subroutines

Two basic subroutines, logical shift right at both single and double precision are presented as a package.

Digital-8-21-F Signed 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 (DEC-08-YQ1B through YQ4B), except that certain routines have been speeded up by the use of the Extended Arithmetic Element.

For a detailed description of floating-point arithmetic and the interpretive Floating-Point Packages, the reader is referred to DEC-08-YQYB-D or the Programming Languages Handbook.

7.2.3 Utility Program

DEC-08-LRAA-D, 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 33 ASR Perforated Tape Reader (also high speed reader version).

DEC-08-LBAA-D, Binary Loader (33 ASR, PR12, MC12 Memory Extension)

The Binary Loader is a routine for reading and storing the information in binary-coded tapes via the 33 ASR Perforated Tape Reader or by means of the Type PR12 High-Speed Perforated Tape Reader.

DEC-08-PMP0-D, RIM Punch

This program provides a means of punching out the information in selected blocks of core memory as (Read-In Mode) RIM-coded tape via the 33 ASR Perforated Tape Reader or the High Speed Punch PP12.

DEC-08-YX1A, Binary Punch (33 ASR)
DEC-08-YX2A, Binary Punch (PP12)

This program provides a means of punching out the information in selected blocks of core memory as binary-coded tape via the 33 ASR Perforated Tape Punch or the High Speed Punch PP12.

DEC-08-YPPA, 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; it repeats this until the block is exhausted, at which time the user may repeat the operation.

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.

Digital-8-20-U-SYM Character String Typeout Subroutine

This basic subroutine types messages stored internally as a string of coded characters. All 33 ASR characters are legal.

Digital-8-21-U-SYM Symbolic Tape Formal 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, Single Precision

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 two's complement number. If bit 0 of the computer word is a 1, the remaining bits represent a negative integer in two'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-FFAC-D or Programming Languages Handbook). 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, two's complement number. If bit 0 of the high order word is a 1, the remaining 23 bits represent a negative integer in two'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 Decimal-to-Binary Conversion & 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 two'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.

Digital-8-29-U-SYM Double Precision Decimal-to-Binary Conversion & 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 two'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.

7.3 DECUS PROGRAM

DECUS is the Digital Equipment Corporation Users' 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. 5-5, 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 with 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. 5/8-9, 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

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 DEC-08-YQYB-D described in Programming Languages Handbook.

DECUS NO. 5/8-18a, BINARY TAPE DISASSEMBLY PROGRAM

Disassembles a PDP-8 program, which is on tape in BIN format. It prints the margin setting, address, octal contents, and mnemonic interpretation (PAL) of the octal contents. A normal program or a program which uses Floating Point may be disassembled.

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

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. 5-25, 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. 8-26c, XCBL – EXTENDED MEMORY CBL LOADER

XCBL is used to load binary tapes punched in CBL format into any 4K memory field. 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 extended memory through the control of the keyboard of the on-line Teletype.

DECUS NO. 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 starts 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.

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, FORPLOT – FORTRAN PLOTTING PROGRAM

FORPLOT is a general-purpose plotting program for use in conjunction with the CALCOMP 560 Plotter. It is self-contained and occupies memory locations 0000 to 4177. FORPLOT accepts decimal data inputted on paper tape in either fixed or floating point formats. Formats can be mixed at will. FORTRAN output tapes are acceptable directly, and any comments 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 program around in memory to make the most efficient use of computer storage.

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. 5/8-35, 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. 5/3-38, 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 μ 2000). 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. DDTYPE 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. 5/8-43, 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 with the 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 (DEC-08-YQYA-D)

The Floating Point Package (DEC-08-YQYA-D) 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.

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. 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 center-of-mass angles, the program calculates angles and energies of the emitted particles in the Lab frame. If the process if 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 tables similar 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, with 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} registers.

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. 5/8-56, FIXED POINT TRACE NO. 1

A minimum size monitor program which executes the user's 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 of 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 1/4 second 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. 8-60, SQUARE ROOT FUNCTION BY SUBTRACTION REDUCTION

A single precision square routine using EAE. This routine is usually 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 by 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. 8-65, A PROGRAMMED ASSOCIATIVE MULTICHANNEL ANALYZER

The program describes the use of a small computer as an associate analyzer 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 a LABEL for PDP-8, 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 computer 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 of 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 parabola least squares fit to approximate a given data curve. Approximately 400 individual parabolas are computed.

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%.

Other Programs Needed: FORTRAN Operating System (DEC-08-CFA3-PB), dated March 2, 1967.

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 sweepout, 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 EQUATION: 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 Complier 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. 8-76, PDP NAVIG 2/2

This program utilizes the output of the U.S. Navy's AN/SRN-9 satellite navigation receiver to obtain fixes. 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 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 when using 7 decimal digits per word.

DECUS NO. 8-77, PDP-8 DUAL PROCESS SYSTEM

The purpose of this system is to expedite the programming of multiprocessing problems on the PDP-8. 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 about 0.1% overhead 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, become I/O Bound.

The system requires 600_8 registers for two TTYs plus buffer space. Several device configurations are possible.

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 avoid 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 routines in requiring more memory space than his first one (DECUS No. 8-56) (five pages as opposed to two) and in not possessing the trace-suppression features of his second one (DECUS No. 8-57). 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. 8-79, TAC-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 an eigenvector 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 reloaded, and the next eigenvalue in line may be calculated. In this manner, all the real eigenvalues may be computed in order.

DECUS NO. 5/8-83A 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 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. 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. 5/8-126, CUMULATIVE GAUSSIAN DISTRIBUTION CURVE FITTING

This is a curve fitting program that will take a set of any number of points with any spacing describing a cumulative Gaussian distribution and determine the mean and standard deviation by an iterative least squares differential-correction technique. The mean square error of the final fitted curve is also computed. The program is coded in PDP-8 FORTRAN.

DECUS NO. 8-133, FIRST ORDER KINETICS

First order kinetic processes are common in chemistry and in other areas. This program accepts up to 42 data points, calculates the rate constant and intercept by the method of least squares, and gives the rms deviation, the correlation coeffecient, and an estimate of the error in slope. It permits graphical (CRT) examination of deviations from the least squares line and iteration to a "best" infinity value. It also provides options for plotting the deviation between observed and calculated quantities on a CRT, and may be used in other cases in which one wishes to correlate the natural logarithm of one quantity with another, as in linear free energy relationships.

DECUS NO. 8-134, LSQ: LEAST SQUARES SUBROUTINE

The subroutine calculates the slope and intercept for the equation $y_j = mx_j + b$ by the method of least squares. It also returns the ms deviation of y, the correlation coefficent, and an estimate of the error in the slope. The calculated values of y and the differences between the given and calculated values are also available on return from the subroutine.

Other Programs Needed: FLOAT, floating point interpreter "C" – (DEC-08-YQ3B-PB)

DECUS NO. 8-136, FOURIER TRANSFORM PROGRAM IN FORTRAN II

The program, written in PDP-8 FORTRAN II, performs the discrete Fourier Transform of a function defined over $N(N<200)$ evenly spaced points. I/O is via the ASR-33. The program requests the number of function points, then that number of function values, and then prints out the values of the sine and cosine components of the function at each defined harmonic. A conventional (not Cooley-Tukey) algorithm is used, since I/O time relative to computing time is significant.

DECUS NO. 8-143, FFTS-R: FAST FOURIER TRANSFORM SUBROUTINE FOR REAL VALUED FUNCTIONS

This subroutine computes the Fast Fourier Transform (FFT) or its inverse of a data sequence which has been stored in core. It will accommodate up to 2048 time samples, and will transform that number in under 5 seconds.

Minimum Hardware: PDP-8, PDP-8/I, or PDP-12 with EAE

DECUS NO. 8-144, FFTS-C: FAST FOURIER TRANSFORM SUBROUTINE FOR COMPLEX DATA

FFTS-C enables computation of the Discrete Fourier Transformation in a minimum amount of time. By using the Cooley-Tukey algorithm, up to 1024 points may be transformed in only 4.5 seconds, introducing a reduction of 99 percent in computation time.

Minimum Hardware: PDP-8, PDP-8/I, or PDP-12

DECUS NO. 8-192, TALC (TAYLOR'S ALGEBRAIC LINEAR CALCULATOR)

TALC is a general-purpose calculator designed to evaluate a general algebraic equation, given all quantities involved in the equation. In effect, TALC turns any of the family-of-eight computers into a powerful desk calculator capable of evaluating complex algebraic, trigonometric, and logarthmic functions. In addition, TALC utilizes the concept of "idiot-proofing" to virtually eliminate the possibility of an operator error invalidating the equation. Thus, TALC is easy to use and presents unlimited possibilities in any field where fast and accurate calculations are required.

Minimum Hardware: 4K PDP-8 or PDP-12 High-Speed Reader, DF32 Disk File, and ASR-33/35

DECUS NO. 8-195, POLY BASIC

POLY BASIC is a compiler and operating stand-alone system designed for the PDP-8 family. It has a total user program storage of 32K characters in which the disk is utilized. Some of the features of the compiler are:

a. It has all BASIC system commands.

b. It has all BASIC operations.

c. It contains all built-in functions except TAN.

d. Its accuracy is 1 part in 2^{23} rather than 1 part in 2^{35}, because of word size difference.

e. Maximum program size is 6144 characters as in regular (Dartmouth) BASIC.

f. Maximum usable statement number 4095 rather than 99999.

g. Maximum array space is 3600 characters, and maximum number of statements is 330; however, these can be traded off against one another at the rate of 25 array elements per statement.

h. There are no matrix operations.

i. The argument "EDIT resequence" is implemented, and the command "EDID" renumbers the user file from line number 100 in steps of 10.

j. There is a set of error messages to signal compilation errors and a set for execution errors.

Minimum Hardware: PDP-8 or PDP-12 with DECtape, DF32 Disk, RF08 Disk, or LINCtape

DECUS NO. 8-202, PLOT

PLOT will plot data points on a graph, calculate and plot a linear, least squares regression line, and print out the coefficient of correlation, the equation of the regression line, and other pertinent parameters.

Minimum Hardware: 4K PDP-8 or PDP-12 with a Houston Instrument Complot Plotter Model 6650, DP-1-1, or equivalent

DECUS NO. 12-2, PDP-12 UTILITY AND DATA REDUCTION PROGRAMS

Contains a variety of programs written for the classic LINC or LINC-8 and modified to run in the PDP-12. Included are data reduction programs which perform autocorrelation, fourier analysis, power spectral analysis, and convolution. Utility programs allow selected blocks of LINCtape to be searched, compared, or typed out. Also included are programs which allow the user to convert LAP-4 or LAP-6 manuscripts into LAP6-DIAL or to disassemble binary code into LAP6 or LAP6-DIAL source. The current version of this tape contains binary and/or source for the programs listed below. Also listed are the original authors. Documents, but no listings, are available from DECUS.

FREQAN	SIMONS
AUTOCOR	HANCE
QAFILTER	GLAESER
FFOURIER	BRYAN
FFSAMPLE	BRYAN
DATAM	HANCE
FRQANA	ENGEBRETSON
TAPEDUMP	LANAHAN
SEARCH	STEIN
COMPAR	DAVISSON
COMPARE	NICHOLS
LAP4-6	BJERKE
BINLAP6	BJERKE
L8SIM0	LANGBEIN
LAP4Q+A	McDONALD
DSCTABLE	OVERTON
PARAMS	NICHOLS
DIVSUB	STEIN
SPFLT	DILL
QIKDIV	HANCE
RANDOM	LEWIS
NU INDEX	CLAYTON

DECUS NO. 12-3, 8K OPERATING SYSTEM

This modification of DECUS NO. L-80 adapts it to run on the PDP-12.

DECUS NO. 8-203, ALPHA

ALPHA is used for titling graphs on the plotter. It can be used in conjunction with DECUS NO. 8-202.

Minimum Hardware: 4K PDP-8 with a Houston Instrument Complot Plotter Model 6650, DP-1-1, or equivalent

Storage Requirement: 2, 10, 11, 12, 20-167, 200-3654, 4000-777 is reserved for storage.

Restriction: When used in conjunction with PLOT (DECUS No. 8-202), extended memory is required.

DECUS NO. FOCAL-14, LEAST SQUARES FIT TO A STRAIGHT LINE

This is a program using the principle of least squares to fit a straight line to a set of up to 35 experimental data points.

The program requires one pass of the data. At the end of the pass the output gives the values of the slope and the intercept of the straight line equation. In addition, the calculated values of the experimental data based on the straight line equation are pointed out. Finally, the program gives the value of R which is a criterion of fitness of equation to the input data.

DECUS NO. FOCAL-15, LEAST SQUARES FIT TO A CUBIC POLYNOMIAL

This is a program using the method of least squares to fit a cubic polynomial to a set of experimental data. The program demands two passes of the data for its completion; however, the coefficients of the polynomial are outputted after the first pass, and, at the end of the second pass, the output gives the value of R which is a criterion of fitness and gives the calculated values of the experimental data based on the cubic polynomial.

In addition, a section of the program can be used as a self-contained program for solution of a set of N by N linear equations.

DECUS NO. FOCAL-16, ONE-SAMPLE STATISTICS, TWO-SAMPLE STATISTICS, WELCH PROCEDURE; ONE-WAY ANALYSIS OF VARIANCE; SHEFFE'S CONTRAST BETWEEN MEANS

A three part program used to perform one-sample and two-sample statistics, Welch Procedure; One-way Analysis of Variance: and Sheffe's Contrast Between Means, which allows one to investigate more thoroughly the source of the difference between group means.

DECUS NO. FOCAL-17, FOCAL: HOW TO WRITE NEW SUBROUTINES AND USE INTERNAL ROUTINES

The aim of this document is to help the user to code specific routines in FOCAL so that his dialect of FOCAL can be applied to his application (without being forced to understand in detail all the working of FOCAL). In this way, perhaps, each user can make his particular dialect of FOCAL "perfect".

It is assumed that the reader has a basic knowledge of PDP-8 processor instructions and PAL mnemonics, as well as a familiarity with the Floating Point Package. In addition, he should be familiar with the FOCAL language.

This document is an attempt to explain how user-developed software can be interfaced with the basic FOCAL package, without requiring the user to spend valuable time trying to understand all of its detailed workings. Section II deals with a general description of how FOCAL works. Section III is concerned with the philosophy of the language; and the last few sections are technically oriented toward helping the user actually code his additions. Several examples and ready-coded routines which may be used to simplify the user's problems are included.

DECUS NO. FOCAL-25, PAYROLL CALCULATIONS (CALIFORNIA, 1968)

This routine is used to calculate payrolls. It is based on the California State Unemployment Insurance rate, FICA rate, and withholding tax.

This program could be modified easily to fit the rules of any particular state. If some of the pay ranges would not be used, they could be omitted from the two tables, making more room for other routines, such as providing running totals on gross pay, deductions, and net pay.

DECUS NO. FOCAL-26, CURVE FITTING

This program finds the best curve of a set of points. There are three types of curves involved:

Exponential Curve $Y = Ae^{BX}$

The variables solved for are A and B. The function is reduced to linear form by logarithms: LOG Y = BX + LOG A. A table of values is formed and solved simultaneously to get the values of A and B.

Power Curve $Y = AX^N$

This function is reduced to linear form: LOG Y = N LOG X + LOG A. Once a table of values is made, it is solved simultaneously for A and N.

Linear Line Y = MX + B

A table is made and solved simultaneously for the value of M and B.

DECUS NO. FOCAL-33, SQUARE MATRIX MULTIPLY

The arduous task of multiplying two square matrices is quickly done by this FOCAL Matrix Multiplication routine. The user inputs "N", indicating the number of rows and columns each matrix will have. The computer then requests input of the elements of the two matrices. The result of the multiplication is typed out in an understandable matrix-like format.

Notable characteristics of this program are:

a. It is expressed in only five lines of FOCAL script so that it loads quickly.

b. It will process matrices of varying dimensions. Size of each matrix is limited only by memory capacity. (In 4K FOCAL the limit is about 6 rows and columns.)

c. Because it inputs and outputs the matrix values in a matrix-like format, input transcription errors are less likely to occur.

DECUS NO. FOCAL-37, Nth DEGREE POLYNOMIAL DATA POINT FITTING ROUTINE, Nth DEGREE POLYNOMIAL DATA POINT FITTING ROUTINE WITH RMS ERROR

a. Nth Degree Polynomial Data Point Fitting Routine – This program accepts the x- and y-coordinates for an unlimited number of data points and calculates for the equation

$$Y = A_0 + A_1X + A_2X^2 + \ldots + A_NX^N,$$

the coefficients A_N which best fit the equation to the data points. The fitting criterion is "least squares". The program allows the user to select the degree, N, of the fitting equation. N may be as large as 7.

b. Nth Degree Polynomial Data Point Fitting Routine with RMS Error – This program is the same as Nth Degree Polynomial Fitting Routine except that it calculates the RMS error between the y-coordinates of the data points and the evaluated fitting equation. It will accept only a limited number of data points and the maximum equation degree allowed is inversely related to this number.

DECUS NO. FOCAL-40, SIMPLE CHI-SQUARE TEST

The program will type out the data matrix and cell contents. Each cell will contain two values, 0 = xxx.xxx and E = xxx.xxx. The "O=" number is the "OBSERVED" value which was typed in by the user. The "E=" value is the expected value calculated by the program. The program will also type out row sums (RS=) and column sums (CS=), and the grand total (T=). The last line of output will be "X2= " and "DF= ". These are the CHI-SQUARE and degrees of freedom.

DECUS NO. L-11, DATUM8

DATUM8 is a revision of an addition to DATAM by James Hance contained in the general library supplied with the LINC-8 computer. This program has retained all the features of DATAM. Some of the original routines have been changed in order to eliminate undesired features. In addition, DATUM8 has the ability to multiply, subtract, and display the data with two cursors. The data not included between the cursors can be suppressed, allowing, for instance, integration between definite limits. The program has been recoded to facilitate future modifications.

Minimum Hardware: LINC (2K) or LINC-8

DECUS NO. L-60, FORTRAN WITH LINCTAPE

The system adapts 4K PDP-8 FORTRAN to make use of the magnetic tape facility of the basic LINC-8 computer. FORTRAN programs can be called from the left-hand tape (unit 0) by statements in the FORTRAN program in the core; data can be stored and recalled by WRITE and READ statements, using the right-hand tape (unit 1).

The programs may be called in any order, and may be read in at less than full length in order to leave data in core. Data can be transferred to and from tape in blocks of any length which may be chosen to correspond with program variables. Incorporation of the program-calling facility does not involve any sacrifice of core space for program or data; use of the data transfer facilities uses from half to one page of core.

The system includes a modified, improved version of SYMBOL PRINT, increases the maximum length of the statement number tables, and corrects some errors in the compiler.

Other Programs Needed: PDP-8 FORTRAN System (DEC-08-AFCO)

Source Language: PAL III

DECUS NO. L-80 AND 12-3, 8K OPERATING SYSTEM

The 8K Operating System was developed at the University of Michigan (Cooley Electronics Lab). It is a two-tape-resident operating system for the LINC-8 with a special 33ASR reader modification. It allows a user to enter and edit programs from the Teletype keyboard, store them in files on magnetic tape, assemble or compile these files into PDP-8 machine language, and run the resulting program. These operations are done entirely from the Teletype keyboard without the use of paper tapes or manipulation of the switches. One of the language translators is the DEC 8-K FORTRAN system (February 1969 version).

The 8K Operating System consists of 3 system components: a filing system, a system of loaders for running programs, and a command language interpreter. A file is a collection of information (which the user generates) in machine-readable form; for example, it may be the set of characters comprising a FORTRAN program. Using the filing system, the user may name this set of characters and store it on magnetic tape. To use the file containing the program (for example, in a FORTRAN compilation) the user need only refer to the file name and the system will handle all bookkeeping necessary to communicate the file to the FORTRAN compiler. A user may change the contents of files, destroy files, and copy files from other users' tapes. The assembler processes both LINC mode and PDP-8 mode statements, assembling programs up to 8K in length. Six character alphanumeric symbols can be defined; source listings, assembly listings, or abbreviated assembly listings may be (optionally) obtained on the Teletype or, if present, the Line Printer. To facilitate the preparation of system programs and/or large programs, conditional assembly pseudo ops are provided, along with a facility to preserve the tag table definitions from one assembly to another, or to several others.

Upon loading, programs may be started automatically in either LINC or PDP-8 mode at any memory location permanent file by means of a "SAVE" command.

The loading system contains two separate loaders, the absolute loader and the relocating loader. These loaders do not load programs directly into core memory, but rather into a core memory image kept on magnetic tape. (This intermediate step is necessary because of core space limitations and the flexibility desired of the loaders.) Both loaders take their binary (and core image) input from the system files; both include provisions for changing and viewing the contents of various core image locations.

The absolute loader operates on the binary output of either the MACRO-8 of PAL-III assemblers to produce a core image. It is capable of merging an extant core image into the core image it is building. It will also page-relocate binary input information to be within a different memory page from that for which it was assembled.

The relocating loader operates on FORTRAN compiler and SABR assembler output. It is the final part of the 3-part FORTRAN/SABR/Relocating Loader package, which allows the programmer to write programs for the PDP-8 disregarding the memory addressing structure of the machine. After the loading process is complete, the core image file containing the program is available to the user in -B, and may be saved or run immediately using the "RUN" command.

The "RUN" command software is also part of the loading system. It allows execution of a program contained in a core image file.

7.4 DIAGNOSTIC PROGRAMS

PDP-12 PROCESSOR TESTS
Maindec-12-D0AB Tests SKIPs and data handling (LINC mode).
These diagnostics test the LINC mode processor and memory control instructions.

PDP-12 TAPE CONTROL TESTS (TC12)
Maindec-12-D3AC Tests tape control logic and interregister transfers.
These diagnostics make 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 (VR12)
Maindec-12-D6BA Tests display system using DIS and DSC instructions.
The diagnostic tests both the VC12 CRT display control and the VR12 CRT display oscilloscope.

PDP-12 REAL-TIME INTERFACE TEST (KW12)
Maindec-12-D8CB
This diagnostic thoroughly tests the KW12-A Real-Time Interface.

PDP-12 A-D TEST (AD12)
Maindec-12-D6CB-Tests A-D converter and calibration.
This test verifies the logic of the AD12 and allows checking and alignment of all 32 channels of the A-D converter.

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-D04B-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-D05B-D Random JMP-JMS Test

This program tests 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 JMS.

Maindec-08-D07B-D Random ISZ Test

This program tests 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 fixed set of FROM, TO, and OPERAND numbers.

Maindec-08-D0AA-D Instruction Test (EAE) Part 3A

This program tests 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 divide and multiply are used to test the results of the hardware divide and multiply.

Maindec-08-D1AC-D Memory Power On/Off Test

This program is a Memory Data Validity Test to be used after a simulated power failure.

Maindec-08-D1L1-D Memory Checkerboard Test Low
Maindec-08-D1L2-D Memory Checkerboard Test High

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-D1L1-D, Maindec-08-D1L2-D and Maindec-08-D1DA-D.

Maindec-08-D1DA-D Extended Memory Checkerboard Part 2

This preliminary program tests 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-D2PE-D Teletype Reader Test

This program tests performance of the Teletype Model 33 Perforated Tape Reader, using the reader to scan a closed-top test page 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 exerciser program tests for the Teletype Paper-Tape Reader. Test tapes are read in a random start-stop fashion and errors are reported on the Teletype printer.

Maindec-08-D2CA-D Teletype Punch Test

This program 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

This program tests the Teleprinter 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-D2GF-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 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 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 separate tests. The first causes the PP12 and PP8/I High-Speed Paper-Tape Punch to produce a tape containing a sequence of pseudo-random character codes. This tape is checked for accuracy, using either the high-speed reader or 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-D6CB-D Calcomp Plotter Test

This program tests the CALCOMP or HOUSTON Plotters and their control. All control and plotting functions are tested.

PDP-12 JMP SELF MAINDEC-12-D1BA-D

This program tests the ability of the read/write gates to switch rapidly between read and write.

PDP-12 Instruction Test 1 MAINDEC-12-D0BA

This program tests for the basic overall test of most of the LINC mode instructions.

PDP-12 Address Test (R/W Gate) MAINDEC-12-D1CA

This program tests the ability of the R/W gates to rapidly change addresses. It generates two random addresses, loads these addresses with jumps to each other, and allows the jump sequence to continue until interrupted.

PDP-12 Memory Data Test (Float 1s and 0s) MAINDEC-12-D1EB

This program floats a single one then a single zero through all of memory to evaluate data handling.

PDP-12 Memory Checkerboard MAINDEC-12-D1DA

Tests memory for core failure on half-select lines under the worst possible conditions for reading and writing. It is used primarily for testing the operation of memory.

APPENDIX A
LINC MODE INSTRUCTIONS

The following table lists all LINC mode instructions and their mnemonics, octal codes, and execution times. Full instruction descriptions are in Chapter 3.

Table A-1. LINC Mode Instructions

Mnemonic	Function	Octal	Time (μs)
	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 AC)	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	0240	1.6-6.4
ROR N	Rotate right N places	0300	1.6-6.4
SCR N	Scale right N places	0340	1.6-6.4

* See Note at end of table.

Table A-1. LINC Mode Instructions (cont)

Mnemonic	Function	Octal	Time (μs)
	OPERATE		
HLT	Halt	0000	1.6
NOP	No operation	0016	1.6
CLR	Clear AC and LINC	0011	1.6
SET	Set register N to contents or 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 Function to AC	0024	1.6
QAC	MQ transfer to AC	0005	1.6
	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	0456	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	1.6-18.2
SAM N	Sample analog chan N	0100+N	1.6**
DIS	Display point on oscilloscope	0140	3.2-23**

**See Note at end of table.

Table A-1. LINC Mode Instructions (cont)

Mnemonic	Function	Octal	Time (μs)
DSC	Display character on oscilloscope (2 x 6 matrix)	1740	4.8-56**
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
LINC	8 Mode to LINC Mode	6141	4.25
	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 idle	0416	1.6
TWC	Skip if tape word complete	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 (Chapter 3 LINC Mode Instructions.)

APPENDIX B
8-MODE INSTRUCTIONS

Table B-1. 8-Mode Memory Reference Instructions

Mnemonic Symbol	Operation Code	Direct Addr.		Indirect Addr.		Operation
		States Entered	Execution Time (μs)	States Entered	Execution Time (μs)	
AND Y	0000	F,E	3.2	F,D,E	4.8	Logical AND. The AND operation is performed between the contents of memory location Y and the contents of the AC. The result is left in the AC, the original contents of the AC are lost, and the contents of Y are restored. Corresponding bits of the AC and Y are operated upon independently. $AC_j \wedge Y_j \rightarrow AC_j$
TAD Y	1000	F,E	3.2	F,D,E	4.8	Two's complement add. The contents of memory location Y are added to the contents of the AC in two's complement arithmetic. The result of this addition is held in the AC, the original contents of the AC are lost, and the contents of Y are restored. If there is a carry from AC_0, the link is complemented. $AC + Y \rightarrow AC$

Table B-1. 8-Mode Memory Reference Instructions (cont)

Mnemonic Symbol	Operation Code	Direct Addr. States Entered	Direct Addr. Execution Time (μs)	Indirect Addr. States Entered	Indirect Addr. Execution Time (μs)	Operation
ISZ Y	2000	F,E	3.2	F,D,E	4.8	Increment and skip if zero. The contents of memory location Y are incremented by one. If the resultant contents of Y equals zero, the contents of the PC are incremented and the next instruction is skipped. If the resultant contents of Y do not equal zero, the program proceeds to the next instruction. The incremented contents of Y are restored to memory. If resultant Y = 0, PC + 1 → PC.
DCA	3000	F,E	3.2	F,D,E	4.8	Deposit and clear AC. The contents of the AC are deposited in core memory at address Y and the AC is cleared. The previous contents of memory location Y are lost. AC → Y 0 → AC
JMS Y	4000	F,E	3.2	F,D,E	4.8	Jump to subroutine. The contents of the PC are 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 contents of the PC are lost. Y → PC

Table B-2. 8 Mode Group 1 Operate Microinstructions

Mnemonic Symbol	Octal Code	Sequence	Operation
NOP	7000	–	No operation. Causes a 1.6 μs program delay.
IAC	7001	3	Increment AC. The contents of the AC are incremented by one in two's complement arithmetic.
RAL	7004	4	Rotate AC and L left. The contents 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 contents 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 contents of the AC are set to the one's complement of its current contents.
CIA	7041	2,3	Complement and increment accumulator. Used to form two's complement number.
CLL	7100	1	Clear L. The L is cleared to contain a binary zero.
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 one.
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 CLA CMA	7240	2	Set AC = -1. Each bit of the AC is set to contain a binary one.

Table B-3. 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	3	OR with Right Switch register. The OR function is performed between the contents of the RSW and the contents 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.

Table B-3. 8 Mode Group 2 Operate Microinstructions (cont)

Mnemonic Symbol	Octal Code	Sequence	Operation
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 AC contains a negative number, the next instruction is skipped.
SPA	7510	1	Skip on positive AC. If the AC contains 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 $\leqslant$ 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	2,3	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 > 0, then clear AC.

Table B-4. 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 instruction 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 $\rightarrow$ 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 instruction). At the conclusion of this instruction 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 $\div$ Y $\rightarrow$ MQ.
NMI	7411	3	Normalize. This instruction is used as part of the conversion of a binary number to a fraction and its exponent for use in floating-point arithmetic. The combined contents of the AC and the MQ are shifted left by this one instruction until the content of AC_0 is not equal to the content of AC_1, thus forming the fraction. Zeros are shifted into vacated MQ_{11} positions for each shift.

Table B-4. 8 Mode Extended Arithmetic Element Microinstructions (cont)

Mnemonic Symbol	Octal Code	Sequence	Operation
			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. $AC_j \rightarrow AC_{j-1}$ $AC_0 \rightarrow L$ $MQ_0 \rightarrow AC_{11}$ $MQ_j \rightarrow MQ_{j-1}$ $0 \rightarrow MQ_{11}$ until $AC_0 \neq AC_1$
SHL	7413	3	Shift left. This instruction shifts the combined contents of the AC and MQ to the left one position more than the number of positions indicated by the contents of the core memory at address PC + 1 (or the next successive core memory location following the SHL instruction). During the shifting, zeros are shifted into vacated MQ_{11} positions. Shift Y + 1 positions as follows: $AC_j \rightarrow AC_{j-1}$ $AC_0 \rightarrow L$ $MQ_0 \rightarrow AC_{11}$ $MQ_j \rightarrow MQ_{j-1}$ $0 \rightarrow MQ_{11}$
ASR	7415	3	Arithmetic shift right. The combined contents of the AC and the MQ are shifted right one position more than the number contained in memory location PC + 1 (or the next successive core memory location following the ASR instruction). The sign bit, contained in AC_0, enters vacated positions, the sign bit is preserved, information shifted out of MQ_{11} is lost, and the L is undisturbed during this operation. Shift Y + 1 positions as follows: $AC_0 \rightarrow AC_0$ $AC_j \rightarrow AC_{j+1}$ $AC_{11} \rightarrow MQ_0$ $MQ_j \rightarrow MQ_{j+1}$
LSR	7417	3	Logical shift right. The combined contents of the AC and MQ are shifted right one position more than the number contained in memory location PC + 1 (or the next successive core memory location following the LSR instruction). This instruction is similar to the ASR instruction except that zeros enter vacated positions instead of the sign bit. Information shifted out of MQ_{11} is lost and the L is undisturbed during this operation. Shift Y + 1 positions as follows: $0 \rightarrow AC_0$ $AC_j \rightarrow AC_{j+1}$ $AC_{11} \rightarrow MQ_0$ $MQ_j \rightarrow MQ_{j+1}$

Table B-4. 8 Mode Extended Arithmetic Element Microinstructions (cont)

Mnemonic Symbol	Octal Code	Sequence	Operation
MQL	7421	2	Load multiplier quotient. This instruction clears the MQ, loads the contents of the AC into the MQ, then clears the AC. $0 \rightarrow MQ$ $AC \rightarrow MQ$ $0 \rightarrow AC$
SCA	7441	2	Step counter load into accumulator. The contents of the step counter are transferred into the AC. The AC should be cleared prior to issuing this instruction or the CLA instruction may be combined with the SCA to clear the AC, then effect the transfer. $SC \vee AC \rightarrow AC$
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. Two's $MB_{7-11} \rightarrow SC$ $PC + 2 \rightarrow PC$
MQA	7501	2	Multiplier quotient load into accumulator. The contents of the MQ are transferred into the AC. This instruction 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 instruction or the CLA instruction can be combined with the MQA to clear the AC then effect the transfer. $MQ \vee AC \rightarrow AC$
CLA	7601	1	Clear accumulator. The AC is cleared during sequence 1, allowing this instruction to be combined with other EAE instructions that load the AC during sequence 2 (such as SCA and MQA). $0 \rightarrow AC$
CAM	7621	1,2	Clear accumulator and multiplier quotient. CAM = CLA MQL

APPENDIX C
I/O BUS INSTRUCTIONS

In the 8 mode these instructions are given directly, whereas in the LINC mode they must be 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 Type PR12		
RSF	6011	Skip if reader flag is a 1.
RRB	6012	Read the contents of the reader buffer and clear the reader flag. (This instruction does not clear the AC). RB V $AC_{4-11} \rightarrow AC_{4-11}$
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 Type PP12		
PSF	6021	Skip if punch flag is a 1.
PCF	6022	Clear punch flag and punch buffer.
PPC	6024	Load the punch buffer from AC_{4-11} and punch the character. (This instruction does not clear the punch flag or punch buffer). AC_{4-11} V PB $\rightarrow$ PB
PLS	6026	Clear the punch flag, clear the punch buffer, load the punch buffer from AC_{4-11}, punch the character, and set the punch flag when done.

Table C-1. IOT Instructions (Cont)

Mnemonic	Octal	Operation
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 $AC_{4-11} \rightarrow AC_{4-11}$
KRB	6036	Clear AC, clear keyboard flag, and read the contents of the keyboard buffer into AC_{4-11}. TTO $VAC_{4-11} \rightarrow AC_{4-11}$
Teletype Teleprinter/Punch		
TSF	6041	Skip if teleprinter flag is a 1.
TCF	6042	Clear teleprinter flag.
TPC	6044	Load the TTO from AC_{4-11} and print and/or punch the character.
TLS	6046	Load the TTO from AC_{4-11}, clear the teleprinter flag, and print and/or punch the character.
Power Fail/Restart Type KP12		
SPL	6102	Skip if power is low.
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 instruction.
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 AC_{9-11} respectively.
RMF	6244	Restore memory field. Used to exit from a program interrupt.
Incremental 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 paper.
PLPR	6511	Plotter pen right.
PLDU	6512	Plotter drum (paper) upward.
PLDD	6514	Plotter drum (paper) downward.

Table C-1. IOT Instructions (Cont)

Mnemonic	Octal	Operation
PLPL	6521	Plotter pen left.
PLUD	6522	Plotter drum (paper) upward. (Same as 6512).
PLPD	6524	Plotter pen down. Lower pen to paper.
Random Access Disk File Type DF32		
DCMA	6601	Clears memory address register, parity error and completion flags. This instruction clears the disk memory request flag and interrupt flags.
DMAR	6603	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.
DMAW	6605	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.
DCEA	6611	Clears the disk 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 disk extended address extension registers are cleared and loaded with the track data held in the AC.
DEAC	6616	Clears the AC, then loads the contents of the disk extended address register into the AC to allow program evaluation. Skips 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	Skips next instruction if the completion flag is a 1. Indicates data transfer is complete.
DMAC	6626	Clears the AC, then loads contents of disk memory address register into the AC to allow program evaluation.
Disk Control and Disk File Type RF08/RS08		
DCMA	6601	Clears memory address register, parity error and completion flags. This instruction clears the disk memory request flag and interrupt flags.
DMAR	6603	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 and completion flags. Clears interrupt flags.
DMAW	6605	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.
DCIM	6611	Clears the disk 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.
DFSC	6621	Skip on error condition.
DFSC	6622	Skip next instruction if the completion flag is a 1. Indicates data transfer is complete.
DMAC	6626	Clears the AC then loads contents of disk memory address register into the AC to allow program evaluation.

Table C-1. IOT Instructions (cont)

Mnemonic	Octal	Operation
Disk Cartridge Memory Type RK8/RK01		
DLDC	6732	Loads the command register from the AC and then clears the AC.
DLDR	6733	Loads the track, surface, and sector address from the AC; the instruction then clears the AC and starts to read data from the disk if command register bit 4 is a 0.
DLDW	6735	Loads the track, surface, and sector address from the AC. The instruction then clears the AC and starts to write on the disk if command register bit 4 is a 0.
DCHP	6737	Loads the track, surface, and sector address from the AC. The instruction then clears the AC and reads data and checks parity if command register bit 4 is a 0.
DRDA	6734	Clears the AC and then reads the Track Address Counter and surface/sector counter into the AC.
DRDC	6736	Clears the AC and then reads the command register into the AC.
DRDS	6741	Clears the AC and then reads the status register into the AC.
DCLS	6742	Clears the status register.
DMNT	6743	Load maintenance register. This instruction loads the maintenance register from the AC and carries out the operation specified. The bits will remain set until DMNT is reissued with all AC bits cleared to 0.
DLDA	6731	Loads the disk address. This instruction is a maintenance operation.
DSKD	6745	Skip On Transfer Done flag equal to 0.
DSKE	6747	Skip when the error flag is set to 1.
DCLA	6751	Clear All. This instruction clears the selected disk to Track 000 and then clears all the control registers and status flags except the disk selection. Transfer Done is set when the disk is positioned on Track 000.
DRWC	6752	Read word count register. This instruction clears the AC, then reads the contents of the word count register into the AC.
DLWC	6753	Load word count register. This instruction loads the word count register from the AC and then clears the AC.
DLCA	6755	Load current address register. This instruction loads the current address register from the AC and then clears the AC.
DRCA	6757	Read current address register. This instruction clears the AC and then reads the contents of the current address register into the AC.
Real Time Interface Type KW12A		
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.
Fixed-Interval Clock Type KW12-B, C		
CSOF	6131	Skip on clock flag.
CTOC	6132	Turn off the clock, clears the clock flag and disables the clock interrupt.
CTON	6134	Turn the clock on and clears the flag.

Mnemonic	Octal	Operation
Fixed-Interval Clock Type KW12-B,C (cont)		
CRUN	6135	Clock running. Turns on the clock, enables the clock interrupt, and clears the clock flag. If the clock flag was set when the instruction was issued, it will skip.
Automatic Line Printer and Control Type LP12		
LSE	6651	Skip if the printer done flag is a 1.
LCF	6652	Clear both sections of the printing buffer.
LLB	6654	Skip if line printer error flag is a 1.
LSD	6661	Clear line printer done and error flags.
LCB	6662	Load printing buffer from the contents of AC_{6-11} and clear the AC.
LPR	6664	Clear the format register, load the format register from the contents 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.
Line Printer Type LP08		
LSF	6661	Skip on character flag.
LCF	6662	Clear the character flag.
LSR	6663	Skip on error status line. (Error status line monitors interrupts caused by the printer not being ready; i.e., power error, paper jam, out of paper, etc.)
LPC	6664	Load the character into the print buffer and print if buffer is full or character was a control function. This instruction does not clear the AC.
LIN	6665	Set the program interrupts. Enable
LLS	6666	Microprogram combination of LCF and LPC.
LIF	6667	Clear the program interrupt flag.
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 AC_{6-11}, and the data ready flag is cleared.
RCRB	6634	The binary data in a card column is transferred into AC_{0-11}, and the data ready flag is cleared.
RCSP	6671	Skip if card reader card done flag is a 1.
RCSE	6672	Clear the card one 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 contents of the PC are incremented by one skipping the next sequential instruction.

Table C-1. IOT Instructions (cont)

Mnemonic	Octal	Operation
Automatic Magnetic Tape Control Type TC58 (cont)		
MTCR	6711	Skip on tape control ready (TCR). If the tape control is ready to receive an instructionn, the PC is incremented by one skipping 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.
MTRC	6724	Inclusively OR the contents of the command register into AC_{0-11}.
MTCM	6714	Inclusively OR the contents of AC_{0-5} and AC_{9-11} into the command register; JAM transfer bits 6, 7, 8 (command function).
MTLC	6716	Load the contents of AC_{0-11} into the command register.
	6704	Inclusively OR the contents of the status register into AC_{0-11}.
MTRS	6706	Read the contents of the status register into AC_{0-11}.
MTGO	6722	Set "to" bit to execute instructions in the command register if instruction is legal.
MCLA	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 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.

APPENDIX D
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-trail code
0	1	A1			.	A2			Absolute address to contain next 4 digits
0	0	A3			.	A4			
0	0	X1			.	X2			Contents of previous 4-digit address
0	0	X3			.	X4			
0	1	A1			.	A2			
0	0	A3			.	X2 A4			Address
0	0	X1			.	X2			
0	0	X3			.	X4			Content
		(Etc.)							(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 33 ASR. (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 1 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
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 1 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

Manually loading the RIM loader in core memory 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 and then press the FILL STEP 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.
7. To ensure that RIM is in core, press Exam. The MA will = 7756, and the MB should = the first RIM instruction (6032 for low-speed reader).
8. To check sequential core locations, press Step Exam, and observe the contents of the MB.

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 (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 a 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 selection 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 selection 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.

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			origin setting
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	
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

To load BIN

1. Select the 8 mode.
2. Press the IO Preset switch twice.
3. Put tape in the TTY reader, turn the reader on.
4. Press Start LS (still set to 7756).
5. When tape is read in, turn the reader off and stop the computer by pressing the Stop switch.

To load binary tapes with the BIN loader

1. Place 7777 in Left Switches.
2. Set bit 0 of Right Switches to 1 for low-speed reader (TTY) or set bit 0 of Right Switches to 0 for high-speed reader.
3. Press Start LS.
4. Tape is read in and computer halts.
5. If AC unequal to 0000 a checksum error has occurred.

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 perforated-tape 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

TAPE MAINTENANCE INSTRUCTIONS

E.1 GENERAL

In the Maintenance Mode, all signals from the tape transport are inhibited and data is simulated under program control. This mode is especially useful for troubleshooting and checking the operation of the TC12 Tape Control Logic.

E.2 TAPE MAINTENANCE INSTRUCTIONS

E.2.1 IOT 6141 6151 ?

This instruction is used to load the Tape Maintenance Instruction Register (used with IOT 6154) and to provide data and timing information to the TC12 tape control unit when it is operating in the maintenance mode. It also checks the status of the Tape Done flag. This instruction is not to be microprogrammed and it should not be used to generate timing information during tape transfers, although it can be used to set the Tape Maintenance Instruction Register at any time.

Any deviation from the above should be carefully checked with the logic diagrams to determine the expected result. The functions of the AC bits during this instruction are shown in Table E-1.

Table E-1. AC Bit Functions

AC Bit Number	Function
00	Load into the Tape Maintenance Register (Bit 0)
01	Load into the Tape Maintenance Register (Bit 1)
02	Load into the Tape Maintenance Register (Bit 2)
03	Load into the Tape Maintenance Register (Bit 3)
04	Clear the Tape Done Flag
05	Skip if Tape Done Flag is set
06	Generate TT0
07	Generate TT3
08	Simulate input data Mark Channel
09	Simulate input data Data Channel 1
10	Simulăte input data Data Channel 2
11	Simulate input data Data Channel 3

E.2.2 **IOT 6152**

This instruction is used to generate various control pulses within the TC12 Tape control unit. These control signals are used primarily for maintenance purposes. In special situations these controls are useful for handling nonstandard tape formats.

The bits of instruction 6152 are gated with individual bits of AC register to generate a specific control function. These functions are intended to be used singly, and not while a tape instruction is in progress, and this instruction cannot be microprogrammed. Any deviation from the above should be carefully checked with the logic diagrams to determine the exact result. The functions of the AC bits during this instruction are shown in Table E-2.

Table E-2. AC Bit Functions

AC Bit Number	Function
00	Tape Preset
01	Shift RWB
02	TB → RWB
03	TB + TAC → TAC
04*	Clear 8 Block
05*	Set Direction Backward
06*	Select Unit number 1
07*	Set Direction Forward
08*	Set Write Sync
09*	Set Motion. Set Direction Forward if motion was cleared
10*	Select 8 Tape
11*	AC_{11} → 8 Write Flip-Flop

*These Bits are used in the TC12-F Tape Control

E.2.3 **IOT 6154**

This instruction is used to load and examine the various registers of the TC12 Tape Control Unit. The action of this instruction is controlled by the contents of the Tape Maintenance Instruction Register and this action is shown in Table E-3. This instruction is not to be used during the operation of a tape instruction.

Table E-3. IOT 6154 Effect of Tape Maintenance Instruction

Register Contents: Contents of bits 0, 1, 2, & 3		Function Performed
Binary	Octal	
0000	00	AC → TB
0001	01	AC → TBN
0010	02	AC → TAC
0011	03	AC → TMA
0100	04	TMA Setup → AC
0101	05	TBN → AC

Table E-3. IOT 6154 Effect of Tape Maintenance Instruction (cont)

Register Contents Contents of bits 0, 1, 2, & 3		Function Performed
Binary	Octal	
0110	06	TB → AC
0111	07	RWB → AC
1000	10	Mark Window → AC
1001	11	Control States, Timing → AC
1010	12	Units & MTN → AC
1011	13	Tape Inst → AC
1100	14	Misc. Status 1 → AC
1101	15	Misc. Status 2 → AC
1110	16	TMA → AC
1111	17	

E.2.4 **IOT 6154 Detailed Transfer Information**

The Tape Maintenance Instruction Register is decoded to give a count of 00_8 to 17_8. These count levels generate the signals necessary to enable the selected data onto the Tape Bus for a 12-bit parallel transfer between the AC and the Tape Registers. The following tables show the contents of the AC after the instruction associated with each table is given.

Table E-4. Tape Maintenance Instruction Register Equal 10_8

Mark Window → AC	
AC Bit	**Data Read into AC**
00	LWN Wind Shade (1)
01	LWN Wind 00 (1)
02	LWN Wind 01 (1)
03	LWN Wind 02 (1)
04	LWN Wind 03 (1)
05	LWN EM
06	LWN CM
07	LWN GM
08	LWN DM
09	LWN FM
10	LWN BM
11	LWN IM

Table E-5. Tape Maintenance Instruction Register Equals 11_8

Control States & Timing → AC	
AC Bit	**Data Read Into AC**
00	TAC = 7777
01	LCS Idle (1)
02	LCS Search (1)
03	LCS Block (1)
04	LCS Check Word (1)
05	LCS Turn Around (1)
06	LCS Write (1)
07	LCS Write Cycle (1)
08	LTD ACIP
09	LTD TTOK
10	LTD Timing OK
11	LTD Tape Fail Delay

Table E-6. Tape Maintenance Instructions Register Equals 12_8

Units & Motion – AC	
AC Bit	**Data Read Into AC**
00	LTC Unit 0 (0)
01	LTC Unit 1 (0)
02	LTC Unit 2 (0)
03	LTC Unit 3 (0)
04	LTC Unit 4 (0)
05	LTC Unit 5 (0)
06	LTC Unit 6 (0)
07	LTC Unit 7 (0)
08	LMU motion (1)
09	LMU Direction (1)
10	LCS Tape OK (1)
11	LTC Write EN (1)

Table E-7. Tape Maintenance Instruction Register Equals 13_8

TINSTR -- AC	
AC Bit	**Data Read Into AC**
00	LIN RDC
01	LIN RCG
02	LIN RDE
03	LIN MTB
04	LIN WRC
05	LIN WRG
06	LIN WRI

Table E-7. Tape Maintenance Instruction Register Equals 13_8 (cont)

TINSTR -- AC	
AC Bit	**Data Read Into AC**
07	LIN CHK
08	LIN I (1)
09	LGP GP 0 (1)
10	LGP GP 1 (1)
11	LGP GP 2 (1)

Table E-8. Tape Maintenance Instruction Register Equals 14_8

MSC Status 1	
AC Bit	**Data Read Into AC**
00	LTS PHASE
01	LIP IN PROGRESS (1)
02	LTS LC 00 (1)
03	LTS LC 01 (1)
04	Mark Chan Write
05	Data Chan 1 Write
06	Data Chan 2 Write
07	Data Chan 3 Write
08	LGP GP = GPC (1)
09	LGP GPCNT 0 (1)
10	LGP GPCNT 1 (1)
11	LGP GPCNT 2 (1)

Table E-9. Tape Maintenance Instruction Register Equals 15_8

MSC Status 2	
AC Bit	**Data Read Into AC**
00	TC12-F WRITE SEL (1)
01	TC12-F TAPE SEL (1)
02–11	Not used

APPENDIX F
TABLES OF CODES

Table F-1. Model 33/35 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	alt-mode	375
8	270	escape	233
9	271		

Table F-2. Model 33 ASR/KSR Teletype Code (ASCII) in Binary Form

1 = HOLE PUNCHED = MARK
0 = NO HOLE PUNCHED = SPACE

MOST SIGNIFICANT BIT
LEAST SIGNIFICANT BIT

				8	7	6	5	4	S	3	2	1
	@	SPACE	NULL/IDLE				0	0		0	0	0
	A	!	START OF MESSAGE				0	0		0	0	1
	B	''	END OF ADDRESS				0	0		0	1	0
	C	#	END OF MESSAGE				0	0		0	1	1
	D	$	END OF TRANSMISSION				0	0		1	0	0
	E	%	WHO ARE YOU				0	0		1	0	1
	F	&	ARE YOU				0	0		1	1	0
	G	,	BELL				0	0		1	1	1
	H	(	FORMAT EFFECTOR				0	1		0	0	0
	I	)	HORIZONTAL TAB				0	1		0	0	1
	J	•	LINE FEED				0	1		0	1	0
	K	+	VERTICAL TAB				0	1		0	1	1
	L	,	FORM FEED				0	1		1	0	0
	M	—	CARRIAGE RETURN				0	1		1	0	1
	N		SHIFT OUT				0	1		1	1	0
	O	/	SHIFT IN				0	1		1	1	1
	P	0	DCO				1	0		0	0	0
	Q	1	READER ON				1	0		0	0	1
	R	2	TAPE (AUX ON)				1	0		0	1	0
	S	3	READER OFF				1	0		0	1	1
	T	4	(AUX OFF)				1	0		1	0	0
	U	5	ERROR				1	0		1	0	1
	V	6	SYNCHRONOUS IDLE				1	0		1	1	0
	W	7	LOGICAL END OF MEDIA				1	0		1	1	1
	X	8	S 0				1	1		0	0	0
	Y	9	S 1				1	1		0	0	1
	Z	:	S 2				1	1		0	1	0
	[	;	S 3				1	1		0	1	1
	\	<	S 4				1	1		1	0	0
	]	=	S 5				1	1		1	0	1
	↑	>	S 6				1	1		1	1	0
RUB OUT	←	?	S 7				1	1		1	1	1

1	0	0	SAME
1	0	1	SAME
1	1	0	SAME
1	1	1	SAME

Table F-3. LT-37 Transmit and Receive Code Table

37 KSR MODE				33 ASR MODE			
SHIFT		NOT SHIFT		SHIFT		NOT SHIFT	
CHARACTER	CODE	CHARACTER	CODE	CHARACTER	CODE	CHARACTER	CODE
A	101	a	341			A	301
B	102	b	342			B	302
C	303	c	143			C	303
D	104	d	344			D	304
E	305	e	145			E	305
F	306	f	146			F	306
G	107	g	347			G	307
H	110	h	350			H	310
I	311	i	151			I	311
J	312	j	152			J	312
K	113	k	353			K	313
L	314	l	154			L	314
M	115	m	355			M	315
N	116	n	356			N	316
O	317	o	157			O	317
P	120	p	360			P	320
Q	321	q	161			Q	321
R	322	r	162			R	322
S	123	s	363			S	323
T	324	t	164			T	324
U	125	u	365			U	325
V	126	v	366			V	326
W	327	w	167			W	327
X	330	x	170			X	330
Y	131	y	371			Y	331
Z	132	z	372			Z	332
SPACE	240	0	60	SPACE	240	0	260
!	41	1	261	!	241	1	261
”	42	2	262	”	242	2	262
#	243	3	63	#	243	3	263
$	44	4	264	$	244	4	264
%	245	5	65	%	245	5	265
&	246	6	66	&	246	6	266
’	47	7	267	’	247	7	267
(	250	8	270	(	250	8	270
)	251	9	71	)	251	9	271
:	72	*	252	:	272	*	252
;	273	+	53	;	273	+	253
<	74	,	254	<	274	,	254
=	275	–	55	=	275	–	255
>	276	.	56	>	276	.	256
?	77	/	257	?	277	/	257
@	300	`	140	@	300		
[	333	{	173	[	333		
\	134	¦	374	\	334		

Table F-3. LT-37 Transmit and Receive Code Table (cont)

37 KSR MODE				33 ASR MODE			
SHIFT		NOT SHIFT		SHIFT		NOT SHIFT	
CHARACTER	CODE	CHARACTER	CODE	CHARACTER	CODE	CHARACTER	CODE
]	335	}	175	]	335	ALT	375
↑	336	~	176	↑	336	OR MODE	
→	137	DEL	377	→	337	DEL	377
NULL	000	DLE	220	NULL	200	DLE	220
SOH	201	DC1	21	SOH	201	DC1	221
STX	202	DC2	22	STX	202	DC2	222
ETX	3	DC3	223	EXT	203	DC3	223
EOT	204	DC4	24	EOT	204	DC4	224
ENQ	5	NAK	225	ENQ	205	NAK	225
ACK	6	SYNC	226	ACK	206	SYNC	226
BELL	207	ETB	27	BELL	207	ETB	227
BACK SPACE	210	CANCEL	30	BACK SPACE	210	CANCEL	230
→ TAB	11	EM	231	→ TAB	211	EM	231
NEW LINE	12	SUB	232	NEW LINE	212	SUB	232
v TAB	213	ESCAPE	33	v TAB	213	ESCAPE	233
FORM	14	FS	234	FORM	214	FS	234
RETURN	215	GS	35	RETURN	215	GS	233
So	216	RS	36	So	216	RS	236
Si	217	QS	237	Si	217	QS	237

NOTE

Letter codes transmitted from the 37 KSR keyboard in 33 mode will be upper case regardless of the position of the shift key; however, letter codes received by the KSR 37 in 33 mode will be printed in upper or lower case, as received.

Table F-4. Card Reader Codes

The following table gives the octal representation of the internal (binary) codes for the listed punch combinations. These internal codes are generated by the card reader and are transmitted to the PDP-12 upon execution of the appropriate IOT instruction. Any combination of punches which is not shown in the table is invalid, and the card reader can not detect invalid combinations.

Card Code Zone	Card Code Num.	Internal Code	Character	Card Code Zone	Card Code Num.	Internal Code	Character
—	—	00 0000	Space	12	7	11 0111	G
12	8·2	11 1010	[	12	8	11 1000	H
12	8·3	11 1011	.	12	9	11 1001	I
12	8·4	11 1100	<	11	1	10 0001	J
12	8·5	11 1101	(	11	2	10 0010	K
12	8·6	11 1110	+	11	3	10 0011	L
12	8·7	11 1111	↑	11	4	10 0100	M
12	—	11 0000	&	11	5	10 0101	N
11	8·2	10 1010	!	11	6	10 0110	O
11	8·3	10 1011	$	11	7	10 0111	P
11	8·4	10 1100	*	11	8	10 1000	Q
11	8·5	10 1101	)	11	9	10 1001	R
11	8·6	10 1110	;	0	2	01 0010	S
11	8·7	10 1111	\	0	3	01 0011	T
11	—	10 0000	–	0	4	01 0100	U
0	1	01 0001	/	0	5	01 0101	V
0	8·2	01 1010	]	0	6	01 0110	W
0	8·3	01 1011	,	0	7	01 0111	X
0	8·4	01 1100	%	0	8	01 1000	Y
0	8·5	01 1101	←	0	9	01 1001	Z
0	8·6	01 1110	>	—	0	00 1010	0
0	8·7	01 1111	?	—	1	00 0001	1
—	8·3	00 1011	#	—	2	00 0010	2
—	8·4	00 1100	@	—	3	00 0011	3
—	8·5	00 1101	'	—	4	00 0100	4
—	8·6	00 1110	=	—	5	00 0101	5
—	8·7	00 1111	"	—	6	00 0110	6
12	1	11 0001	A	—	7	00 0111	7
12	2	11 0010	B	—	8	00 1000	8
12	3	11 0011	C	—	9	00 1001	9
12	4	11 0100	D				
12	5	11 0101	E				
12	6	11 0110	F				

Table F-5. LP08 Line Printer Code

64-Character Printer

96-Character Printer

AC_5 AC_6 AC_7				0 0 0	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
AC_8	AC_9	AC_{10}	AC_{11}							
0	0	0	0		Space	0	@	p P	‘	p
0	0	0	1		!	1	A	Q	a	q
0	0	1	0		”	2	B	R	b	r
0	0	1	1		#	3	C	S	c	s
0	1	0	0		$	4	D	T	d	t
0	1	0	1		%	5	E	U	e	u
0	1	1	0		&	6	F	V	f	v
0	1	1	1			7	G	W	g	w
1	0	0	0		(	8	H	X	h	x
1	0	0	1		)	9	I	Y	i	y
1	0	1	0	PF	*	:	J	Z	j	z
1	0	1	1		+	;	K	[	k	{
1	1	0	0	FF	’	<	L	◇	l	¬
1	1	0	1	CR	—	=	M	]	m	}
1	1	1	0		•	>	N	Λ	n	|
1	1	1	1		/	?	O	♡	o	□

Table F-6. LP12 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 G
CABLE CONNECTIONS TO PDP-12 FRONT PANEL

Access to analog channels 10-17 and the extension scope socket is by means of the relay and analog input panel (see Figure 2-3). The addition of 16 channels of analog input with the AG12 option, 16 additional preamplifiers, connects channels 20_8 to 37_8 to the analog extension panel supplied with the option.

Analog knobs 0_8 to 7_8, analog channels 10_8 to 37_8, console scope display, and extension scope display utilize the following connectors and connect to the following module slots in the EM12 memory wing:

Device	Front Panel Connector	Module Slot
Analog Knobs 0_8-7_8	5K, 10-turn Potentiometers	F33
Analog Chan 10_8-17_8	Switchcraft JAX #13B	F32
Analog Chan 20_8-27_8	Amphenol 26-4401-32P	F31
Analog Chan 30_8-37_8	Amphenol 26-4401-32P	F30
Extension Scope Display	Amphenol 26-4401-24P	F39
Console Display	Amphenol 57-30240	F38

Table G-1 lists cable connections and signal names from each front panel termination to its corresponding slot in the EM12 memory wing.

Table G-1. Cable Connections for PDP-12 Front Panel

DEVICE	SIGNAL NAME	FRONT PANEL PIN	SIGNAL NAME	FRONT PANEL PIN	MODULE SLOT PIN
Analog	+7 volts		- Analog Chan 10	Analog Phone Jack 10	A1
Knobs	- 7 volts		+ Analog Chan 10	Analog Phone Jack 10	B1
0-7	Analog Chan 7	All Knobs	- Analog Chan 11	Analog Phone Jack 11	C1
	Analog Chan 6	All Knobs	+ Analog Chan 11	Analog Phone Jack 11	D1
	Analog Chan 5	Knob 7	- Analog Chan 12	Analog Phone Jack 12	E1
Analog	Analog Chan 4	Knob 6	+ Analog Chan 12	Analog Phone Jack 12	H1
Channels	Analog Chan 3	Knob 5	- Analog Chan 13	Analog Phone Jack 13	J1
10-17	Analog Chan 2	Knob 4	+ Analog Chan 13	Analog Phone Jack 13	K1
	Analog Chan 1	Knob 3	- Analog Chan 14	Analog Phone Jack 14	L1
	Analog Chan 0	Knob 2	+ Analog Chan 14	Analog Phone Jack 14	M1
		Knob 1	- Analog Chan 15	Analog Phone Jack 15	N1
		Knob 0	+ Analog Chan 15	Analog Phone Jack 15	P1
			- Analog Chan 16	Analog Phone Jack 16	S1
			+ Analog Chan 16	Analog Phone Jack 16	T1
			- Analog Chan 17	Analog Phone Jack 17	U1
			+ Analog Chan 17	Analog Phone Jack 17	V1
Analog	- Analog Chan 20	1	- Analog Chan 30	1	A1
Channels	+ Analog Chan 20	2	+ Analog Chan 30	2	B1
20-37	DRAIN	3	DRAIN	3	GND
	- Analog Chan 21	4	- Analog Chan 31	4	C1
	+ Analog Chan 21	5	+ Analog Chan 31	5	D1
	DRAIN	6	DRAIN	6	GND
	- Analog Chan 22	7	- Analog Chan 32	7	E1
	+ Analog Chan 22	8	+ Analog Chan 32	8	H1
	DRAIN	9	DRAIN	9	GND
	- Analog Chan 23	10	- Analog Chan 33	10	J1
	+ Analog Chan 23	11	+ Analog Chan 33	11	K1
	DRAIN	12	DRAIN	12	GND
	- Analog Chan 24	13	- Analog Chan 34	13	L1
	+ Analog Chan 24	14	+ Analog Chan 34	14	M1
	DRAIN	15	DRAIN	15	GND
	- Analog Chan 25	16	- Analog Chan 35	16	N1
	+ Analog Chan 25	32	+ Analog Chan 35	32	P1
	DRAIN	31	DRAIN	31	GND
	- Analog Chan 26	30	- Analog Chan 36	30	S1
	+ Analog Chan 26	29	+ Analog Chan 36	29	T1
	DRAIN	28	DRAIN	28	GND
	- Analog Chan 27	27	- Analog Chan 37	27	U1
	+ Analog Chan 27	26	+ Analog Chan 37	26	V1
	DRAIN	25	DRAIN	25	GND
Console	DSX Chan Sel. (1) H	1			A1
Display	System GND	2			B1
and	DRAIN	3			GND
Extension	DSC Intensity L	4			C1
Scope	System GND	5			D1
Display	DRAIN	6			GND
	X HQ GND	7			E1
	DSX X Defl.	8			H1
	DRAIN	9			GND
	Y HQ GND	10			J1
	DSY Y Defl.	11			K1
	DRAIN	12			GND
Type 503 Scope	503 Intensify	19			S1
Connections	System GND	20			T1
	DRAIN	21			GND

APPENDIX H
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

$10^{\pm n}$ IN Octal

10^n	n	10^{-n}	10^n	n	10^{-n}
1	0	1.000 000 000 000 000 000 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

Binary Scale

Addition

$$0 + 1 = \begin{matrix} 0 + 0 = 0 \\ 1 + 0 = 1 \\ 1 + 1 = 10 \end{matrix}$$

Multiplication

$$0 \times 1 = \begin{matrix} 0 \times 0 = 0 \\ 1 \times 0 = 0 \\ 1 \times 1 = 1 \end{matrix}$$

Octal Scale

Addition

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

Multiplication

1	02	03	04	05	06	07	10	11	12
2	04	06	10	12	14	16	20	22	24
3	06	11	14	17	22	25	30	33	36
4	10	14	20	24	30	34	40	44	50
5	12	17	24	31	36	43	50	55	62
6	14	22	30	36	44	52	60	66	74
7	16	25	34	43	52	61	70	77	106

Mathematical Constants in Octal Scale

π = 3.11037 552421₈	e = 2.55760 521305₈	γ = 0.44742 147707₈
π^{-1} = 0.24276 301556₈	e^{-1} = 0.27426 530661₈	ln γ = − 0.43127 233602₈
$\sqrt{\pi}$ = 1.61337 611067₈	$\sqrt{e}$ = 1.51411 230704₈	$\log_2 \gamma$ = − .0.62573 030645₈
ln π = 1.11206 404435₈	$\log_{10} e$ = 0.33626 754251₈	$\sqrt{2}$ = 1.32404 746320₈
$\log_2 \pi$ = 1.51544 163223₈	$\log_2 e$ = 1.34252 166245₈	ln 2 = 0.54271 027760₈
$\sqrt{10}$ = 3.12305 407267₈	$\log_2 10$ = 3.24464 741136₈	ln 10 = 2.23273 067355₈

Powers of Two

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

Octal-Decimal Integer Conversion Table

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

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

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

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

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

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

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

Octal-Decimal Integer Conversion Table (Cont)

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

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

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

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

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

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

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

Octal-Decimal Integer Conversion Table (Cont)

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

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

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

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

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

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

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

Octal-Decimal Integer Conversion Table (Cont)

	0	1	2	3	4	5	6	7
6000	3072	3073	3074	3075	3076	3077	3078	3079
6010	3080	3081	3082	3083	3084	3085	3086	3087
6020	3088	3089	3090	3091	3092	3093	3094	3095
6030	3096	3097	3098	3099	3100	3101	3102	3103
6040	3104	3105	3106	3107	3108	3109	3110	3111
6050	3112	3113	3114	3115	3116	3117	3118	3119
6060	3120	3121	3122	3123	3124	3125	3126	3127
6070	3128	3129	3130	3131	3132	3133	3134	3135
6100	3136	3137	3138	3139	3140	3141	3142	3143
6110	3144	3145	3146	3147	3148	3149	3150	3151
6120	3152	3153	3154	3155	3156	3157	3158	3159
6130	3160	3161	3162	3163	3164	3165	3166	3167
6140	3168	3169	3170	3171	3172	3173	3174	3175
6150	3176	3177	3178	3179	3180	3181	3182	3183
6160	3184	3185	3186	3187	3188	3189	3190	3191
6170	3192	3193	3194	3195	3196	3197	3198	3199
6200	3200	3201	3202	3203	3204	3205	3206	3207
6210	3208	3209	3210	3211	3212	3213	3214	3215
6220	3216	3217	3218	3219	3220	3221	3222	3223
6230	3224	3225	3226	3227	3228	3229	3230	3231
6240	3232	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	.0003[illegible]	.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[illegible]	.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

Index

D

J-K

L

M

N

O

P

Q-R

S

T

U-V-W

X-Y-Z

READER'S COMMENTS

PDP-12 SYSTEM REFERENCE MANUAL
DEC-12-SRZC-D

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

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

What features are most useful? ______________________________

What faults do you find with the manual? ______________________________

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

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

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

Please describe your position. ______________

Name ______________ Organization ______________

Street ______________ Department ______________

City ______________ State ______________ Zip or Country ______________

Fold Here

Do Not Tear - Fold Here and Staple

FIRST CLASS
PERMIT NO. 33
MAYNARD, MASS.

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

Postage will be paid by:

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