DEC-9L-GRVA-D

# PDP-9/L USER HANDBOOK

JUNE 1968

DIGITAL EQUIPMENT CORPORATION • MAYNARD, MASSACHUSETTS

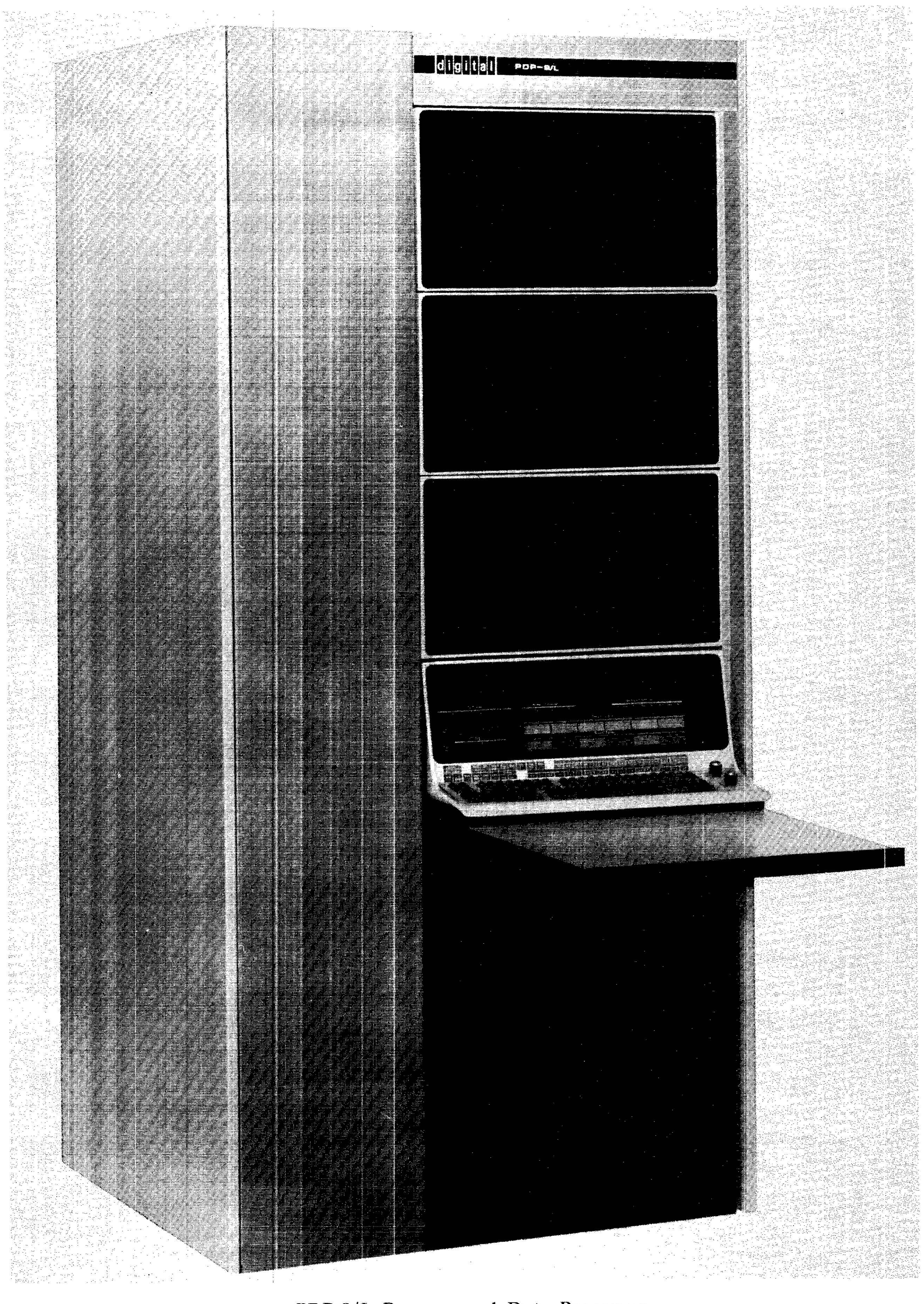
# TABLE OF CONTENTS

## TABLE OF CONTENTS (continued)

TABLE OF CONTENTS (continued)

# LIST OF ILLUSTRATIONS

# LIST OF TABLES



PDP-9/L Programmed Data Processor

# CHAPTER 1
# SYSTEM INTRODUCTION

## GENERAL

The PDP-9/L® programmed data processing system is a general purpose computer, incorporating FLIP CHIP hybrid integrated circuits throughout. The PDP-9/L features:

- High performance at low cost
- Demonstrated reliability
- Simple input/output interfacing
- Extensive software

Flexible, high capacity, input/output provisions coupled with a complete line of peripheral equipment allow system planning to satisfy a variety of applications. The PDP-9/L can be easily configured to perform equally well the role of central data processing facility, control element, or satellite processor. The ease with which its modular hardware and software adapt to the requirements of data acquisition, process control, and on-line processing in real-time environments makes it the ideal small scale system for scientific and industrial use.

The PDP-9/L system is a single address, fixed word length (18 bits), parallel binary computer. The minimum system configuration (see frontispiece) has 4096 words of core memory storage, paper tape input and output, console teleprinter keyboard input and printer output at 10-Hz (ASR-33).

The system readily interfaces to optional peripherals such as punched card equipment, line printers, magnetic tape transports, analog-to-digital converters, digital-to-analog converters, CRT displays, data communication equipment, and disk systems. Equipment of special design is easily adapted for interfacing to the PDP-9/L. The FLIP CHIP module line offers proven reliability plus simple, inexpensive fabrication of compatible interface controls for special equipment, or for the special-purpose equipment itself. Peripherals can be interfaced to the system as processing requirements expand, without modification of the central processor.

## CHARACTERISTICS

*Complete cycle time* of 1.5 microseconds for the random access ferrite core memory.

*Real-time clock* (option) generates a clock pulse every 16.7 msec (every 20 msec for 50-Hz systems) to increment a time counter stored in system memory. The counter initiates a program interrupt when a programmed preset time interval is completed. The clock can be enabled or disabled under program control.

*True direct addressing* is provided for 4096 18-bit word locations in the basic core memory module configuration or any memory module containing up to 8129 words appended to the system. The system allows indirect addressing up to the memory expansion limit of 32,768 locations. Core memory is expanded in increments of 4096 words. System software expands to make efficient use of all available core memory storage.

*Power failure protection* can be optionally implemented to protect against data loss due to internal power interruptions. With this option, the PDP-9/L is unaffected by power interruptions of less than 25 msec duration. In the event of a longer interruption, the option can save the active register contents and automatically restart the interrupted program at a specified address when power is restored. Without the "power failure protection" option, power interruptions of 10 msec duration, or longer, may result in loss of active register contents and memory contents.

*Automatic readin* is provided of binary-coded programs from paper tape via the ASR-33 paper tape reader when provided. A user-initiated and hardware-implemented control transfers 18-bit words (three tape lines) from tape to a block of sequentially addressed core memory locations and executes the instruction defined by the last word without further user intervention.

® PDP is a registered trademark of the Digital Equipment Corporation

*A built-in test program,* user-initiated and hardware-implemented, circulates a self-incrementing count through all central processor registers for the purpose of validating both their operation and the internal transfer paths. The user can monitor and verify register operation by observing the respective register display on the control console.

*All input/output transfers* are executed in parallel bytes up to 18 bits in length.

*Bidirectional input/output bus* is provided for program controlled data/command transmissions between the central processor and up to 256 external devices. All program controlled I/O transfers pass through the central processor's accumulator (AC), the 18-bit primary arithmetic register. Memory referencing instructions convey data between the AC and system core memory. IOT (input/output transfer) instructions select appropriate devices and effect the data transfer between the AC and information registers in the devices.

*Eight buffered data channels* allow fast, nonoverlapping data transmission between system core memory and eight devices interfaced to the I/O bus. Data channel transfers occur via the memory buffer (MB) register in the central processor and do not disturb the contents of other major registers in the processor. Thus, a data channel transfer suspends rather than interrupts execution of the program in progress. The maximum transfer capacity of the data channel facility is between 160,000 and 220,000 words per second, depending on the mix of input and output transfers (each output transfer steals four machine cycles; each input transfer steals three cycles). Provisions are made in system memory for word counter registers and current address registers unique to each data channel.

*Program interrupt control* frees the program in progress from the necessity of monitoring the status of peripheral devices. The program continues until a device signals a request for service. A subroutine, entered automatically upon the processor's granting of the interrupt request, stores the interrupted program's status, determines the device making the request, and transfers control to the appropriate service subroutine. At completion of the device servicing, the interrupted program is restored to control. The program interrupt control facility is suitable for those peripheral devices having low data rates.

*Multilevel automatic priority interrupt option* (API) affords immediate access to device handling and data handling subroutines on a ranked priority basis. Of the eight priority levels added by this option the four higher levels are assigned to device use, and the lower four are assigned to software use. The priority levels are fully nested; i.e., a higher priority request can interrupt in-process servicing of a lower priority. The restoration of an interrupted service subroutine does not require additional programming considerations. Likewise, the return to an interrupted main program segment is easily implemented.

The granting of priority interrupt requests, at completion of the current instruction, is rated above program and program interrupt activity and below data channel or direct memory access channel activity, or real-time clock counting.

The API system has 32 channels of which 28 are allocated to external device interrupting (hardware priority levels) and 4 are allocated to programmed interrupting (software priority levels). A channel assignment defines the core memory location of the unique entry to an interrupt subroutine. Device channels function independently of priority; up to eight device channels may be assigned to the same priority level. Device channels also may be multiplexed without limit, in which case the channel address defines the entry to a search routine rather than unique entry to one routine.

Additional provisions include dynamic reallocation of device priority level assignments (device control must be designed with logic circuits to accomplish reassignment) and programmed raising of the active interrupt to a priority level higher than the normal assignment, when the situation requires exclusion of interrupt requests at specific priority levels. The API is program enabled or disabled. Specific devices can be inhibited from interrupting by appropriate control inputs to their interfaces.

*The basic machine* has fixed-point hardware capability and floating-point software capability for performing binary arithmetic in 1s and 2s complement notations. Floating-point software offers choice of 6 or 9 decimal digit precision. The program library supplied includes extensive repertoire of multi- and single-precision subroutines.

*Add or subtract* (complementary addition) is performed in 3 microseconds with fetch of operand from effectively addressed core memory location. Overflow indication is furnished for 1s complement addition where absolute value of algebraic summed result exceeds capacity of the accumulator ($2^{17}$ - 1). Algorithms for 2s complement addition and subtraction treat overflow from accumulator as a carry into a 1-bit register called the link.

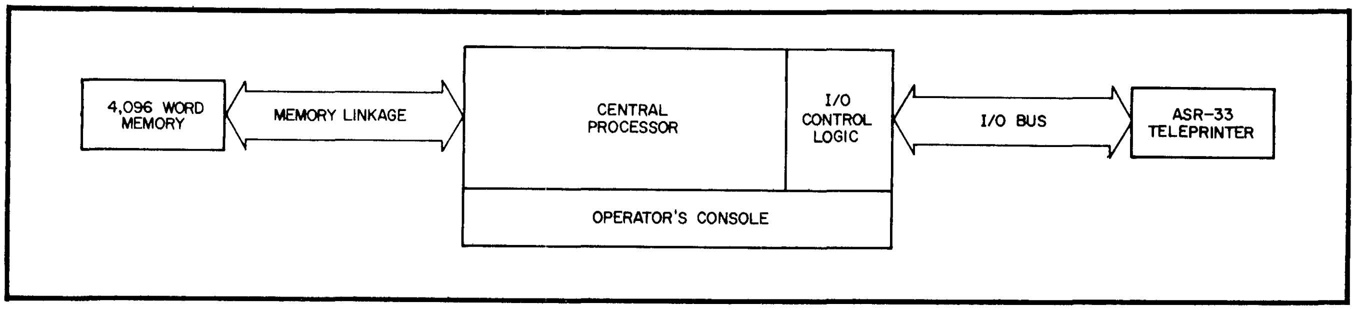


Figure 1-1. Basic PDP-9/L

*Extended arithmetic element option* offers fast, flexible, hardware execution of the following assigned or unassigned functions:

*Shifting* the contents of the primary arithmetic registers (AC or MQ), right or left, in 3 to 19 microseconds.

*Normalizing* the quantity in the primary arithmetic registers; i.e., shifting the contents left to remove leading binary 0s for the purpose of preserving as many significant bits as possible. Time required is 3 to 19 microseconds.

*Multiplication* in 4.5 to 12.5 microseconds.

*Division,* including integer and fractional, in 4.5 to 13.5 microseconds. Divide overflow indication is furnished when division would produce quotient exceeding $2^{17} - 1$ magnitude.

## DESIGN

The compactness of the PDP-9/L affords maximum computing facility in a minimum of space; its modular construction provides for ease of system growth to meet future processing requirements–external devices and additional core memory append with minimum effort and no effect on the central processor. (Chapter 4, Peripherals,

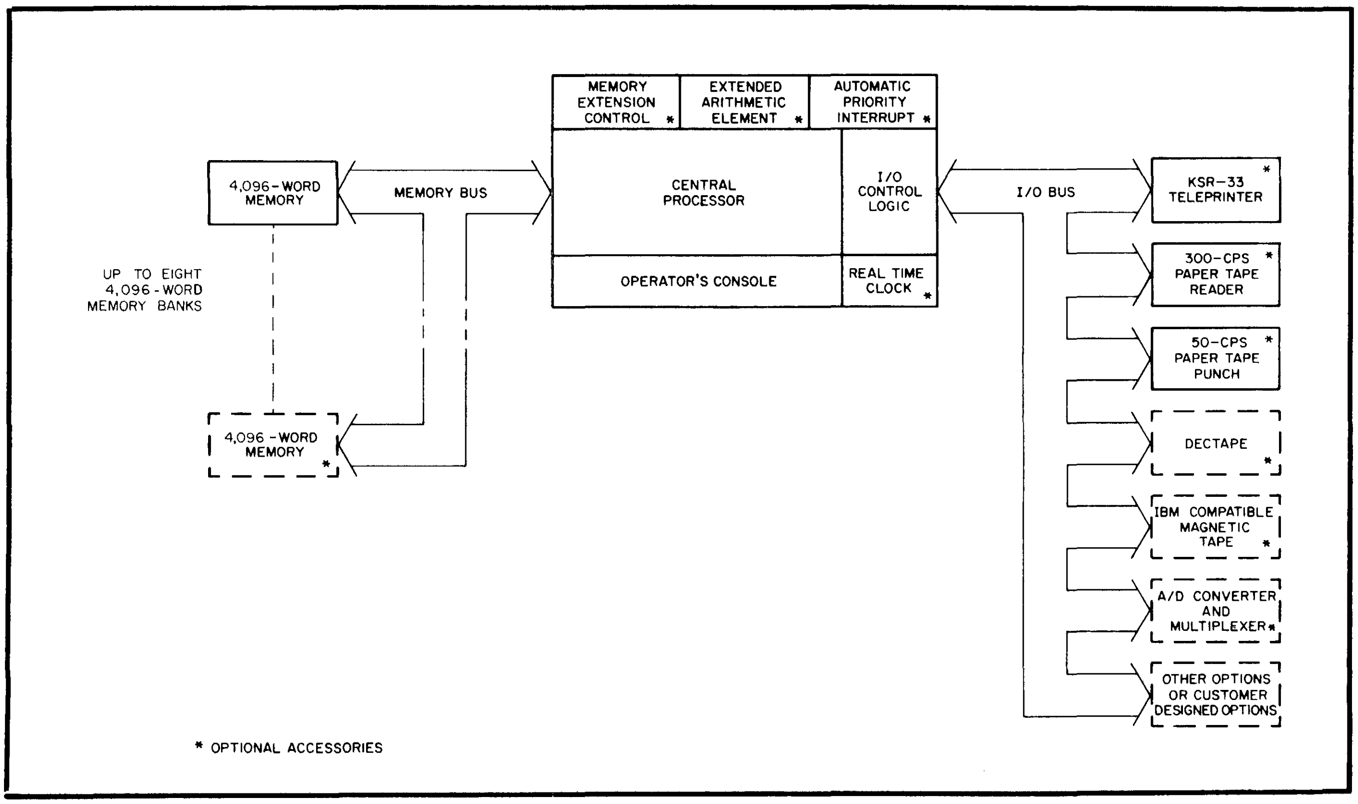

MEMORY EXTENSION CONTROL *
EXTENDED ARITHMETIC ELEMENT *
AUTOMATIC PRIORITY INTERRUPT *
4,096-WORD MEMORY
MEMORY BUS
CENTRAL PROCESSOR
I/O CONTROL LOGIC
I/O BUS
KSR-33 TELEPRINTER *
UP TO EIGHT 4,096-WORD MEMORY BANKS
OPERATOR'S CONSOLE
REAL TIME CLOCK *
300-CPS PAPER TAPE READER *
50-CPS PAPER TAPE PUNCH *
4,096-WORD MEMORY *
DECTAPE *
IBM COMPATIBLE MAGNETIC TAPE *
A/D CONVERTER AND MULTIPLEXER *
OTHER OPTIONS OR CUSTOMER DESIGNED OPTIONS
* OPTIONAL ACCESSORIES

Figure 1-2. Expanded PDP-9/L System Configuration, Block Diagram

presents complete details on interfacing special purpose or user-designed external device to the PDP-9/L input/output facilities.) PDP-9/L is completely self-contained, and does not require special air conditioning or humidity control. Internal power supplies generate all the required power from a 115-volt, 60-Hz, single-phase power source. Systems can be equipped to operate with 50-Hz power at a variety of voltage levels.

## CONFIGURATIONS

The basic PDP-9/L configuration (figure 1-1) consists of the following.

1. Central processor with integrated control console, work shelf and chair.

2. Core memory stack of 4096 18-bit words.

3. Teletype ASR-33 provides paper tape reader and punch and teleprinter. It operates at 10 character/second. (Teletype Modes ASR-35 can be optionally supplied and is recommended for applications where extreme use is to be made of the teleprinter's output function.)

4. Real-time clock (option).

5. Input/output facilities: I/O bus, eight data channels, program interrupt control, I/O status word provision, and conditional skip on external device status.

The PDP-9/L expands into a variety of configurations by:

Adding a 300 character/second paper tape reader and 50 character/second paper tape punch.

Increasing system core memory from the basic-supplied 4096 words up to 32,768 words in increments of 4096 words.

Adding peripheral equipment selected from the PDP-9/L line, or interfacing the system to special purpose or user-designed equipment.

Interfacing a basic expanded PDP-9/L to a data processing complex.

Incorporating central processor options to increase the system's computing and data handling power.

Figure 1-2 illustrates a typical expanded PDP-9/L system.

# CHAPTER 2
# SOFTWARE SYSTEMS

## GENERAL

PDP-9/L offers two complete software systems, PDP-9/L COMPACT and PDP-9 ADVANCED, plus an extensive library of arithmetic subroutines, utility programs, and maintenance and diagnostic routines.

The PDP-9/L COMPACT software functions in a paper tape input/output environment (i.e., source and object forms of programs reside on paper tape) with its field-tested components running in stand-alone fashion to provide the user with powerful single-job capabilities.

The PDP-9 ADVANCED software is an all new package of systems and utility programs, combining the latest concepts in device-independent programming with the power of FORTRAN IV and a macro assembler. PDP-9 ADVANCED software requires 8192 words of memory and is available in the following two compatible versions:

> Under control of a simple input/output monitor for PDP-9/L's with paper tape input/output or card input. This version will greatly expand the single-job capabilities available with PDP-9/L COMPACT software.
>
> In a more sophisticated monitor environment for all PDP-9/L's with some form of auxiliary bulk storage (DECtape, magnetic tape, or disk). This latter version will permit device independent programming under control of a keyboard monitor and a sophisticated input/output programming system.

## PDP-9/L COMPACT SOFTWARE

### Assembler

The PDP-9/L Assembler is a two-pass assembly which requires less than 3K of memory. It makes machine language programming on the PDP-9/L much easier, faster and more efficient. It permits the programmer to use mnemonic symbols to represent instruction operation codes, locations, and numeric quantities. By using symbols to identify instructions and data in his program, the programmer can easily refer to any point in his program without knowing actual machine locations.

Among the features of the assembler are a powerful set of pseudo operation instructions which are used to:

1. Reserve a block of core.
2. Set a desired radix.
3. Conditionalize sections of coding.
4. Output text strings.
5. Control the listing produced by the assembler.

The assembler also allows the user to replace symbolic data references by representing them as literals. The PDP-9/L assembler is upward compatible with MACRO-9 at the source language level. Input and output devices may be assigned by communicating with the assembler.

### Symbolic Editor

The symbolic editor of the PDP-9/L software system provides the user with the ability to conveniently create, examine and modify symbolic ASCII text material. The editor operates on lines of symbolic text, delimited by carriage return characters and organized into pages or blocks. These lines can be read into a buffer, selectively examined, deleted, or modified, and written out. New text may be substituted, inserted or appended.

Editor operation codes are divided into two basic categories: control instructions and editor commands. Control instructions determine the editor's operation level: input text or edit text. The input text level is used to create new symbolic material. The edit text level uses the editor commands which fall into four classes: I/O requests, pointer manipulation, editing requests, and examination requests. The pointer is a software device which places the current line in a special work area of core to facilitate edit request processing.

The editor is most frequently used to modify PDP-9/L source programs, but may be used to edit any symbolic ASCII text. It operates with all standard PDP-9/L peripheral devices.

## ODT-9

ODT-9 (Octal Debugging Technique) is a powerful program checkout aid which allows the user to carry on an interactive on-line debugging process with teletype commands and octal numbers. As errors are found, they may be corrected on-line and then the program can be executed immediately to test the correction. In general, the user types commands to ODT-9 which control the following functions:

1. Initiating the user's program.
2. Stopping and subsequently continuing the user's program at selected points called *breakpoints.*
3. Examining and/or modifying the accumulator and link at breakpoints.
4. Examining and/or modifying the contents of any memory location.
5. Searching user defined areas for memory locations containing specific bit configurations.
6. Dumping areas of memory for later loading, debugging and execution.

## TRACE-9

TRACE-9 is an on-line debugging tool which allows the user to trace the execution of specified portions of his program. In general, TRACE-9 generates a listing on the teleprinter which details the flow of control as the program executes. The information printed includes, for each traced instruction, the location of the instruction, the instruction itself, the location of the next instruction to be executed, and the values in the accumulator and link after execution of the instruction. TRACE-9 is controlled through teletype commands which specify tracing parameters such as:

1. Start address for tracing.
2. Stop address for tracing.
3. Various counts to specify when to start and stop.

Teletype commands may also be used to specify functions such as:

1. Suppressing the accumulator printout.
2. Printing only on CAL, JMS, JMP and SKIP instructions.
3. Tracing the program interrupts.
4. Halting or continuing the user's program upon completion of the trace.

## SCAN

SCAN is a small program used to scan areas of memory for particular bit configurations. The user specifies the start and stop address for the area to be scanned, the bit configuration to look for, and the bit positions to be tested (i.e., a mask). SCAN then scans the area. When a match is found, the address of the match is printed along with the unmasked matching word. Proper selection of the operating parameters allows SCAN to be used as a dump.

## FAST-9

FAST-9 (Fast Acquisition of System Tape) is a loading system for use in the PDP-9/L Software System to retrieve frequently used programs from DECtape and to create system tapes. The main advantages of the system are speed and ease of access.

The FAST-9 system tape, as distributed by DEC, contains commonly used Software System programs such as the Symbolic Editor, the PDP-9/L Assembler and ODT. Since these can be called from DECtape with only a small bootstrap, papertape handling is eliminated for systems programs. This results not only in a significant time savings, but also in increased reliability. FAST-9 is by no means restricted to Digital systems programs; it can very conveniently by employed for frequently accessed user created programs. This manual contains complete directions for use of the FAST-9 system tape as well as directions for adding user programs to the system.

## HRM-Puncher

The HRM Puncher is a self-relocating paper tape dump program. It may be loaded, by the Hardware Read-in (HRI), into any block of memory. It relocates itself and then punches out a user specified area of memory in the HRI format.

### Floating Point Package

These routines perform all arithmetic operations on normalized floating point data. The routines must be assembled with the user's program. He can use either single (20-bit accuracy) or double (35-bit accuracy) precision data. The actual arithmetic is performed using software accumulators by program FPOINT. Programs SINGLE and DOUBLE are required to perform the setup for FPOINT.

### Integer Arithmetic

These routines provide PDP-9/L users, without the EAE option, with simulated logical and signed multiply and divide routines.

### FLIO (Floating Input/Output)

These routines allow the user to input and output signed decimal data in floating point format, from either the teletype or paper tape reader.

### TOD (Teletype Octal Dump)

The TOD allows the user to obtain a typed output of the contents of any register or set of registers he specifies. This program is provided in two versions one for loading into high core (start at 7602) and one for low core (start at 22).

### TTYIO (Teletype Input/Output Conversion)

This package consists of routines which allow text to be input and output through the teleprinter. Formatting routines are also available in this package.

### DIP-OPS (Decimal Integer Print/and Octal Print Subroutines)

Allows the user to output his data to the teleprinter in any of four modes: 1) signed decimal, 2) right justified octal, 3) left justified octal, and 4) octal with no zero suppression.

### PTLIST

This routine lists a paper tape from either the paper tape reader or the keyboard, and prints it on the teletype.

### MTDUPE (Master Tape Duplicator)

The MTDUPE duplicates and verifies tapes. The MTDUPE duplicates any tape. By using switches the user is allowed to have a title punched in a readable format. The verifying routine checks the parity and will stop at every frame where the parity is off.

### Trig Functions

These routines allow the PDP-9/L user to perform trigonometric calculations on his floating point data in either single or double precision mode.

## PDP-9 ADVANCED SOFTWARE

PDP-9 ADVANCED software is an all-new completely relocatable software package combining sophisticated programming features with flexibility and ease of use. This new package includes a FORTRAN IV compiler, a macro assembler, an on-line debugging system, a symbolic editor, a peripheral interchange program, a linking loader, an input/output programming system, and a monitor.

Two versions of the PDP-9 ADVANCED software system are available: a simple input/output monitor system with 8K and paper tape input/output (using optional PC09A) and the option of punched card input facilities for basic PDP-9 users. The system will also include a more sophisticated monitor-based, device-independent system for PDP-9s with one or more forms of auxiliary bulk storage (magnetic tape, DECtape, or disk). Both systems are compatible.

### Paper Tape (or Card) System

Basic or extended-memory PDP-9/L systems without auxiliary bulk storage use the FORTRAN IV compiler, macro assembler (MACRO-9), debugging system (DDT-9), Symbolic Editor, peripheral Interchange Program (PIP-9), and Linking Loader under the control of the Input/Output Monitor. Only paper tape input/output or card input is provided. All systems programs have full features, are completely relocatable, and handle or produce relocatable code. Maximum utilization will be made of optional central processor features, such as the Extended Arithmetic Element and Automatic Priority Interrupt, when they are available.

**Device-Independent System**

The inclusion of bulk storage in a PDP-9/L will allow the use of the keyboard monitor-based, device-independent version of PDP-9 ADVANCED software. The minimum bulk storage requirements are:

2 DECtape transports (TU55) and control (TC02), or

2 IBM-compatible transports (TU20 or equivalent) and control (TC59), or

1 disk system and control

With the addition of one of these forms of bulk storage to a PDP-9/L system, the Keyboard Monitor (KM-9), and the Input/Output Programming System (IOPS) can be used to automatically store, retrieve, load and execute PDP-9 programs. With this device-independent version of PDP-9 ADVANCED software, the user can call his system programs from any bulk storage device (designated the system device), compile or assemble from any input device to any proper output device, and (*optionally*) obtain listings on any printing device or magnetic tape. By using the keyboard monitor to change the system and object program device assignment tables, the operator may designate at load time which devices shall serve as system device, object program storage, and data input and output devices. The linking loader, with access to the input/output programming system, automatically loads the input/output programs required by the device assignment tables.

**System Components**

The component packages of the new PDP-9 ADVANCED software system are described briefly below. More complete descriptions will be found in the individual language specification or operator's manual for each program.

**FORTRAN IV**

The PDP-9 FORTRAN IV compiler is a two-pass system which accepts statements written in the FORTRAN language and produces relocatable object code capable of being loaded by the linking loader program. It is fully compatible with USA FORTRAN IV, as defined in the USA Standard X3.9-1966, with the exception of the following few features which were modified to allow the compiler to operate in 8,192 words of core storage:

1. Complex arithmetic will not be available.
2. Adjustable array dimensions will not be allowed.
3. Blank COMMON will be treated as named COMMON.
4. The implied DO feature will be deleted from the DATA statement.

This FORTRAN IV compiler operates with the PDP-9 program interrupt facility enabled and generates real-time programs that both operate with the program interrupt enabled and can work in conjunction with assembly language programs that recognize and service real-time devices. Subroutines written in either FORTRAN IV or the Macro Assembler language can be loaded with and called by FORTRAN IV main programs. Comprehensive source language diagnostics are produced during compilation; a symbol table is generated for use in on-line debugging.

**Macro Assembler (MACRO-9)**

With the Macro Assembler, PDP-9/L users are able to utilize highly sophisticated macro generating and calling facilities within the context of a symbolic assembler. Among the more prominent features of MACRO-9 are:

1. The ability to define and call nested and recursive macros.
2. Conditional assembly based on the computational results of symbols or expressions.
3. Repeat functions.
4. A variety of fixed- and floating-point symbolic formats for constants.
5. Boolean manipulation.
6. Optional octal and symbolic listings.
7. Three forms of radix control (binary, octal, decimal) and two text modes (ASCII and 6-bit trimmed ASCII).
8. Global symbols for easy linking of separately assembled programs.
9. Choice of output format: relocatable, absolute binary (checksummed); full binary (unchecksummed), capable of being loaded via the READ IN key (console).
10. Ability to call input/output system macros which expand into IOPS calling sequences.

**Debugging System (DDT-9)**

DDT-9 adds the flexibility of relocatability and real-time operation to the capabilities of PDP-9 BASIC DDT. With it, the user may load and operate his program in a real-time environment while maintaining strict control over running of each section. DDT-9 allows the operator to insert and delete break-points, examine and change registers, patch programs, and search for specific constants or word formats.

The DDT breakpoints features allow for the insertion and simultaneous use of up to four breakpoints and any one or all of these breakpoints may be removed with a single keyboard command. The search facility allows the operator to specify a search through any part, or all of an object program with printout of the locations of all registers that are equal (or unequal) to specified constant. This search feature also works for portions of words as modified by a mask.

With DDT-9, registers may be examined and modified in either instruction format or octal code, and addresses may be specified in symbolic relative, octal relative, or octal absolute. Patches may be inserted in either source language or octal.

**Symbolic Editor**

The Symbolic Editor of the PDP-9 ADVANCED software system provides the ability to read symbolic text from any input device (paper tape reader, card reader, disk, drum, DECtape, magnetic tape, etc.), to examine and correct it, and to write it on any output device. It can also be used to create new symbolic programs.

The Editor operates on lines of symbolic text, delimited by carriage return (C/R) characters and organized into blocks or pages. These lines can be read into a buffer, selectively examined, deleted, or modified, and written out. New text may be substituted, inserted, or appended. Among the types of caommands available to the operator through the Symbolic Editor are:

1. *Input Operations*
   a. Read a page of text.
   b. Read n lines of text.
   c. Skip a page of text.
   d. Skip n lines of text.
2. *Examination Operations*
   a. Print the entire buffer.
   b. Print n pages
   c. Print lines n through m.
   d. Print line n.
   e. Print the entire buffer without comments.
   f. Print lines n through m without comments.
   g. Print line n without comments.
   h. Search for and print the next location symbol.
   i. Back up and print the previous line.
3. *Editing Operations*
   a. Delete line n.
   b. Delete lines n through m.
   c. Append text.
   d. Insert text before line n.
   e. Change line n.
   f. Change lines n through m.
   g. Read the next n lines and insert them after line m in the buffer.
   h. Erase the buffer.
   i. Move a line.
4. *Output Operations*
   a. Output the entire buffer.
   b. Output line n.
   c. Output lines n through m.
   d. Punch a form feed.

Several commands which combine input and output commands are also available to the user. With these, the buffer may be cleared and a new page automatically read in after the output of an edited page, or one or more pages may be read and punched in one operation (for duplication of sections of a symbolic tape).

**Peripheral Interchange Program (PIP-9)**

The primary function of PIP-9 is to facilitate the manipulation and transfer of data files from any input device to any output device. It can be used to update file descriptions, delete, insert, or combine files, perform code conversions, rewind tapes, and retrieve storage space that is no longer required by a file device.

**Linking Loader**

The Linking Loader has the task of loading any PDP-9 FORTRAN IV or MACRO-9 object program which exists in relocatable or absolute format. Among its tasks are loading and relocation of programs, loading of called subroutines, retrieval and loading of implied subroutines and IOPS routines, and loading and relocation of the necessary symbol tables.

**Input/Output Programming System (IOPS)**

IOPS provides a standardized programming interface to all input/output devices in a PDP-9/L system. Symbolic input/output unit assignments allow users' programs to select I/O devices specified symbolically to the macro assembler or the FORTRAN IV compiler. IOPS consists of a modular collection of relocatable input/output and utility subroutines which transmit data to and from peripheral devices and make data readily available for processing. IOPS data and file handling routines include device independence for all systems programs.

Specifically, IOPS provides the user with three levels of I/O programs.

1. Device handling - providing the basic subroutines to allow the user to operate a device, doing code conversion where required.
2. Data handling - providing the data buffering and internal line and character transmission facilities.
3. File handling - providing for the manipulation of named files on the system level.

IOPS completely eliminates the need for the programmer to program the standard peripheral devices. The programmer is no longer concerned with input/output problems such as timing, overlap, and the differences in the characteristics of peripheral devices. This permits him to concentrate on his primary task, the processing of data inside the computer.

IOPS is coded on a modular basis to allow addition of new devices and modifications to the handling of current devices. It requires considerably less effort to modify an IOPS subprogram than to revise individual input/output sections to fit a new configuration. By the use of IOPS, programs can be changed more rapidly to fit the expanding configurations.

**Input/Output Monitor**

The simple input/output monitor includes the device assignment tables and input/output routines necessary for programs being run on PDP-9/L's without bulk storage. The function of the monitor is to handle all input and output for system programs or users' programs so that the programmer need not be concerned with device or data manipulation routines. Normally only paper tape input/output or card input is allowed by the input/output monitor.

**Keyboard Monitor (KM-9)**

The Keyboard Monitor allows convenient single job processing through commands to the console teleprinter. Its primary functions are: to allow the user direct and immediate access to system user programs stored on bulk storage devices; to facilitate the creation and storage of new programs by the user; and to allow input/output device-independent programming with "load-time" specification of peripherals.

KM-9 consists of bootstrap loader, a keyboard listener program, a monitor command decoder, IOPS routines, an error diagnostic program, and device assignment tables (DATs).

The bootstrap loader always resides in upper memory and is responsible for loading the monitor into lower memory. Return calls from system or user programs cause restoration of control to the Monitor. The keyboard listener (KLIST) accepts input commands from the teletype and handles Monitor initialization and bookkeeping.

The Monitor command decoder (MCD) recognizes requests for system programs and loads the system loader to bring in the requested program. In response to keyboard commands, it also manipulates the device assignment tables to provide the device-independent programming features. The Monitor IOPS include data handling subroutines, and device handlers and interrupt service routines for the Teletype keyboard and printer. The routines for the system device will also be present with the Keyboard Monitor, but all other IOPS routines will be stored on the system device until required by object programs.

The monitor also contains device assignment tables. The purpose of the DATs is to relate logical I/O units to the actual hardware devices. Each input or output reference within a system or user program refers to a position in a device assignment table. This table, in turn, contains a device assignment for each table entry that may be used. Since the contents of the tables can be altered by commands to the Keyboard Monitor, actual I/O devices may be changed without altering the program references to these devices.

### Expansion of PDP-9 ADVANCED Software System

The above-mentioned features of the PDP-9 ADVANCED software system are available at this time. Included in the initial system design, but not scheduled for completion until November 1968 are:

> Time-shared use of the PDP-9/L by a protected priority foreground program and unprotected system or user background program. This feature will utilize a background/foreground monitor and will require 16,384 words of core, bulk storage, and the memory protection option.
>
> Batch processing from any peripheral device, including the ability to accept monitor commands from any system input device.

Although these features will not be available with the initial versions of PDP-9 ADVANCED software, all necessary provisions have been made for them in the system programs.

## MAINDEC DIAGNOSTIC PROGRAMS

MAINDEC diagnostic programs are provided for locating hardware malfunctions within the processor, memory, and I/O equipment.

These programs are designed to make troubleshooting fast and straightforward by selectively exercising every circuit in the machine. Instructions and procedures for loading, operating, and interpreting the results of diagnostic tests are written in clear, simple language, so that beginning maintenance technicians can use them easily.

Detailed error messages are printed out to tell the technician exactly which instruction, or bit configuration, has failed. Error codes direct the troubleshooter to specific modules when a fault condition is detected.

The program consists of two parts: the Basic Processor Test and the Extended Processor Test. The Basic Test incrementally checks the entire instruction repertoire, performing 1500 unique tests, and in each case, halts with specific instructions for the troubleshooter. If the Basic Test fails to detect the trouble, the Extended Test uses random number techniques to test the logic for many combinations of data manipulation and addressing problems, runs memory test patterns, performs system tests on I/O devices and controls, and many other tests.

A valuable tool for check-out and troubleshooting, MAINDEC diagnostics contribute to the high productivity of the PDP-9/L by minimizing downtime.

# CHAPTER 3
# SYSTEM ORGANIZATION

## GENERAL

The major functional components of the PDP-9/L system are the central processor, core memory, and input/output facilities. This chapter discusses these components and the options offered for each.

## CENTRAL PROCESSOR UNIT

Figure 3-1 illustrates the internal organization of the PDP-9/L central processor. Employing a transfer bus system, data is jam-transferred between registers at DC levels to minimize timing problems (level logic is used instead of pulse logic in the central processor). All active registers use simple circuit designs (no logic delays).

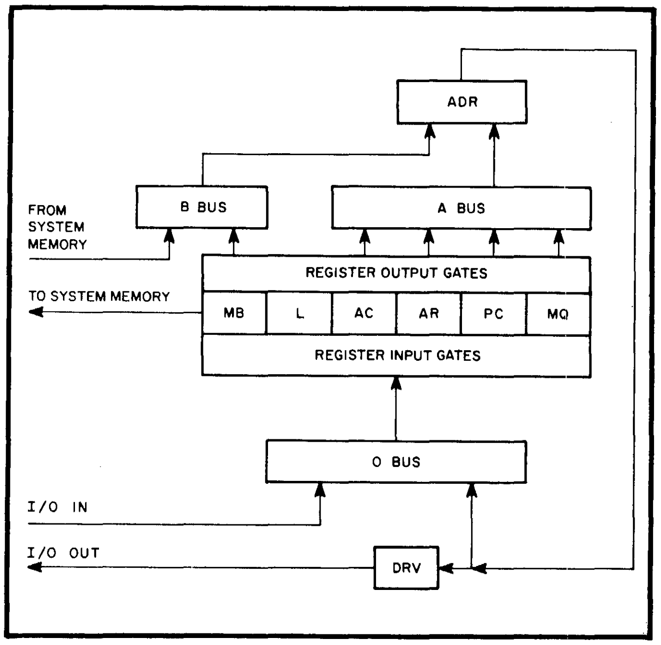


Figure 3-1. Central Processor-Major Register Organization

Control Elements (not illustrated) that govern the gating of information transfers include:

*Instruction Register (IR)* - The IR accepts the five most significant bits of each instruction word fetched from memory. The four most significant bits constitute the operation code and, when decoded, indicate the entry point to the control memory microinstruction sequence necessary to effect system response. The fifth bit signals when the fetched instruction indicates indirect addressing.

*Control Memory (CM)* - The CM stores all sequences of internal microinstructions required to fetch and execute a program's instructions, to effect operation of the data channels, and to respond to operator commands initiated at the control console. It is a very fast, read only, magnetic-core storage unit, prewired with the sequences.

*Control Register (CR)* - The CR delivers gate control signals to the transfer busses and to the active registers. The register supplies new address information to the CM based on conditions sensed.

Major registers in the processor are:

*Adder (ADR)* - The 19-bit ADR functions as a fast adder for arithmetic operations, and as the transfer path for all inter-register transfers and shift operations. It also increments the PC and MB registers, as required. Entry to the ADR is via the A bus and/or the B bus, under control of CR-developed gating control levels. The ADR operates at a 5 mc rate to provide an inter-register transfer time of 200 nsec.

*Accumulator (AC)* - The AC, an 18-bit register, retains the result of arithmetic/logical operations for the interim between instructions. The AC can be cleared and complemented. Its contents can be rotated right or left with the link. The contents of the memory buffer register can be added to the contents of the AC with the result left in the AC. The contents of both registers can be combined by the logical operations AND and exclusive OR, the result remaining in the AC. The inclusive OR can be formed between the AC and the Data switches on the operator console and the result left in the AC. For all program controlled transfers, information is transferred between core memory and an external device through the accumulator.

*Link (L)* - This 1-bit register is used to extend the arithmetic capability of the accumulator. In 1s complement arithmetic, the link is an overflow indicator; in 2s complement arithmetic, it logically extends the AC to 19 bits and functions as a carry register. The program can check overflow into the link from the accumulator to greatly simplify and speed up single and multiple precision arithmetic routines. The link can be cleared and complemented and its state sensed independent of the AC. It is included with the AC in rotate operations and in logical shifts.

*Arithmetic Register (AR)* - The AR functions with the AC to perform arithmetic and logical operations. It is not accessible to the programmer. Its operation is a function of the micro-instruction sequences in the CM.

*Multiplier-Quotient Register (MQ)* - The optionally implemented extended arithmetic element (EAE) adds the logic of the MQ to the basic PDP-9/L. The MQ is 18 bits long and holds the multiplier during multiplying instructions and receives the low-order 18 bits of the resulting product. During division operations it holds the low-order 18 bits of the dividend and, at the completion of the divide instruction, it contains the quotient. It can also be used an an extension of the AC for 36-bit shift operations and for data normalizing operations.

*Program Counter (PC)* - The PC determines the program sequence; that is, the order in which instructions are performed. This 13-bit register contains the address of the memory cell from which the next instruction is to be taken. Addition of the memory extension control option expands the PC to 15 bits for the addressing of up to 32,768 locations.

*Memory Buffer Register (MB)* - All information transferred into or out of core memory passes through the 18-bit MB. Information is read from a memory cell into the MB and rewritten into the cell in one cycle time. Instructions and data are brought from core memory into the MB for processing. The MB serves also as a buffer for information transferred between core memory and an external device in data channel transfers.

## CORE MEMORY

The PDP-9/L utilizes a 4-wire 30-stack core memory with a complete cycle time of 1.5 microseconds. Each 4,096 word core memory module contains a core stack, sense amplifiers, drivers, and a memory address (MA) register. The MA sets up the memory location (address) to be used for data retrieval or storage.

System core memory can be expanded from the basic 4,096 words up to 32,768 words in 4,096 word increments. Expansion beyond 8,192 words requires implementation of the optional Memory Extension Control, Type KG09A to extend the PDP-9/L addressing capability.

## INPUT/OUTPUT FACILITIES

The following text briefly describes the input/output facilities provided with the PDP-9/L in its minimum (basic) configuration. Detailed descriptions concerning the operation and use of these facilities are presented in chapter 9, Input/Output Operations.

Basic PDP-9 I/O facilities include:

1. A I/O bus system which chain links all the device controls for all peripheral devices to the central processor unit (CPU).
2. A data channel control governing concurrent (non-overlapping) operation of eight data channels.
3. A program interrupt control.
4. An I/O status-read provision.
5. A conditional skip-on-device-status provision.

The PDP-9/L apportions I/O control between the CPU and the various device controls interfaced to the I/O facilities. The complexity of any device control is thus a function of the type of device and of the facilities which it must make use of to accomplish its system purpose. This scheme has several benefits for the user.

1. It negates the need for expensive I/O processors or controllers.
2. The structure of the I/O system can be expanded or reconfigured at anytime without modification of the CPU.
3. User-designed or special purpose equipment can be easily interfaced to the PDP-9 through the inexpensive fabrication of the required device control units.

Peripheral equipment may either be asynchronous with no timed transfer rates or synchronous with a timed-transfer rate. Devices such as the CRT displays, teleprinter-keyboard, and the line printer can operate at any speed up to the maximum without loss of efficiency. These asynchronous devices remain on and ready to accept data; they do not turn themselves off between transfers. Devices such as magnetic tape, DECtape, and card equipment are timed-transfer devices and must operate at or near their maximum speeds to be efficient.

The I/O bus consists of command lines and bidirectional data lines for use in accomplishing program-controlled transfers or data channel transfers; plus the use of the program interrupt control, the I/O status-read and conditional skip-on-device-status provisions, and the Automatic Priority Interrupt system option Type KF09A, if implemented. The program-controlled mode functions with single word or byte (up to 18 bits, parallel) transfers made under control of programmed instructions. The data channel mode permits block transfers at high speed without interruption of the program in progress.

All I/O data transfers function with the precedence of the following priority structure.

1. Data channel requests
2. Real-time clock counting (clock is considered to be an I/O device)
3. Priority interrupts, 8 levels (optional)
4. Program interrupts
5. Main program in progress (lowest priority)

A higher priority request for service interrupts any in-process service of a lower priority at the end of the current instruction. Program interrupts and priority interrupts require that the main program transfer control to specific service subroutines. These routines restore control to the program at completion of the service interval. Computer-granted breaks satisfy data channel requests; i.e., program execution is delayed but not disturbed while the data channel transfers information between memory and the requesting device via the MB.

**Program Controlled Transfers**

Program controlled transfers are made by IOT (input/output transfer) instructions contained in the main program or in service subroutines. These instructions are microcoded to effect response only for a particular device. The microcoding includes issuing a unique device selection code and appropriate processor-generated pulses to initiate the specified operation. All program controlled transfers are executed through the accumulator (AC) in parallel bytes up to 18 bits in length.

For an "out" transfer, the program reads a data word from memory into the AC. A subsequent IOT instruction places the data on the bus, selects the device, and causes it to enter the word in its data buffer register. For an "in" transfer, the process reverses. An IOT instruction selects the device and causes the contents of its data buffer to be gated onto the I/O bus. In turn, the word is strobed into the AC and read, by the program, into memory.

**Conditional Skip on Device Status**

The PDP-9/L order code has in its IOT family a group of instructions for testing the status of peripherals. An instruction of this type directs the processor to either skip or proceed to the next instruction as a result of the test. The feature can be thought of as programmed decision making. For example, a program accepting typed input from the console teleprinter might make use of the following sequence in a teleprinter service subroutine.

```
.
.
.
KSF          /SKIP NEXT IF CHARACTER IN BUFFER
JMP .-1      /RECHECK KEYBOARD BUFFER
KRB          /READ BUFFER INTO AC
.
.
.
```

Although simple to implement, I/O servicing of this form adds to a program's "overhead" as the subroutine remains in the two-instruction loop until the skipping condition is satisfied.

**Input/Output Read Status**

The input/output read status facility provides for programmed interrogation of external device status. Upon execution of an IORS (input/output read status) instruction, the states of those device flags (done, busy, not ready, etc.) interfaced to the facility by the I/O bus are transferred to specific assigned bit positions of the AC. The program may check for specific flag conditions or the user may view the flag states as selectively displayed in indicators on the control console. For bit assignments, see Figure 9-1.

**Program Interrupt**

The program interrupt (PI) facility offers a more efficient method of I/O servicing. The computer continues with execution of a program until a previously selected peripheral signals that it is ready. At that time, the program in process interrupts and transfers control to a service subroutine. When completed, the subroutine restores the computer to the status prior to the interrupt, allowing the interrupted program segment to continue. Where multiple peripherals are connected to the PI, a search routine with device status testing (skipping) instructions must be added to determine which device initiated the interrupt request. This routine transfers control to the appropriate I/O service subroutine. The PI is itself considered an I/O device in that a program or subroutine thereof can include instructions for enabling or disabling the facility. When disabled, the PI ignores all services requests. Such requests normally remain on-line, however, and are answered when the PI is again enabled. The PI is automatically disabled when an interrupt request is honored. The interrupt-accessed subroutine is responsible for re-activating the facility.

**Data Channels**

The eight data channels included in the basic PDP-9/L I/O facilities employ the I/O bus system for block data transfers between core memory and high data rate peripherals such as DECtape and standard magnetic tape transports. The data channel control can concurrently service up to eight devices. The priority of service is established by the hardware interface. Data channel requests for service are answered upon completion of the instruction currently being executed by the computer. An in-process data channel transfer cannot be interrupted by a data channel request of lower priority, but it can be interrupted by a direct memory access channel request for service at completion of the current machine cycle. An interrupted data channel transfer continues with completion of the DMA channel action. On-line data channel requests for service are answered in turn on the basis of their priority relationship.

Each data channel functions with processor-granted breaks to interleave its transfers with execution of the program in progress. These transfers occur via the MB and do not disturb the contents of other active registers in the processor (AC, PC, etc.). Data is read into memory in three machine cycles and out of memory in four cycles (the additional cycle allows I/O bus settling and the setting of control gates prior to the strobing of the data word into the device's buffer register).

The block transfer is initiated by an IOT (input/output transfer) instruction following initialization of a word count register and a current address register, both held in sequential memory locations for each data channel (30, $31_8$ for data channel 0; 32, 33 for 1; 34, 35 for 2; and 36, 37 for 3). Memory allocation for the remaining four channels is at the user's discretion. The word count register is initialized to minus the number of words to be transferred plus one. The current address register is initialized to the starting address minus one of the sequential block of memory locations which are to deliver data to or receive data from the peripheral device.

When the data channel request is granted by the processor, the device transmits the address of its data channel word count register. During the first cycle, the contents of this register are incremented by one and then the effective address of the current address register is established. In the second cycle, the contents of the current address register are incremented by one to establish the effective address of the memory location delivering or receiving the data word. During the third cycle, or fourth cycle in the case of out transfers, the actual data transfer occurs.

The device continues to request service until it receives an indication that the block transfer has been completed. At that time, it can initiate a program interrupt to access a subroutine for the re-initialization of the data channel's word count and current address registers. Because the block transfers are automatic in nature, the programmer need only concern himself with providing the appropriate subroutine or subroutines to initialize data channel operations.

**Add-to-Memory Capability**

This capability permits incrementing the contents of an externally specified memory location in one memory cycle, or adding the contents of an external register to the contents of a memory word in four memory cycles. A device on the DCH system can request these actions by signaling on appropriate lines of the I/O bus. These features were originally included in the extra-cost KH09A option.

The increment capability (MB+1) is commonly used when data in histogram form is desired (such as pulse height analysis data). After each data point is developed by external hardware, it is placed on the address lines (thereby specifying a memory location) and an Increment MB request is made. An I/O OVFLO pulse is returned to the device if the increment causes the specified word to go to zero.

The request for an increment cycle is honored after completion of the current instruction. If successive increments are requested, one instruction of the current program will be executed between each increment break.

The add-to-memory capability is used in signal averaging where on successive scans through an independent variable (such as time), the data from the dependent variable (such as voltage) is added into that already collected. Data placed on the data lines is added to the contents of the location specified by the data channel current address counter. A DATA OVFLO pulse is returned to the device if the sum exceeds $2^{18}$-1.

The request for an add-to-memory operation is honored after completion of the current instruction. If successive requests are made, they will be honored back-to-back, every four memory cycles.

## OPTIONS

Incorporation of the following central processor related options expands the data processing capabilities of the basic PDP-9 system, provides increased efficiency for input/output operations, and simplifies the programming and execution of arithmetic operations.

### Extended Arithmetic Element, Type KE09A

The extended arithmetic element facilitates high-speed multiplication, division, shifting, normalizing, and register manipulation. Installation of the EAE adds an 18-bit multiplier quotient register (MQ) to the computer as well as a 6-bit step counter register (SC). The contents of the MQ are displayed by the REGISTER indicators on the operator's console when the REGISTER DISPLAY control is in the MQ position. The option and the basic computer cycle operate asynchronously, permitting computations to be performed in the minimum possible time. Further, the EAE instructions are microcoded so that several operations can be performed by one instruction to simplify arithmetic programming. Average multiplication time is 11 microseconds; average division time is 12 microseconds. The PDP-9/L program library offers a complete package of single- and multi-precision routines for use with this option. EAE instructions are described in chapter 7, Instructions.

### Memory Extension Control, Type KG09A

The memory extension control allows expansion of PDP-9/L core memory from 8,192 to 32,768 words in increments of 4,096 words. The option includes a 2-bit extended program counter, 2-bit extended memory address register, and an extend mode control. Locations outside the current 8,192-word field are accessed by indirect addressing while in the extend mode. In this mode, bits 3-17 in the effective address of an indirectly addressed instruction specify the memory bank number (bits 3 and 4) and the memory address (bits 5-17). If not in the extend mode, bits 3 and 4 of the effective address are ignored and the bank number is taken from the extended program counter. Instructions for the option are discussed in chapter 6 under Extend Mode Addressing. The state (i.e., on or off) of the mode is automatically saved in the event of a program interrupt, or CAL or JMS instruction (refer to chapter 6 under "reserved address"). A program interrupt disables the extend mode. The saved state can be automatically restored by the instruction sequence of DBR followed immediately by JMP I, where the address for the latter instruction is 00000, 00020 or Y for, respectively, a program interrupt, CAL, or JMS (Y refers to the address previously specified by the JMS instruction).

### Additional Core Memory

Up to seven 4096 word core memory modules may be added to a PDP-9/L for expansion of random-access storage up to 32,768 18-bit words. The memory extension control Type KG09A is required when more than 8,192 words are implemented.

### Power Failure Protection, Type KP09A

The basic PDP-9/L is not affected by power interruptions of less than 10 msec duration. Active registers in the processor (AC, AR, PC, etc) will lose their contents for interruptions of longer duration but memory is not disturbed. The power failure protection option extends the period of nonaffect by power interruption to 25 msec. In addition, the option provides for the saving of active register contents in the event of longer power interrupts and the automatic restart of the system when power is restored. The restart feature is switch-selected by the operator to be enabled or disabled. When enabled, the program in progress resumes execution at location 00000. The system must be operating with the program interrupt facility enabled to sense the option's initiation of a program interrupt to save the register contents at the time of the line power failure. The option adds the following instruction:

| | |
|---|---|
| $703201_8$ - | Skip next instruction if power-low flag is set. |

### Memory Protection Option Type KX09A

The memory protection feature establishes a foreground/background environment for PDP-9/L processing activity by specifying the boundary between protected (lower) and unprotected (upper) regions of system core memory. Allocation of memory locations (in increments of 1024 words) to the protected region is dynamic and program controlled. A Boundary Register, added by the option, stores the location of the upper limit of the protected region. It is loaded from bits 3-7 of the AC by a MPLS instruction.

The KX09A monitors the instruction about to be executed, and transfers control to a monitor program (should the instruction be in the category of "illegal instructions") before the instruction is executed. If a program tries to reference a nonexistent memory bank, the KX09A, if it has been enabled, transfers control to the monitor program.

The memory protect (or user mode) may be enabled either by programmed instruction, or by placing the PRTCT switch on the console UP, and pressing the START key. When enabled, the option will trap the following:

IOT - Input/Output
CAF - Clear All Flags
XCT of XCT - Chained Execute Instructions
HLT - Halt
OAS/LAS - Load AC from Data Switches
References to nonexistent memory
References to locations below the boundary limit

Trapping causes the execution of an effective JMS instruction after the machine cycle that attempts to violate. The address referenced by the effective JMS instruction will be location absolute 20 if the program interrupt facility is disabled, or location absolute 0 if the program interrupt facility is enabled. The Violation Flag is set.

The NonExistent Memory Flag is also set if the violation was caused by a program or DCH reference to nonexistent memory.

User mode is disabled in the following ways:

I/O Reset Key
The detection of a violation
CAL Instruction (which never causes a violation)
A Program Interrupt
An API Interrupt

If user mode is enabled when an API break starts, and the API Channel Address contains a HLT, OAS, or IOT - rather than the normal JMS - that instruction will be inhibited, user mode will be disabled in the normal fashion and no violation will be detected.

If user mode is disabled when a reference to nonexistent memory is made, the Non-Existent Memory Flag is set, no trap occurs, and the program continues after a one microsecond pause. If a reference to nonexistent memory occurs when user mode is enabled, the Violation Flag is also set and the trap occurs.

HLT, OAS and IOT instructions are totally inhibited when the memory protect option is enabled. If the HLT or OAS is combined with any other operate group instruction (micro-programming), the other parts of the operate group instruction are executed before the trap. (The exception is SKP which is not executed - see Example 2). The second XCT in a chain of XCT instructions is trapped before execution.

The state of the protect mode (a "1" for user mode) is stored in bit 2 - the storage word by those operations that save the state of the machine (CAL, JMS, PI). The stored PC will contain one more than the location of the violating instruction, except for JMP to a protected area. In this case, the stored PC will contain the protected address.

The sole operator control is the PRTCT switch, which has an indicator above it. This indicator lights when in user mode. The PRTCT switch is used in conjunction with the START key to establish the proper mode at the beginning of program execution. If the switch is UP, then the program is started in user mode. The switch has no further effect.

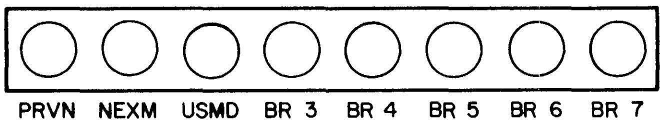
The IO RESET key clears the boundary register, Violation and NonExistent Memory Flags, and user mode (i.e., memory protect is turned off).

The option adds the instructions to the PDP-9/L listed in Table 3-1.

TABLE 3-1 MEMORY PROTECTION TYPE KX09A INSTRUCTIONS

| Mnemonic | Octal Code | Operation Executed |
|---|---|---|
| MPSNE | 701741 | Skip on NonExistent Memory Flag. The NonExistent Memory Flag is set whenever the processor attempts to reference a nonexistent area of core. For a 32K machine, the flag would never get set. |
| MPSK | 701701 | Skip on Violation Flag. The Memory Protect Violation Flag will be set whenever the execution of an instruction has violated the provision of memory protection (see above). |
| MPEU | 701742 | Enter user (protect) mode. Memory protect mode will be entered at the end of the next instruction that is not an IOT. |
| MPCV | 701702 | Clear Violation Flag. |
| MPCNE | 701744 | Clear NonExistent Memory Flag. |
| MPLD | 701704 | Load the memory protection boundary register with the contents of AC3-7. The boundary register will store the number of 1024 word blocks to be protected. |

Associated with the option are additional indicators (shown below) which are located on the main console.



| | |
|---|---|
| PRVN: | Lights when Violation flag is raised. |
| NEXM: | Lights when NonExistent Memory Flag is raised |
| USMD: | Lights to indicate that the memory protect mode (user mode) has been entered. Logically identical to the light above the PRTCT switch on the console. |
| BR 3 through BR 7 | Indicate the upper limit (in 1K increments of the protected region). |

## Examples

*Example 1*

```
 76
 77 - Main Program
100 - XCT 200
200 - XCT 247
247 - LAC 250
```

Should the main program reach location 100 with the user mode ON, the second XCT at location 200 will violate, and control will transfer to location 20 (assuming program interrupt facility is off). The contents of the PC (101) together with the state of the link, Extend Mode and User Mode will be stored in location 20, and the next instruction taken from location 21.

*Example 2*

```
 76
 77 - Main Program
100 - XCT 200
200 - HLT SKP
```

Here, the HLT will cause the violation, (one XCT is allowed). The SKP will be ignored and the PC will be stored as 101.

*Example 3*

```
 76
 77 - Main Program
100 - XCT 200
200 - HLT CLA
```

Same as Example 2 except the AC will be cleared before the trap occurs.

*Example 4*

Assume boundary to be at address 2000, protection is for register 0 through 1777.

```
 400 - 3000
3000 - TAD I 400
```

This is legal, although address 400 (which is below the boundary) has been referenced, because the final reference is above the boundary.

*Example 5*

Boundary again at address 2000

```
 400 - 1000
3000 - TAD I 400
```

This will be trapped, as address 1000 was the final reference, and address 1000 is below the boundary.

**Automatic Priority Interrupt, Type KF09A**

The automatic priority interrupt (API) system adds eight additional levels of programming priority to the PDP-9/L. The upper four levels are assigned to devices and are initiated by flags (interrupt requests) from these attached devices. The lower four levels are assigned to the programming system and are initiated by software requests. The priority network insures that high data rate or critical devices will always interrupt slower device service routines while holding still lower priority interrupt requests off line until they can be serviced. The API identifies the source of the interrupt directly, eliminating the need for a service routine to flag search.

The key elements in an interface to the API are priority level and channel. Each I/O device interfaced to the API is assigned to one of the four device priority levels so as to maximize performance of the I/O system.

The channel assignment of every device and software request is fixed and cannot be changed. Each of the 32 channels has a corresponding channel address in core memory. This address contains a JMS instruction to the service subroutine. The execution of 1 out of 32 unique JMS instructions is the API's method of directly identifying which source caused the interrupt.

The API operates in the following manner. An I/O device requests service by transmitting an interrupt request signal to the processor on a line corresponding to its specific, preassigned priority level. If this priority level is higher than the priority of the device which requested the currently active program segment, an interrupt is granted to the new device. Upon receipt of the grant signal, the device transmits its channel-address back to the processor. The processor executes the instruction in the specified memory address; this is always a JMS to the device service subroutine. The new priority level is remembered and no further servicing of this or lower priority levels is permitted until the device service subroutine is exited.

The hardware insures that simultaneous requests by multiple devices are handled in the proper priority sequence. If interrupt requests occur at different priority levels, the highest priority request will be serviced first. If multiple interrupt requests occur at the same priority level, the device closest on the bus to the processor will be serviced first. The entire API system may be enabled or disabled with a single instruction; however, many devices provide facility to separately connect and disconnect their flags from the interrupt.

A chief advantage of this API system lies in the proper use of the software levels. In the real-time environment, it is necessary to maintain data input/output flow, but it is not possible to perform long, complex calculations at priority levels which shut out these data transfers. With the API, a high priority data input routine which recognizes the need for the complex calculation can call for it with a software level interrupt. Since the calculation is performed at a lower priority than the data handling, the latter can go on undisturbed. Further, there is no need to interface the data collection routine with the lowest priority (background) program which may run independently of the real-time system. Refer to chapter 9 for descriptions of the API instructions and the programming considerations

# CHAPTER 4
# PERIPHERALS

## GENERAL

This chapter describes both the standard peripherals and the optional peripherals offered with PDP-9/L. Information regarding the instruction word format and the coding of the IOT (input/output transfer) instructions can be found in chapter 7 under IOT Instructions.

## STANDARD INPUT/OUTPUT EQUIPMENT

Standard input/output equipment supplied with each PDP-9/L consists of a Model ASR-33 Teletype and control, with associated perforated tape reader and perforated tape punch.

The Teletype Model ASR-33 consists of four functional units: keyboard, teleprinter, paper tape reader, and paper tape punch. All units operate at 10 characters per second. The input and output functions are independent, but the two input units (keyboard and reader) cannot act independently, nor can the two output units (teleprinter and punch).

The teletype interface is both full duplex and half duplex. A switch on the back of the I/O frame is used for selection. The COMPACT software system uses the full duplex teletype. The ADVANCED software system uses the half duplex teletype. (A full duplex interface is one that permits the input and output operations to proceed independently. If input characters are to be printed (echoed), they must be transmitted back out to the teletype by the computer. A half duplex interface does not permit independent input and output operations. Input characters are automatically printed without intervention by the program.)

The keyboard and teleprinter device flags are interfaced to the I/O skip facility, the program interrupt control, and to bits 3 and 4, respectively, of the IORS (input/output read status) word. (Refer to chapter 9, Input/Output Considerations, for a discussion on the use of the IORS word.) The state of these bits can be seen in the REGISTER indicators when the machine is in the stop condition and the REGISTER DISPLAY control is in the STATUS position.

### Keyboard

The keyboard control contains an 8-bit buffer which assembles and holds the code for the last character struck on the keyboard. The keyboard flag becomes a 1 to signify that a character has been assembled and is ready for transfer to the accumulator. This flag may be cleared by command.

The keyboard instructions are listed in table 4-1.

### Reader

Data from the reader enters the computer in the same way that keyboard data does. There

TABLE 4-1 KEYBOARD INSTRUCTIONS

| Mnemonic Symbol | Octal Code | Operation Executed |
|---|---|---|
| KSF | 700301 | Skip if the keyboard flag is set to 1. If the flag is a 0, the next instruction is executed. If it is 1, the next instruction is skipped. The flag is set only when a character has been completely assembled by the buffer. |
| KRB | 700312 | Read the keyboard buffer. The content of the buffer is placed in bits 10-17 of the cleared AC and the keyboard flag is cleared. Bits 0-9 of the AC remain cleared. |
| KRS | 700332 | Select the keyboard reader if the START switch is engaged. |

is an additional IOT instruction to initiate the reading of paper tape.

The reader instructions are listed in table 4-2.

### Teleprinter

The teleprinter control contains an 8-bit buffer which receives a character code from AC bits 10 through 17. The buffer receives the 8-bit code from the AC in parallel and transmits it to the teleprinter serially. When the function called for by the 8-bit code has been executed, the teleprinter flag is set to 1. This flag is connected to the computer program interrupt and input/output skip facility. It is cleared by programmed command.

The teleprinter instructions are listed in table 4-3.

A TLS instruction should not be executed until a TSF instruction verifies that the teleprinter flag is set. The teleprinter requires 110.04 msec to complete the action called for by an entered character code (print a character, line feed, carriage return, etc.). The teleprinter flag is again set at the end of this interval. The time between teleprinter flags is available to the program.

### Punch

Data for the punch is transmitted in the same fashion as for the teleprinter. Note that the punch enable switch on the ASR-33 must be turned on for punching to take place. The instructions for the punch are the same as for the teleprinter and are interpreted as directed in table 4-4.

## OPTIONAL PERIPHERALS

### Teletype Model 33 KSR and Control

The Teletype Model 33 KSR (keyboard send receive) is usually selected for systems using a high speed paper tape reader and punch (PC09A). The 33 KSR can be used to type in or print out information at a rate of up to ten characters per second. Signals transferred between the 33 KSR and the keyboard printer 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 zeros and ones in the control and computer. The start mark and subsequent 8-character bits are represented by single units of time duration followed by a 2-unit stop space.

TABLE 4-2 READER INSTRUCTIONS

| Mnemonic Symbol | Octal Code | Operation Executed |
|---|---|---|
| | 700332 | Clear reader flag. |
| KSF | 700301 | Skip if reader flag is a "1". |
| KRB | 700312 | Read reader buffer. |

TABLE 4-3 TELEPRINTER INSTRUCTIONS

| Mnemonic Symbol | Octal Code | Operation Executed |
|---|---|---|
| TSF | 700401 | Skip the next instruction if the teleprinter flag is set to 1. |
| TCF | 700402 | Clear the teleprinter flag. |
| TLS | 700406 | Load printer buffer. The content of AC bits 10-17 are placed in the buffer. The flag is cleared before transmission takes place and is set when the character has been printer. |

TABLE 4-4 PUNCH INSTRUCTIONS

| Mnemonic Symbol | Octal Code | Operation Executed |
|---|---|---|
| TSF | 700401 | Skip if punch flag is a "1". |
| TCF | 700402 | Clear punch flag. |
| TLS | 700406 | Load punch buffer. |

Each of the 64 printing characters and 32 control characters are represented by an 8-bit standard ASCII code. The Teletype eight-level character code is listed in appendix 2. As the teleprinter input and output functions are logically separate, the programmer can consider the printer and keyboard as individual devices. The console teletype interface is half duplex.

The keyboard and teleprinter device flags are interfaced to the I/O skip facility, the program interrupt control, and to bits 3 and 4, respectively, of the IORS (input/output read status) word. (Refer to chapter 9, I/O Considerations, for a discussion on use of the IORS word.) The state of these bits can be seen in the REGISTER indicators when the machine is in the stop condition and the REGISTER DISPLAY control is in the STATUS position.

**Keyboard**

The keyboard control contains a 8-bit buffer which assembles and holds the code for the, last character struck on the keyboard. The keyboard flag becomes a 1 to signify that a character has been assembled and is ready for transfer to the accumulator. This flag may be cleared by command.

The keyboard instructions are listed in table 4-1.

**Teleprinter**

The teleprinter control contains an 8-bit buffer which receives a character code from AC bits 10 through 17. The buffer receives the 8-bit code from the AC in parallel and transmits it to the teleprinter serially. When the function called for by the 8-bit code has been executed, the teleprinter flag is set to 1. This flag is connected to the computer program interrupt and input/output skip facility. It is cleared by programmed command.

The teleprinter instructions are listed in table 4-3.

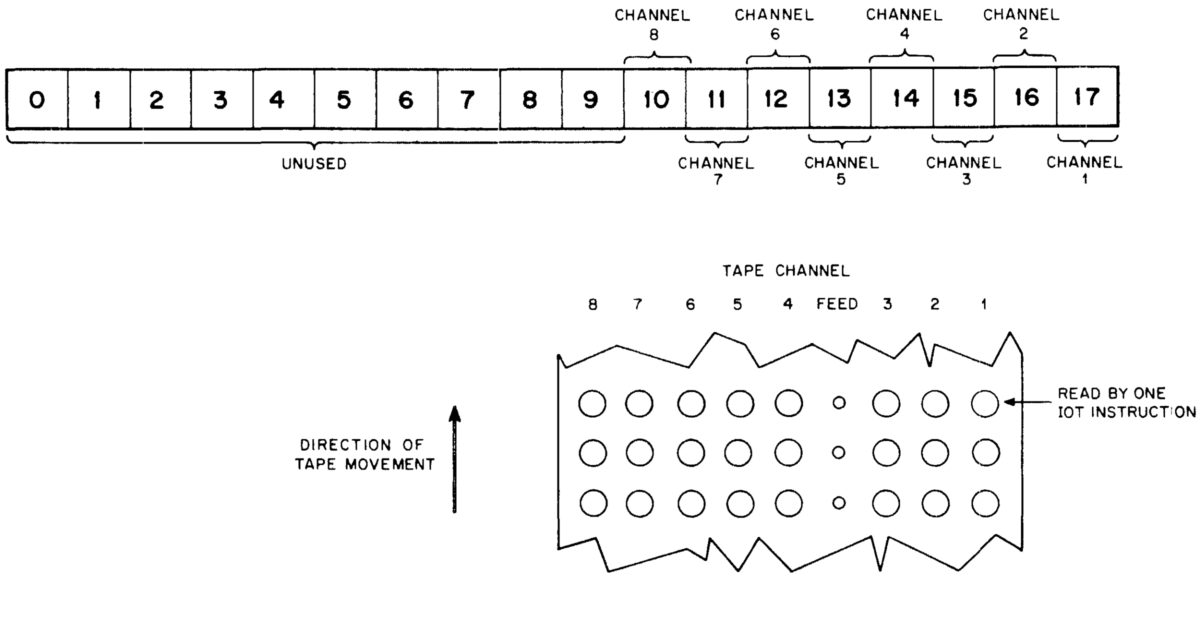
A TLS instruction should not be executed until a TSF instruction verifies that the teleprinter flag is set. The teleprinter requires 110.04 msec to complete the action called for by an entered character code (print a character, line feed, carriage return, etc.). The teleprinter flag is again set at the end of this interval. The time between teleprinter flags is available to the program.

NOTE: In half duplex mode, the keyboard has priority over the teleprinter; i.e., if a key is struck while the teleprinter buffer is transmitting a character code to the Teletype, the character code relating to the struck key will be entered in the keyboard buffer and the keyboard flag will be set to initiate a program interrupt. The disruption of the teleprinter function will garble the character code being sent to the Teletype from the teleprinter buffer. The Teletype action defined by this garbled code is unpredictable. Thus, one character of output data is lost.

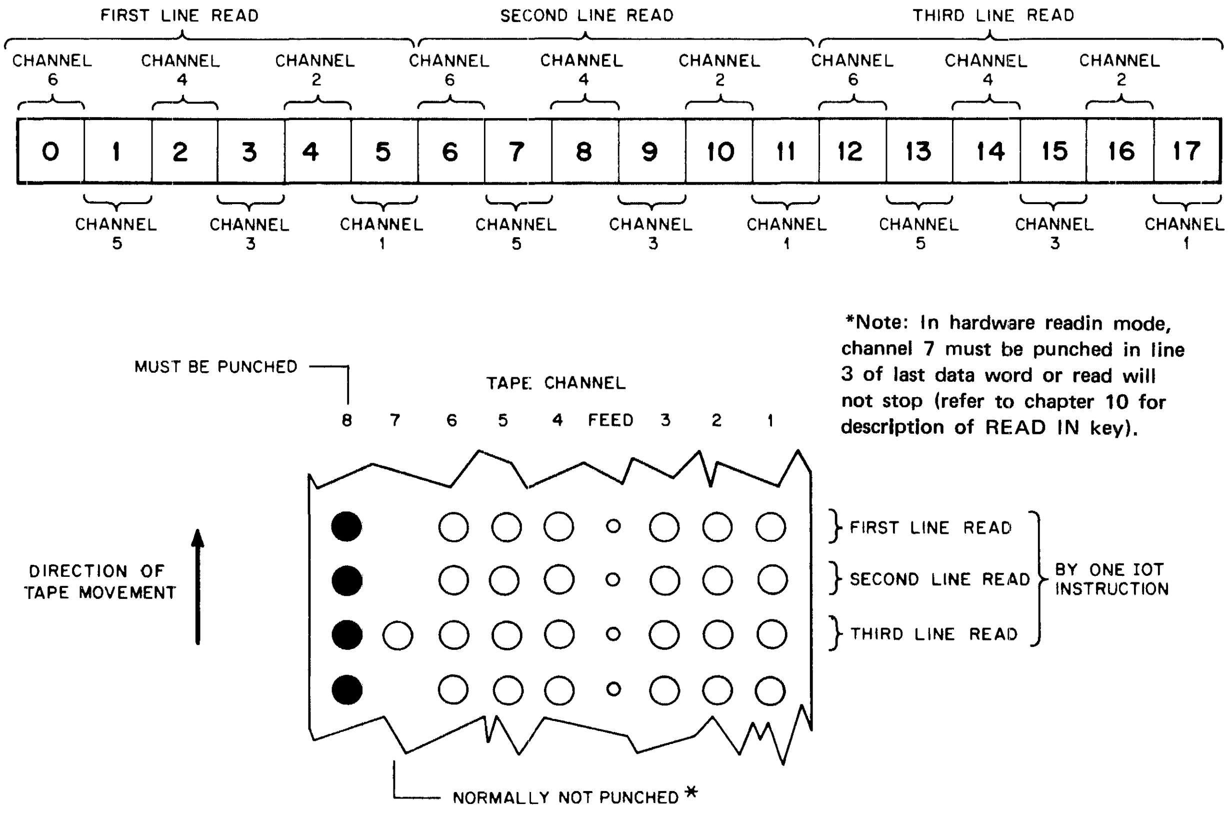
**Perforated Tape Reader Type PC09A**

The perforated tape reader and its associated control are designed and manufactured by the Digital Equipment Corporation. The reader functions with a stepping motor and feed-hole drive to photo-electrically sense 8-channel paper tape at the rate of 300 characters per second. The control requests reader motion, transfers data from the reader to the reader buffer register, and signals the computer when the buffer has assembled data for input to the computer. In order to maintain maximum reader speed (300 cps), a new select IOT must be issued within 1.67 msec.

Data may be read from tape in either alphanumeric or binary modes, as determined by IOT select instructions. In the alphanumeric mode (figure 4-1a), each select instruction causes one line of tape, consisting of eight bits, to be read and placed right justified in the 18-bit buffer register. In the binary mode (figure 4-1b), each select instruction causes three lines of tape to be



a. Tape Format and Reader Buffer Register Bit Assignments in Alphanumeric Mode



*Note: In hardware readin mode, channel 7 must be punched in line 3 of last data word or read will not stop (refer to chapter 10 for description of READ IN key).

b. Tape Format and Reader Buffer Register Bit Assignments in Binary Mode

Figure 4-1. Perforated Tape Format

read. Six bits of each tape line read are assembled in the buffer to form one 18-bit computer word. The seventh bit of a tape line is ignored. A line is not read, however, unless the eighth bit is punched. A "reader buffer" instruction transfers the contents of the reader buffer to the computer's accumulator.

Reader facilities include a right-hand bin for supply of the tape being read, a left-hand bin for receiving the tape, and a feed-through control to complete passage of the tape into the receiving bin following the readin operation. A snap-action tape retainer on the reader platform allows simple tape loading.

Primary power is made available to the reader when the computer POWER control is set to ON. All operations of the reader are under program control.

The tape reader device flag is interfaced to the I/O skip facility, the program interrupt control, and to bit 1 of the IORS (input/output read status) word. The tape-reader-no-tape flag* is interfaced to bit 8 of the IORS word. (Refer to chapter 9, I/O Considerations, for a discussion on use of the IORS word.) The state of these status bits can be seen in the REGISTER indicators when the machine is in the stop condition and the REGISTER DISPLAY control is in the STATUS position. The tape reader device flag is also interfaced with the API (priority level, address 50).

Tape Reader IOT Instructions - Observe the following sequence for instructions required to effect the transfer from paper tape to the AC.

1. Select the mode and clear the buffer.**
2. Wait for reader flag (indicates that buffer has been loaded).
3. Transfer buffer contents to the AC.

*Note: All program tapes should be provided with trailers (i.e., sections of feed-hole only punched tape) since the reader may not halt immediately upon detection of a no-tape indication. Reading of a non-trailered tape will result in the entry of invalid data as the reader indexes beyond the end of tape point. A no-tape condition, in addition to setting the tape-reader-no-tape flag, sets the reader flag to initiate a program interrupt.

**Note: A programmed check of tape presence (i.e., not a no tape condition) may precede this sequence (refer to IORS discussion in chapter 9).

The tape reader IOT instructions are listed in table 4-5.

*Tape Reader Use* - In loading tape for readin, observe the following practices:

1. Raise the tape retainer and load the tape into the right-hand bin with channel one (figure 4-1) toward the rear. Place several folds of the tape leader in the left-hand bin and position the tape on the platform with the feed holes engaged by teeth of the drive gear. Snap the retainer down.
2. Momentarily depress the tape feed control. This action corrects any misalignment of the tape with respect to the drive teeth, and it clears the reader out-of-tape flag.
3. Set the address switches (numbered 3-17 on the console) to the starting address for the readin and depress the I/O Reset key and then the READ IN key to initiate reading of the tape.

## Perforated Tape Punch

The perforated tape punch unit, packaged on the same chassis as the tape reader, consists of a solid-state control and a mechanical punch mechanism. It perforates paper tape at the rate of 50 characters per second. When the punch is selected by an IOT instruction, data in the accumulator is transferred to the punch buffer and then without further instruction punched in the tape.

A magazine for unpunched tape and a box for tape chad are located internally. Both are accessible when the reader-punch drawer is pulled forward on its slides. Power for the punch motor is available when the computer POWER control is in the ON position. The motor runs only when the punch has been selected, however. Operation of the punch is by programmed instructions. An out-of-tape switch, on the punch mechanism and through which the unpunched tape passes, closes to inhibit punch operation when approximately one inch of tape is left.

The paper tape punch device flag is interfaced to the I/O skip facility, the program interrupt control, and to bit 2 of the IORS (input/output read status) word. The tape-punch-no-tape flag* is interfaced to bit 9 of the IORS word. (Refer

*Note: The tape-punch-no-punch condition will not initiate a program interrupt.

TABLE 4-5 TAPE READER IOT INSTRUCTIONS

| Mnemonic Symbol | Octal Code | Operation Executed |
|---|---|---|
| RSF | 700101 | Skip the next instruction if reader flag is a 1. |
| RCF | 700102 | Read reader buffer. Clear reader flag, then inclusively OR contents of reader buffer into the AC. RB V AC $\rightarrow$ AC |
| RRB | 700112 | Read reader buffer. Clear reader flag. Clear AC and then transfer contents of reader buffer to AC. RB $\rightarrow$ AC |
| RSA | 700104 | Select reader in alphanumeric mode. One 8-bit character is read and placed in the reader buffer (right justified). The reader flag is cleared before the character is read. When transmission is complete, the flag is set to 1. |
| RSB | 700144 | Select reader in binary mode. Three 6-bit characters are read and assembled in the reader buffer. The flag is cleared during assembly and set when character assembly is completed. |

to chapter 9, I/O Considerations, for a discussion on use of the IORS word.) The state of these status bits can be seen in the REGISTER indicators when the machine is in the stop condition and the REGISTER DISPLAY control is in the STATUS position.

Information is handled by the punch in either alphanumeric or binary modes. In the alphanumeric mode each select instruction causes one line of tape, consisting of eight bits to be punched. Each hole punched in a tape channel corresponds to a binary 1 in the appropriate bit of the punch buffer. A feed hole is punched for each command, even if the buffer contains all zeros. The correlation between tape channels and accumulator bits shown in figure 4-1a applies to the tape punch in alphanumeric mode. In the binary mode each select instruction causes one line of tape, consisting of eight bits, to be punched. Holes are punched in channels 6 through 1 as a function of binary ones in bits 12 through 17 of the accumulator, respectively. Channel 8 is always punched and channel 7 is normally not punched, thereby conforming to standard binary tape information format.

*Use of Paper Tape Punch* - The paper tape punch is operated by programmed instructions. The functions of the buttons located on the punch enclosure are included here for user convenience.

FEED Button - while the button is depressed, the punch produces feed hole only punched tape for tape leader or trailer purposes.

Out-of-Tape Button - this button functionally inhibits program use of the punch by simulating the out-of-tape condition for the punch. Since punch I/O routines normally verify that the punch out-of-tape flag is not set before selecting the device, this simulation permits the user to replenish the tape magazine when its contents have been exhausted, splice the new tape to the existing tape (the FEED button can be used to produce whatever length of feed hole tape is necessary), and then deactivate the out-of-tape flag at his convenience. Without this provision, the punch could be program operated the instant that the out-of-tape switch on the punch mechanism opened as the new tape was fed in.

TABLE 4-6 TAPE PUNCH IOT INSTRUCTIONS

| Mnemonic Symbol | Octal Code | Operation Executed |
|---|---|---|
| PSF | 700201 | Skip the next instruction if the punch flag is set to 1. |
| PCF | 700202 | Clear the punch flag. |
| PSA | 700204 | Punch a line of tape in alphanumeric mode. The punch flag is immediately cleared and then set when punching is complete. |
| PSB | 700244 | Punch a line of tape in binary mode. The punch flag is immediately cleared and then set when punching is complete. |

NOTE: The following microcoded instruction causes the punching of just a feed hole.

| | | |
|---|---|---|
| | 700214 | Clear AC and punch a feed hole. |

**Card Reader and Control Type CR02B**

*Overall Description* - The CR02B Card Reader reads 80 column 12-row punched cards at rates up to 200 cards per minute. A select instruction starts a card moving past the read station. Information is transferred into a 12-bit register, one column at a time, and a column flag is set. Upon sensing the column flag, the computer reads the data register into the AC, under program control. Once a card is selected, all 80 columns must be read. The card may be selected in either of two modes. In the binary mode, information is transferred into the data register directly as 12 rows of information. In the alphanumeric mode, the 12 rows of information are encoded into Hollerith card code and transferred into the data register as a 6-bit code. Table 4-7 lists the IOT instructions for the CR02B card reader and control.

*Operation* - The read sequence is started by the issuance of a select command. Once the command to select a card is given, the card reader reads all columns in sequence. To read a column, the program must respond to a column ready flag and read the data buffer with a read buffer command. The read buffer command must be given within 2.0 msec after the column flag is set or the data will be incorrect. The read buffer command clears the column flag.

Once a card is selected, a new select instruction can be used to change the mode from alpha to binary or vice versa. If the change is from binary to alphanumeric or if a select alphanumeric command is given when the card is already selected in alphanumeric, the column data is re-read into the data buffer. This re-read must occur within 20 microseconds of the column flag to guarantee accuracy.

There are four flags associated with the CR02B card reader. These flags are the column flag, card done flag, end-of-file flag, and the not ready flag. The card done and column flags are connected to the program interrupt and may be individually tested by skip instructions. The column flag indicates column data is in the data register ready to read. The column flag is set by data available and is cleared by the read column register command. The card done flag is set when a card is completely read, and is cleared by either of the select commands. The end-of-file (EOF) level is set by the EOF button on the reader or the hopper empty condition and is cleared by the hopper full condition or the EOF button. The EOF button is an on-off, push-push button. The EOF level may be skip tested. It is used to indicate to the program that the last card of the current deck in the reader has been read. The not ready level indicates that the card reader is in the not ready condition. This may be caused by a read check, feed check, or by the start button not having been depressed. The not ready condition may be tested with a skip instruction.

TABLE 4-7 CARD READER CR02B IOT INSTRUCTIONS

| Mnemonic Symbol | Octal Code | Operation Executed |
|---|---|---|
| CRSF | 706701 | Skip if column flag is set |
| CRSD | 706721 | Skip if card done flag is set |
| DRSD | 706741 | Skip if reader is ready |
| CREF | 706761 | Skip if EOF flag is set |
|  | 706702 | Inclusive OR buffer to the AC and clear column flag |
| CRRB | 706712 | Clear the AC, inclusive OR buffer into the AC, clear column flag |
| CRSA | 706704 | Select alphanumeric mode. A card is selected and the alphanumeric mode is set. If a card is already selected when this command is given, a data reread will occur. |
| CRCD | 706724 | Clear card done flag |
| CRSB | 706744 | Select binary mode. A card is selected and the binary mode is set. |

*DATA FORMATS* - The binary and ALPHA for- data formats are as follows:

*Operator Control and Status* - Console lights, buttons, and switches associated with the CR02B Card Reader are described in table 4-8.

```
DATA FORMATS                                                    (The Hollerith character code is given
                 Unchanged                    BINARY             in the Appendix, page A2-6.)
                ___________
   AC BIT     0 1 2 3 4 5     6  7 8 9 10 11     12 13 14 15 16 17

  CARD ROW                   12 11 0 1  2  3      4  5  6  7  8  9

                                      ALPHA

   AC BIT     0 1 2 3 4 5     6  7 8 9 10 11     12 13 14 15 16 17
                ___________     ____________       _______________
   USAGE       Unchanged         Unchanged          Character Code
```

### Automatic Line Printer Type 647

The Type 647 Automatic Line Printer prints text in lines of up to 120 characters at a maximum rate of 300 lines per minute for the 647D or 600 lines per minute for the 647E. Printing is performed by solenoid-actuated hammers. The typeface is engraved on the surface of the continuously rotating drum. A 64-character set is provided. Models are available with up to 160 columns and a printing rate of 1000 lines per minute.

TABLE 4-8 CARD READER CR02B, CONSOLE LIGHTS, BUTTONS, AND SWITCHES

| Light | Color | Meaning |
|---|---|---|
| NOT READY | white | On whenever the reader is unavailable to the computer. Turned on by STOP button, empty hopper, full stacker, malfunction (read check, feed check, or validity check when the VALIDITY ON switch is activated), or a power-on sequence. Turned off by pressing the START button. |
| READ CHECK | red | Turned on by the failure in the read circuitry. Turned off by pressing the RESET button. |
| FEED CHECK | red | Turned on when a card fails to reach the read station in the prescribed time. Turned off by pressing RESET. Reader motors stop when this light comes on. |
| VALIDITY CHECK | red | Turned on when VALIDITY ON switch is activated and an invalid punch combination is read in the alphanumeric mode. Turned off by pressing RESET. |
| *Button or Switch* | | *Meaning* |
| POWER ON | | Turns power on to reader and control logic. Button lights green when power is on. NOT READY light also comes on. |
| POWER OFF | | Turns power off to reader. |
| START | | Turns off NOT READY light and places reader in the ready condition if the check lights are off. |
| STOP | | Turns on the NOT READY light and places the reader in the not ready condition. If the reader is in operation when this button is pressed, the reader stops when the current card runs out to the stacker. |
| RESET | | Turns off the three red check lights (READ CHECK, FEED CHECK, and VALIDITY CHECK). Does not place the reader in the ready condition; does not turn off the NOT READY light. |
| END OF FILE | | Signals an end-of-file condition to the computer when this button is pressed and the hopper is empty. Has no effect if the hopper is not empty. The button lights white when an end-of-file condition is present. The light is extinguished when cards are placed in the hopper. |
| VALIDITY ON | | When this switch is on, validity check errors stop the reader (see VALIDITY CHECK light above). When off, validity check errors do not stop the reader. Alternately pressing the button turns the switch on and off. When on, the button lights yellow. |

Information is transferred from computer to printer through a printer interface, which contains a core buffer in which a line to be printed is assembled character by character. Each character is represented by a 6-bit binary code. When a print cycle is initiated, the core buffer is scanned each time a row on the drum comes up to the print station. As the characters are printed, the corresponding core buffer positions are cleared so that at the completion of the print cycle the buffer is clear and ready for the next line.

A print cycle is initiated by a command from the program. Depending on the distribution and number of different characters in the line to be printed, a print cycle may take from about 48 to 180 milliseconds, not including vertical spacing of the paper.

Vertical movement of the paper is under control of a punched format tape. Eight program-selectable channels determine the amount of vertical spacing by sensing the punches in the tape. Spacing may be performed at the completion of a print cycle. The paper and tape then move until a hole in the tape is sensed. The table below shows the increments punched on the standard format tape. The user may also create his own formats for which a special punch is available.

| *AC Bits 15 - 17* | *Tape Channel* | *Spacing Increment* |
|---|---|---|
| 0 | 2 | Every line |
| 1 | 3 | Every 2nd line |
| 2 | 4 | Every 3rd line |
| 3 | 5 | Every 6th line |
| 4 | 6 | Every 11th line (1/6th page) |
| 5 | 7 | Every 22nd line (1/3rd page) |
| 6 | 8 | Every 33rd line (1/2 page) |
| 7 | 1 | Top of next form |

Note that spacing is referenced from the top of the form. A space of one line requires 9 milliseconds. Longer skips vary in time, the first taking 9 milliseconds and then 2 to 3 milliseconds for every line thereafter. The spacing increments assume a page format of 66 lines.

*Operating Controls and Indicators* - With the exception of the main power switch and certain test pushbuttons, all of the operating controls are located on two panels. The main panel is at the left on the front of the printer; the auxiliary panel is at the rear on the same side of the machine. The function of line printer controls and indicators is specified in table 4-9.

In addition to the above paper low alert, no paper, and yoke open alarms, an alarm can be generated by a failure in any part of the printer; such a failure automatically takes the printer off-line.

*Programming* - A line to be printed is assembled in the printer buffer character by character from left to right. When the line is complete, a program command initiates the print cycle. When the cycle is finished, the paper may or may not be spaced vertically. Suppressing vertical movement makes underscoring and overbarring possible. When spacing is performed, the printer buffer becomes available approximately 4 milliseconds before the paper comes to a stop. The program may begin assembling the next line during this time.

Three loading instructions (table 4-10) allow the program to transfer one, two, or three characters at a time from the AC to the printer buffer. If more than one character is transferred, the characters in the most significant bits of the AC are transferred before characters in less significant bits.

The buffer loading instructions perform the inclusive OR transfer of the contents of the AC and the current positions of the printer buffer. Thus, the buffer must be clear before a new line is loaded. Clearing is done automatically during the print cycle; an instruction is provided for initializing the interface and clearing the buffer before starting to print.

The capacity of the printer buffer is 120 characters. The program must keep track of the number of characters transferred; if more than 120 are sent, the done flag is not set.

Two flags are associated with the Type 647: done and error. The done flag is set at completion of an IOT-initiated function. The error flag is set when an alarm signal occurs and can be reset only when the alarm condition is removed. The done flag is connected to the program interrupt control. Both the done flag and the error flag may be sensed by skip instructions.

The logical sequence for use of the line printer is as follows:

1. Check the error flag (LSEF).
2. Clear the printing buffer (LPCB)*.
3. Load the printing buffer (and the spacing buffer, if required)*.
4. Print the contents of the printing buffer (since the printing buffer is automatically cleared*, step 2 may be omitted from the sequence after an initial print cycle has been executed).

* The program must wait for the done flag setting before issuing a new command to the line printer.

TABLE 4-9 LINE PRINTER CONTROLS AND INDICATORS.

| Control or Indicator | Function |
|---|---|
| TRACTOR INDEX | Used for aligning the forms with the format tape when new paper is loaded. This pushbutton works only when the printer is off-line. |
| PAPER LOW ALERT | This red indicator lights when the end of the paper is about to pass through the drag devices below the printer yoke. An alarm signal is sent to the computer at the same time. |
| NO PAPER | When the end of the paper has passed out of the forms tractors, this indicator lights red, and an alarm signal is sent to the computer. |
| YOKE OPEN | When the printer yoke is open, this red indicator lights. An interlock prevents all but the TOP OF FORM and TRACTOR INDEX controls from operating. |
| ALARM STATUS | Whenever an alarm signal is generated, this red indicator lights. |
| ON, OFF | These pushbuttons control application of primary power to the functioning parts of the printer. The main power switch must be turned on for these switches to function. The rest of the controls operate only after ON has been pressed. |
| START | Places the printer on-line; it is then ready to receive information and print it. |
| STOP | Takes the printer off-line as soon as the buffer is clear. If there is information in the buffer, the printer remains on-line until after the next clear buffer instruction or the completion of the next print cycle. When the printer goes off-line, an alarm signal is sent to the computer. |
| TEST PRINT | This pushbutton is used for maintenance at the printer and is not used in normal operation. |
| TOP OF FORM | Moves the paper to the top of the next page. This pushbutton works only when the printer is off-line. |

## Incremental Plotter and Control Type 350

A California Computer Products Incremental Recorder can be operated with a Digital Equipment Type 350 Incremental Plotter Control. Characteristics of the available models are provided in table 4-11.

The principles of operation are the same for each of the models. 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 program interrupt and I/O skip facilities. The instructions for this equipment are listed in table 4-12.

Program sequence must assume that the pen location is known at the start of a routine since there is no means of specifying an absolute pen location in an incremental plotter. Pen location can be preset by the manual controls on the recorder. During a subroutine, the computer can track the location of the pen on the paper by

TABLE 4-10 LINE PRINTER TYPE 647E

| Mnemonic Symbol | Octal Code | Operation Executed |
|---|---|---|
| LSDF | 706501 | Skip if the DONE flag is set. |
| LPCB | 706502 | Clear the DONE flag, clear control print buffer, enable DONE interrupt, initiate a clear sequence in the hue printer. Set the DONE flag when the clear sequence is finished. |
| *LPDI | 706504 | Disable DONE flag interrupt. |
| | 706522 | Clear DONE flag. |
| | 706542 | Clear DONE flag. |
| | 706562 | Clear DONE flag. |
| LPL2 | 706526 | Load printing buffer with two characters; clear DONE flag; the contents of AC 6-11 and AC 12-17 are transferred to the printing buffer as 6-bit bytes in that order. The DONE flag will be set when the load sequence is finished. |
| LPPS | 706646 | Print and Space. This instruction accomplishes the combined actions of LPPB and LPLS instructions. The DONE flag is cleared; the contents of AC 15-17 are transferred to the spacing buffer; the contents of the printing buffer are printed; the paper is spaced vertically; the printing and spacing buffers are cleared; the DONE flag is set upon completion. |
| *LPEI | 706664 | The DONE flag interrupt is enabled. |
| LPLD | 706546 | Load the printing buffer with three characters. The DONE flag is cleared; the contents of AC 0-5, 6-11, and 12-17 are transferred as 6-bit bytes into the printing buffer in that order. The DONE flag is set at the completion of the load sequence. |
| LPL1 | 706566 | Load the printing buffer with one character; clear DONE flag; the contents of AC 12-17 are transferred as a 6-bit byte into the printing buffer in that order. The DONE flag is set at the completion of the load sequence. |
| LPEF | 706601 | Skip if the ERROR flag is set. |
| LPCF | 706602 | Clear DONE flag. |
| | 706622 | Clear DONE flag. |
| | 706642 | Clear DONE flag. |
| | 706662 | Clear DONE flag. |

* Refer to footnote at the end of this table

TABLE 4-10 LINE PRINTER TYPE 647E (continued)

| Mnemonic Symbol | Octal Code | Operation Executed |
|---|---|---|
| LPPB | 706606 | Select printer and initiate printing. The DONE flag is cleared; the contents of the printing buffer are printed; the printing buffer is cleared; the DONE flag is set when the printing sequence is completed. |
| LPLS | 706626 | Load spacing buffer and space; the DONE flag is cleared; the contents of AC 15-17 are transferred into the spacing buffer; the paper is spaced vertically according to the format selected; the spacing buffer is cleared; the DONE flag is set. |

* These instructions have been added to the Line Printer command set to allow enabling and disabling of the interrupt. Since power clear returns the system to the interrupt enabled condition, programs generated for the Type 647B Line Printer (PDP-7), which does not have these instructions, will run correctly.

TABLE 4-11 DIGITAL INCREMENTAL RECORDER CHARACTERISTICS

| CCP Model | Step Size (inches) | Speed (steps/minute) | Paper Width (inches) |
|---|---|---|---|
| 563 | 0.01 or 0.005 | 12,000 or 18,000 | 31 |
| 565 | 0.01 or 0.005 | 18,000 | 12 |

TABLE 4-12 INCREMENTAL PLOTTER AND CONTROL INSTRUCTIONS

| Mnemonic Symbol | Octal Code | Operation Executed |
|---|---|---|
| PLSF | 702401 | Skip if plotter flag is a 1. |
| PLCF | 702402 | Clear plotter flag. |
| PLPU | 702404 | Plotter pen up. Raise pen off of paper. |
| PLPR | 702421 | Plotter pen right. |
| PLDU | 702422 | Plotter drum (paper) upward. |
| PLDD | 702424 | Plotter drum (paper) downward. |
| PLPL | 702441 | Plotter pen left. |
| PLUD | 702442 | Plotter drum (paper) upward. (same as 702422) |
| PLPD | 702444 | Plotter pen down. Lower pen on to paper. |

counting the instructions that increment the positions of the pen and the drum.

## Oscilloscope Display Type 34H

Type 34H is a two-axis digital-to-analog converter and an intensifying circuit, which provides the deflection and intensify signals needed to plot data on an oscilloscope. Coordinate data is loaded into an X buffer (XB) or a Y buffer (YB) from bits 8 through 17 of the accumulator. The binary data in these buffers is converted to a -10 to 0 volt analog deflection signal. The 30-volt, 10-microsecond intensify signal is connected to the grid of the oscilloscope CRT. Points can be plotted at approximately a 30-kilocycle rate. The IOT instructions for this display are identical to those for the Precision CRT Display Type 30D, described under the following heading, with the exception of the DLB command. The 34H has a 2-bit brightness register (BR), the contents of which specify the degree of brightness for the point being displayed. The following indicates the intensity scale:

| *BR Contents* | *Intensity Level* |
|---|---|
| 0 | no display |
| 1 | dimmest |
| 2 | average |
| 3 | brightest |

The instruction 700704 loads the BR with the contents of AC bits 16 and 17.

## Precision CRT Display Type 30D

The Type 30D displays points on the face of a cathode ray tube. Each point is located by its X- and Y-coordinates in a square array whose origin is in the lower left corner of the CRT screen. The array contains 1024 points on a side and measures 9-1/4 by 9-1/4 inches square.

The X- and Y-coordinates each have a 10-bit buffer which is loaded from bits 8-17 of the AC. In addition, there is a 3-bit brightness register (BR) which is loaded from bits 15-17 of the AC. The content of this buffer specifies the brightness of the point being displayed as designated on the following scale. The five brightest intensities are easily visible in a normally lighted room; the dimmest can be seen in a darkened room.

The X- and Y-coordinate buffers (SB and YB) are loaded separately. Each may be loaded without intensifying the CRT. The usual procedure is to load one buffer, then load the second buffer and select in one instruction. The Type 30D requires 50 microseconds to display a point. No flag is associated with this operation. The IOT instructions for the Type 30D display are listed in table 4-13.

| *BR Contents* | *Intensity Level* |
|---|---|
| 3 | brightest |
| 2 | |
| 1 | |
| 0 | average |
| 7 | |
| 6 | |
| 5 | |
| 4 | dimmest |

## Photomultiplier Light Pen Type 370

The high-speed light pen is a photosensitive device which senses displayed points on the face of the CRT. The Type 370 uses a fiber optic light pipe and photomultiplier system, which gives the pen a response time approximately five times faster than that of a photodiode. If the pen is held in front of a point displayed on the face of the CRT, it transmits a signal which sets the display flag to 1. The Type 370 is equipped with a mechanical shutter which prevents the sensing of unwanted information while positioning the pen. Variable fields of view are obtained by means of a series of interchangeable tips with fixed apertures. The IOT instructions for the light pen are listed with the display option instructions in table 4-13.

## Analog-to-Digital Converter and Multiplexer Type AF01B

The General Purpose Multiplexer A/D Converter Type AF01B is used with PDP-9 computer to multiplex up to 64 analog signals and to convert the signals to binary numbers. It replaces the older Type 138E/139E system.

*A/D Converter* - The A/D converter is a general purpose successive-approximation type with the following characteristics:

| | |
|---|---|
| *Accuracy:* | See table 4-16 (includes all linearity and temperature errors). |
| *Conversion Time:* | See table 4-16. |
| *Aperture:* | Same as Converstion time without AH02 sample and hold option. |

| | |
|---|---|
| *Converter Recovery Time:* | None. |
| *Analog Input: Voltage Range* | 0 to -10V standard (see table for amplifier or sample and hold options). |
| *Loading:* | ± 1 microamper and 125 picofarads for standard input. |

The word-length switch selects the A/D Converter Characteristics listed in table 4-14.

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 nanoseconds. The Type A400 may also be used to scale the signal input to accept ±10V, ±5V, or 0 to 10V. The Type A200 amplifier (AH03 option) may be substituted for the Type A400 to accomplish the same signal scaling without reducing the effective aperture.

Both the AH02 and AH03 options may be used to obtain high input impedance and small aperture. All power is contained in the Type AF01 for the amplifier and/or sample and hold options.

Five convenience switches are mounted on the control indicator panel: a power switch to control the AC power to the Type AF01 System, an ADC pushbutton switch to initiate a conversion manually; a CLR pushbutton switch to set the multiplexer address to channel 0 manually; an index pushbutton to increment the multiplexer address manually; and a rotary word-length switch to select the word length, conversion accuracy, and conversion time.

The control indicator panel also contains twelve indicators to display the contents of the ADC buffer, six indicators to display the current multiplexer address, and a power off-on indicator.

*Multiplexer Switches* - The multiplexer can include from 1 to 16 Type A121 Switch Modules. Each module contains four single-pole, high speed, insulated gate FET switches with appropriate gating. The Type A121 Switches are arranged as a 64-channel group of series-switch

TABLE 4-13 OSCILLOSCOPE AND PRECISION DISPLAY INSTRUCTIONS

| Mnemonic Symbol | Octal Code | Operation Executed |
|---|---|---|
| DXL | 700506 | Load the X-coordinate buffer from $AC_{8-17}$. $AC_{8-17}$ = XB. |
| DXS | 700546 | Load the X-coordinate buffer and display the point specified by the XB and YB. |
| DYL | 700606 | Load the Y-coordinate buffer from $AC_{8-17}$. $AC_{8-17}$ = YB. |
| DYS | 700646 | Load the Y-coordinate buffer and display the point specified by the XB and YB. |
| DXC | 700502 | Clear the X-coordinate buffer. |
| DYC | 700602 | Clear the Y-coordinate buffer. |
| DLB | 700706 | Load the brightness register from bits 15-17 of the AC. Note: This instruction clears the display flag associated with the light pen. |
| DSF | 700501 | Skip if display (light pen) flag is a 1. |
| DCF | 700702 | Clear display (light pen) flag. |

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 AF01B System.

*Specifications* (Measured at input connector)

| | |
|---|---|
| Input signal (max) | ± 10 V |
| Input current | 1.0 ma |
| "On" offset voltage | 0 |
| "On" resistance (max) | 450 ohms |
| Turn-on delay | 150 nsec |
| "Off" leakage (max) | 10 na |
| Turn-off delay | 250 nsec |
| Settling time to 1-LSB (source $Z \leq 1$ ohm) | ≤2 microseconds |

*Operation* - The Type AF01B System may be operated in either the random or sequential address modes. In the random address mode, the control routes the analog signal from any selected channel to the A/D converter input. In the sequential address mode, the multiplexer control advances its channel address by one each time an index command is received. After indexing through the maximum number of channels is 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. Two separate A/D Convert I/O Transfer Commands may also initiate one or more conversions on a currently selected channel. Conversion times listed in the word-length table are increased by 2 microseconds when multiplexer channels are switched to allow for settling times of the analog signal at the multiplexer output. (This time is increased by 5 microseconds when AH03 is used.) Each successive conversion on a selected channel requires only the time shown in table 4-14.

**Digital-to-Analog Converter Type AA01A**

This general purpose digital-to-analog converter converts 12-bit binary computer output numbers to analog voltages. The basic option consists of three channels, each containing a 12-bit digital buffer register and a digital-to-analog converter. Digital input to all three buffer registers is provided in common by one 12-bit input channel which interfaces to the PDP-9/L I/O bus. Appropriate precision voltage reference supplies are provided for the converters.

One IOT instruction simultaneously selects a channel and transfers a binary number into the selected buffer register. Each converter operates continuously on the contents of its associated buffer register to produce an analog output voltage. The analog output voltage of a standard converter is from ground to -9.9976 volts (other voltage ranges are available).

All inputs to the converter are assumed to be 12 bits in length with negative numbers represented in 2s complement notation. An input of $4000_8$ yields an analog output of ground potential; an input of $0000_8$ yields an output of -5 volts; and an input of $1777_8$ yields an output of -10 volts minus the analog value of the least significant bit of the input. Output accuracy is ± 0.0125% of full scale; resolution

TABLE 4-14 A/D CONVERTER CHARACTERISTICS

| Word Length (No. of bits) | Max Switching Point Error* (±) | Conversion Time (microseconds) |
|---|---|---|
| 6 | 1.6% | 9.0 |
| 7 | 0.8% | 10.4 |
| 8 | 0.4% | 12.0 |
| 9 | 0.2% | 13.5 |
| 10 | 0.1% | 18.0 |
| 11 | 0.05% | 25.0 |
| 12 | 0.025% | 35.0 |

* ±1/2 LSB for quantizing error

is 0.025% of full scale value. Response time, measured directly at the converter output, is 3 microseconds for a full-scale step change to 1 least significant bit accuracy. Maximum buffer register loading rate is 2 MHz.

Type AA01A Converters can be specified in a variety of basic configuration: with from one to three channels, with or without output operational amplifiers, and with internally or externally supplied reference voltages. Converters can be also supplied with two buffer registers per each channel. This provision permits program control to load the outer buffer register of each channel with an appropriate binary number and then effect a simultaneous transfer of these contents into the inner buffer registers for simultaneous conversion to a summed analog voltage.

A typical instruction for the AA01A is:

|  |  |
|---|---|
| *Mnemonic:* | DAL1 |
| *Octal Code:* | 705501 |
| *Function:* | Load digital-to-analog converter 1. The contents of the AC are entered in the digital buffer register of channel 1. |

The variety of possible configurations makes it necessary that the user, or interface designer, define the appropriate instructions and append them to the symbol table of the Symbolic Assembler of the BASIC Software system.

**Multi-station Teletype Control Type LT09A**

Addition of the LT09A option to the PDP-9/L expands the machine's Teletype facility to accommodate several Teletype units (KSRs and ASRs may be used interchangeably). Operation of the LT09A facility is in full-duplex mode. Each Teletype line added contains logical elements which are functionally identical to those of the control for the standard unit. Instructions and programming considerations are, therefore, similar to those of the standard unit. The following device selection codes have been assigned for four lines of LT09A equipment.

| *Line* | *Teleprinter* | *Keyboard* |
|---|---|---|
| 1 | 40 | 41 |
| 2 | 42 | 43 |
| 3 | 44 | 45 |
| 4 | 46 | 47 |

Instruction mnemonics for the Teletype units must be defined by the user and appended to the PDP-9/L BASIC Symbolic Assembler. Typi-

TABLE 4-15 AF01B A/D CONVERTER AND MULTIPLEXER IOT INSTRUCTIONS

| **Mnemonic Symbol** | **Octal Code** | **Operation Executed** |
|---|---|---|
| ADSF | 701301 | Skip if converter flag is set. This flag is connected to the program interrupt. |
| ADSC | 701304 | Select and convert. The converter flag is cleared and a conversion of an incoming voltage is initiated. When the conversion is complete, the converter flag is set. |
| ADRB | 701312 | Read converter buffer. Places the content of the buffer in the AC, left adjusted. The remaining AC bits are cleared. The converter flag is cleared. |
| ADSM | 701103 | Select MX channel. The content of $AC_{12-17}$ are placed in the MAR. |
| ADIM | 701201 | Increment channel address. The content of the MAR is incremented by 1. Channel 0 follows channel $77_8$. |
| ADRM | 701212 | Read MAR into $AC_{12-17}$. |

cally, the mnemonics are derived by suffixing 'LT' and the line number to the mnemonics for the standard unit. For example, the instruction KSF (skip on keyboard flag) could be represented by KSFLT2 for an instruction to "skip on keyboard flag of line 2".

### Relay Buffer Type DR09A

The Type DR09A is a computer output device that allows data in the computer to control external electrical equipment through relays. The relay buffer consists of an 18-bit flip-flop register, and 18-bit relay register, filters to reduce noise due to contact bounce, and a patchboard. Under program control the flip-flop register can be set to correspond to the content of the accumulator and can be cleared. The commands for the relay buffer are listed in table 4-16.

### Interprocessor Buffers DB99A and DB98A

*Overall Description* - The DB99A and DB98A are bidirectional interprocessor data buffers which operate through the data channel facilities or with programmed data transfers. The DB99A will buffer two PDP-9/Ls together, and the DB98A will buffer one PDP-9/L and one PDP-8. Data may be transferred through the accumulator or data channel, or both simultaneously in full duplex operation. Accumulator word transfer rates as high as 100 kc and data channel rates as high as 110 kc may be achieved.

*General Performance* - There are two basic paths of data flow in the DB99, and DB98. One path is from accumulator to accumulator using programmed data transfers and the other path is memory to memory utilizing the data channel facilities of both machines. A single register at each interface is used for both types of data transfer and the inputs to these registers are multiplexed.

Programmed Transfers - Programmed transfers of data occur between the accumulators of the computers involved. This mode of data transfer is used primarily for transmission of control parameters while the data channel system is simultaneously handling a full duplex (bidirectional) data transfer; however, the programmed data transfer system can also be used as the primary data transmission method. The supervisory overhead of operating this system will be significantly higher, however, and the transfer rates will be slightly lower.

Data Channel Transfers - Data channel transfers can occur between computers in simultaneous full duplex fashion. The standard channel facilities are utilized (three cycle Data Break on the PDP-8; DCH on the PDP-9/L). The access of data to be transmitted and the storage of data received is supervised by the hardware. Each interface has two data channels, one for data transmission and one for data reception. Each data channel is assigned two memory locations to contain the word count and address registers for the channel. The transmit channel is assigned locations 22 and 23 and the receive channel is assigned locations 24 and 25. (Refer to table 4-17 for IOT instructions.)

*Bit Correspondence* - The bit correspondence for data words in the DB98A is given below:

| PDP-9 Bit | 0 1 2 | 3 4 5 | 6 7 8 | 9 10 11 | 12 13 14 | 15 16 17 |
|---|---|---|---|---|---|---|
| PDP-8 Bit | | | 0 1 2 | 3 4 5 | 6 7 8 | 9 10 11 |

### Command Status Register Configuration

| *PDP-8 Bit* | *PDP-9 Bit* | *Meaning* |
|---|---|---|
| 11 | 17 | Transmit flag. |
| 10 | 16 | Receive flag. |
| 9 | 15 | Transmit word count overflow flag. |

TABLE 4-16 RELAY BUFFER COMMANDS

| Mnemonic Symbol | Octal Code | Operation Executed |
|---|---|---|
| ORC | 702101 | Clear output relay buffer flip-flop register. |
| ORS | 702104 | Set output relay buffer flip-flop register to correspond with the contents of the accumulator. |

TABLE 4-17 INTERPROCESSOR BUFFERS DB99A AND DB98A IOT INSTRUCTIONS

| Mnemonic Symbol | Octal Code | Operation Executed |
|---|---|---|
| PDP-9/L IOT Instructions for DB99A | | |
| PBNF | 702201 | Skip if no interrupting flag of this device is set. |
| - | 702202 | Inclusive OR command status register into the AC. |
| PBRS | 702212 | Read command status register into the AC. |
| PBXS | 702204 | Exclusive OR the contents of the AC into the command status register. |
| PBNB | 702221 | Skip if data register not busy. |
| - | 702222 | Clear data register if data register not busy. |
| - | 702224 | OR the AC to data register if data register not busy and set the transmit flag and not busy level and clear the receive flag. |
| PBTF | 702241 | Skip if transmit flag is set. |
| - | 702242 | Inclusive OR data register into the AC, set receive flag and clear transmit flag if the transmit flag is set. |
| PBRD | 702252 | Read data register into the AC, set receive flag and clear transmit flag if the transmit flag is set. |
| Useful PDP-9/L Microinstructions | | |
| PBTL | 702227 | Skip and load data register from the AC, and set transmit flag, and receive flag, if all data register not busy. |
| PBRL | 702253 | Skip if transmit flag is set and read data register into the AC and set receive flag and clear transmit flag, all if transmit flag is set. |
| PBLD | 702226 | Clear and load data register from the AC if the not busy level is set. |
| PDP-8 IOT Instructions for DB98A | | |
| PBNF | 6601 | Skip if no interrupting flag of this device is set. |
| PBRS | 6602 | OR command status register into the AC. |
| PBXS | *6604 | Exclusive OR the contents of the AC (into the command status register). |
| PBNB | 6611 | Skip if data register not busy. |

*See footnote at end of Table.

TABLE 4-17 INTERPROCESSOR BUFFERS DB99A AND DB98A IOT INSTRUCTIONS (Con't)

| Mnemonic Symbol | Octal Code | Operation Executed |
|---|---|---|
| PDP-8 IOT Instructions for DB98A (Con't) | | |
| PBTF | 6612 | Skip if transmit flag is set. |
| PBLD | 6615 | Load data register from the AC set transmit flag, and clear receive flag. All if data register not busy. |
| PBRD | 6616 | Skip if transmit flag is set, read data register into the AC set receive flag and clear transmit flag. |

*If the AC=0 when this command is given the not busy flag will be set to the not busy condition, the CS register will be unchanged.

| *PDP-8 Bit* | *PDP-9 Bit* | *Meaning* |
|---|---|---|
| 8 | 14 | Receive word count overflow flag. |
| 7 | 13 | Enable not busy flag interrupt |
| 6 | 12 | Enable transmit and receive flag interrupts. |
| 5 | 11 | Enable data channel overflow interrupts. |
| 4 | 10 | Enable receive data channel. |
| 3 | 9 | Select transmit data channel. |
| Not in PDP-8 CS register | 8 | Not busy flag. This flag may be manipulated by the XOR to CS instruction, but it may not be read into the accumulator. Power clear returns this flag to the one (not busy) state. |
| 2 | Not in the PDP-9/L CS register | Address extension bit 3. |
| 1 | Not in the PDP-9/L CS register | Address extension bit 2. |
| 0 | Not in the PDP-9/L CS register | Address extension bit 1. |

In the PDP-8 an XOR to CS instruction with the AC set to 0 will cause the not busy flag to be set to the not busy state. The CS register will remain unchanged. Power clear also sets the not busy state.

**PDP-9/L to PDP-7 Interprocessor Buffer Type DB97A**

Controls and buffers the flow of information between one PDP-9/L and one PDP-7, using the program controlled transfer facility of each processor.

**Data Communications System Type DP09A (DP01B)**

*General* - The Bit synchronous Data Communications System Type DP09A provides interface facilities between a PDP-9/L and a bit serial communications device such as a Type 201 or 301 Data Set (Bell System). Characters are transferred between the PDP-9/L accumulator and the DP09A under program control. The DP09A serializes the characters for transmission, and assembles the serial stream into characters for reception. Operation is full duplex. Both the receive and transmit sections are double-buffered to permit one full character transmission time for loading or reading the DP09A.

A character may be 6, 7, 8, or 9-bits long. Character length is determined by a series of jumpers on a 50-pin cannon connector; an appropriate connector for each character length is provided.

*Idle/Active* - The DP09A is in the idle state until made active either by the PDP-9/L transmitting a sync character to the DP09A or the DP09A receiving a sync character from the Data Set.

The sync characters are:

| *Character Length (N)* | *Sync Character* |
|---|---|
| 6 | 010 110 |
| 7 | 0 010 110 |
| 8 | 10 010 110 |
| 9 | X10 010 110 |
| | (X not used in sync character determination) |

The DP09A will return to the inactive (idle) state if -

1. While transmitting, idle mode is disabled and no character has been transferred to the DP09A from the computer within Nt microseconds of the transmit flag being set. (N is the number of bits/character, t is the time to transmit a bit on the line. Nt is therefore the time to transmit a full character on the communications line.)

2. While receiving, the clear receive active (CRA) command is issued, or if no character is received from the communication line for 1.5 bit times.

Idle mode is a feature of the DP09A which permits retaining the communications system in an active (and synchronous) state when no new characters are available. When idle mode is enabled, the last character continues to be transmitted until such time that a new character is ready. The transmit flag continues to be raised at the end of each character transmission.

Idle mode is enabled by the SIM instruction and disabled by the CIM instruction.

*Flags* - The DP09A communicates with the computer through a series of flags. Any one of the flags can cause a program interrupt or API break (if present) to location 62 at priority level two.

| | |
|---|---|
| TRANSMIT FLAG | *Set* when the DP09A is ready to receive a character from the computer for transmission.<br>*Tested* by an STF instruction, skip if flag *not* set.<br>*Cleared* by a CTF instruction |
| RECEIVE FLAG | *Set* when the DP09A is ready to transfer a character to the computer.<br>*Tested* by an SRF instruction, skip if flag *not* set.<br>*Cleared* by an RRB instruction |
| RECEIVE END FLAG | *Set* when no bit is received from the sending device within 1.5*t* (*t* is the normal interbit spacing, the reciprocal of the baud rate). |
| RING FLAG | *Set* when a remote communications device calls up the DP09A and is ready to transmit. Only causes an interrupt if Ring Enable is set (see instruction list).<br>*Tested* by an SRI instruction, skip if flag *not* set.<br>*Cleared* by an CRF instruction |
| DATA SET READY FLAG<br>(This flag can not cause an interrupt) | *Set* when the local Data Set is ready for operation.<br>*Tested* by an SSR instruction, skip if flag *is* set.<br>*Cleared* only when the Data Set becomes unavailable. |

*IOT Commands* - Following are the IOT commands for the Bit Synchronous Data Communications System. Types DP01B and DP09A are listed in table 4-18. (If bit 14 of any of these commands is a 1, the AC will be cleared at event time 1 of the IOT.)

TABLE 4-18 BIT SYNCHRONOUS DATA COMMUNICATIONS SYSTEM TYPES DP01B AND DP09A IOT COMMANDS

| Mnemonic Symbol | Octal DP01B | Octal DP09A | Command |
|---|---|---|---|
| STF | 704501 | 702521 | Skip on Transmit Flag<br><br>This command causes the program flow to skip the next instruction if the transmit flag is NOT set. |
| TAC | 704201 | 702501 | Transmit a Character<br><br>This command transfers the contents of the accumulator (6, 7, 8, or 9-bits right justified) into the transmit character buffer. If the transmit logic is not active, the character will only be transmitted if it is a sync character. |
| CTF | 704202 | 702502 | Clear Transmit Flag<br><br>This command clears the transmit flag. It also causes the program flow to skip the next instruction if the transmit interface is active. Thus, the microprogrammed instruction TAC CTF (704203) loads the next character, clears the transmit flag, and tests the DP09A to be sure that transmit is still active. |
| CIM | 704204 | 702504 | Clear Idle Mode<br><br>This command disables the idle mode. |
| SIM | 704504 | 702524 | Set Idle Mode<br><br>This command enables the idle mode. |
| SRF | 704501 | 702621 | Skip on Receive Flag<br><br>This command causes the program flow to skip the next instruction if the Receive Flag is NOT set. |
| RRB | 704502 | 702522 | Read Receive Buffer<br><br>This command causes the contents of the receive buffer (6, 7, 8, or 9-bits right end justified) to be read into the accumulator. It also clears the Receive flag. |
| SEF | 704601 | 702541 | Skip on Receive End Flag<br><br>This command causes the program flow to skip the next instruction if the "Receive End Flag" is NOT set. |
| CEF | 704602 | 702542 | Clear End Flag<br><br>This command clears the Receive End Flag. |

TABLE 4-18 BIT SYNCHRONOUS DATA COMMUNICATIONS SYSTEM TYPES DP01B AND DP09A IOT COMMANDS (Con't)

| Mnemonic Symbol | Octal DP01B | Octal DP09A | Command |
|---|---|---|---|
| SRE | 704604 | 702544 | Set Ring Enable<br><br>This command causes the Ring Enable gate to be turned on. When the Ring Enable gate is turned on, the Ring flag can cause a program interrupt. |
| CRE | 705004 | 702604 | Clear Ring Enable<br><br>This command causes the Ring Enable gate to be turned off. |
| SRI | 704701 | 702561 | Skip on Ring Indicator<br><br>This command causes the program flow to skip the next instruction if the Ring flag is NOT set. |
| CRF | 704702 | 702562 | Clear Ring Flag<br><br>This command clears the Ring flag. |
| STR | 704704 | 702564 | Set Terminal Ready<br><br>This command causes the Terminal Ready gate to be turned on. This gate indicates to the communication media that the equipment is ready to receive data from the serial line. |
| CTR | 705002 | 702602 | Clear Terminal Ready<br><br>This command turns the Terminal Ready gate off. |
| SSR | 705001 | 702601 | Skip on Data Set Ready<br><br>This command causes the program flow to skip the next instruction if the communication facility is in a ready condition. |
| CRA | 705402 | 702622 | Clear Receive Active<br><br>This command takes the interface (637) out of the receive active state. No further characters will be received until a sync character is detected. |

# CHAPTER 5
# AUXILIARY STORAGE SYSTEMS

## GENERAL

The PDP-9/L presently includes in its line of standard peripherals the following auxiliary storage systems: DECtape systems, industry standard magnetic tape systems, and a high-speed disk.

## DECTAPE SYSTEM

The DECtape system, a standard option for the PDP-9/L, serves as a magnetic tape data storage facility. The system, consisting of TU55 DECtape transports and TC02 DECtape controls, stores information at fixed positions on magnetic tape as in magnetic disk or drum storage devices, rather than at unknown or variable positions as is the case in conventional magnetic tape systems. This feature allows replacement of blocks of data on tape in an ordered fashion without disturbing other previously recorded information. In particular, during the writing of information on tape, the system reads format (mark) and timing information from the tape and uses this information to determine the exact position at which to record the information to be written. Similarly, in reading, the same mark and timing information is used to locate data to be played back from the tape.

This system has a number of features to improve its reliability and make it exceptionally useful for program updating and program editing applications. These features are: phase or polarity sensed recording on redundant tracks, bidirectional reading and writing, and a simple mechanical mechanism utilizing aerodynamically lubricated tape guiding (the tape floats on air and does not touch any metal surfaces).

### DECtape Format

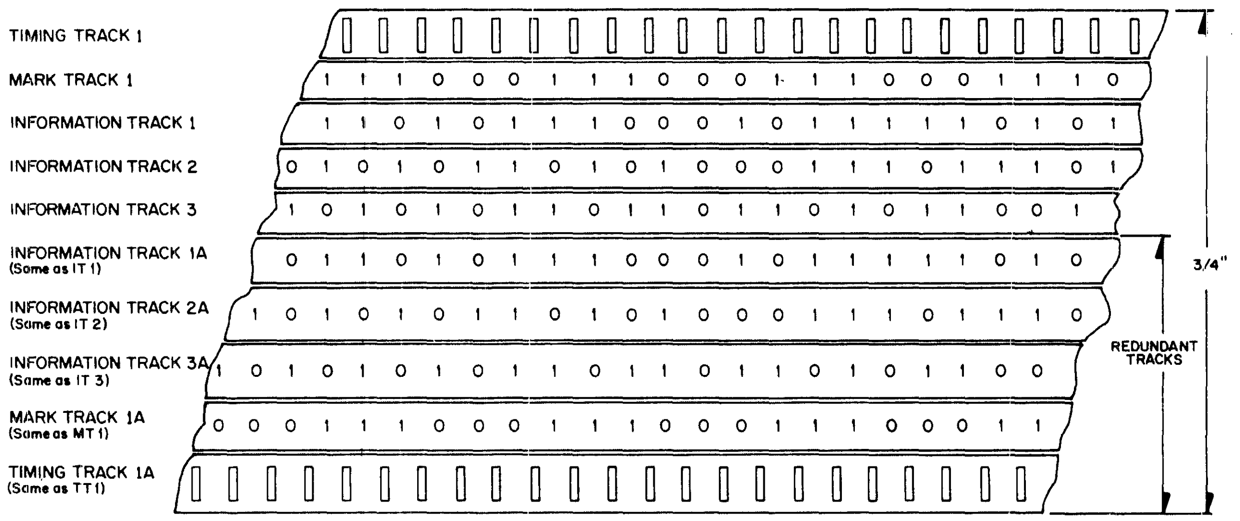
DECtape utilizes a 10-track read/write head. Tracks are arranged in five nonadjacent redundant channels: a timing channel, a mark channel, and three information channels (figure 5-1). Redundant recording of each character bit on nonadjacent tracks materially reduces bit drop out and minimizes the effect of skew. Series connection of corresponding track heads within a channel and the use of Manchester phase recording techniques, rather than amplitude sensing techniques, virtually eliminate dropouts.

The timing and mark channels control the timing of operations within the control unit and establish the format of data contained on the information channels. The timing and mark channels are recorded prior to all normal data reading and writing on the information channels. The timing of operations performed by the tape drive and some control functions are determined by the information on the timing channel. Therefore, wide variations in the speed of tape motion do not affect system performance. Information read from the mark channel is used during reading and writing data, to indicate the beginning and end of data blocks and to determine the functions performed by the system in each control mode.

During normal data reading, the control assembles 18-bit computer length words from six successive lines read from the information channels of the tape. During normal data writing, the control disassembles 18-bit words and distributes the bits so they are recorded on six successive lines on the information channels. A mark channel error check circuit assures that one of the permissible marks is read in every six lines on the tape.

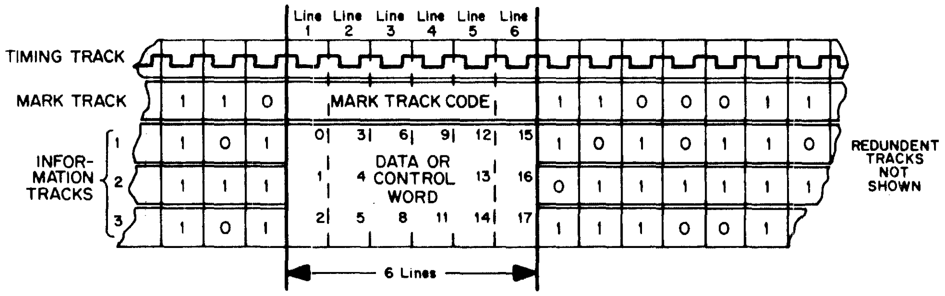
A tape contains a series of data blocks that can be of any length which is an even number of 18-bit words. Block length is determined by information on the mark channel. Usually a uniform block length ($256_{10}$ for the PDP-9/L) is established over the entire length of a reel of tape by a program which writes mark and timing information at specific locations. The ability to write variable-length blocks is useful for certain data formats. For example, small blocks containing index or tag information can be alternated with large blocks of data. The maximum number of blocks addressable is 4096.

Between the blocks of data are areas called interblock zones, consisting of control words. These words are used for cueing the TC02 control, and for block by block parity checking.

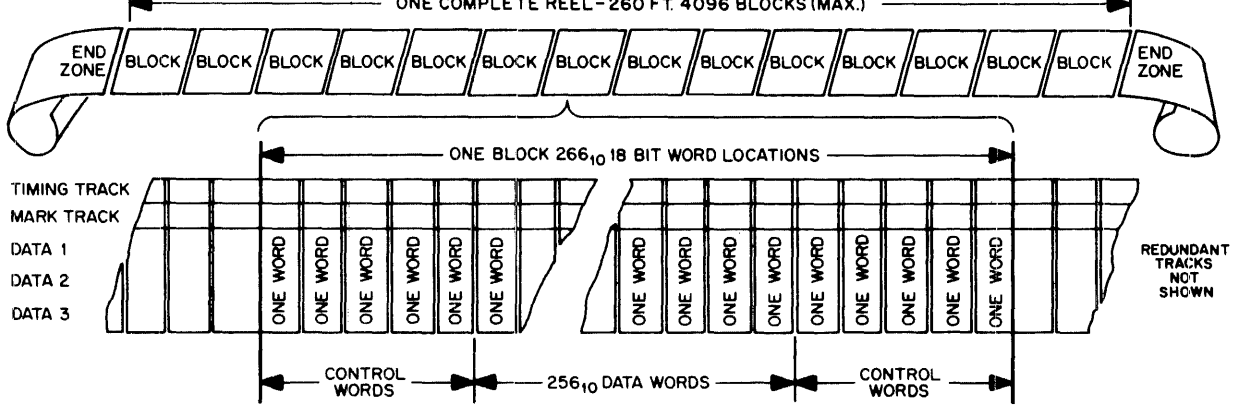


TIMING TRACK 1
MARK TRACK 1
INFORMATION TRACK 1
INFORMATION TRACK 2
INFORMATION TRACK 3
INFORMATION TRACK 1A (Same as IT 1)
INFORMATION TRACK 2A (Same as IT 2)
INFORMATION TRACK 3A (Same as IT 3)
MARK TRACK 1A (Same as MT 1)
TIMING TRACK 1A (Same as TT 1)
3/4"
REDUNDANT TRACKS

Track Allocation Showing Redundantly Paired Tracks

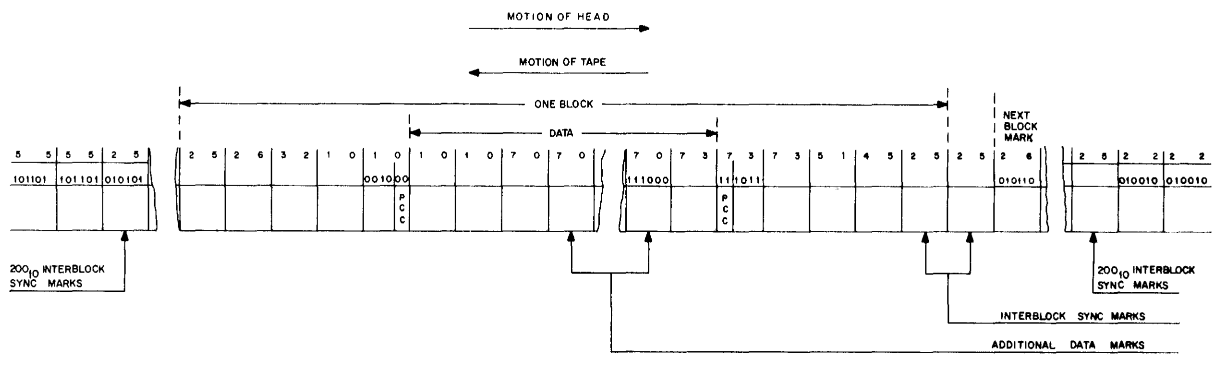


Basic Six Line Tape Unit



Control and Data Word Assignments

Figure 5-1. DECtape Format (Sheet 1)



**DECtape Mark Track Format**

Figure 5-1. DECtape Format (Sheet 2)

Block numbers normally occur on tape in sequence from 0 to N-1, where N is the number of blocks. The total length of tape is equivalent to 884,736 data lines per tape which can be divided into any number of blocks up to 4096 by prerecording of the mark track. However, $576_{10}$ blocks of $256_{10}$ words are considered to be standard format for a PDP-9/L DECtape.

## DECtape Transport Type TU55

The TU55 is a bidirectional magnetic-tape transport consisting of a read/write head for recording and playback of information on five channels of the tape. Connections from the read/write head are made directly to the TC02 control which contains the read and write amplifiers.

The logic circuits of the TU55 control tape movement in either direction over the read/write head. Tape drive motor control is exercised completely through the use of solid state switching circuits to provide fast reliable operation. These switching circuits contain silicon controlled rectifiers which are controlled by normal DEC diode and transistor logic circuits. These circuits control the torque of the two motors which transport the tape across the head according to the established function of the device, i.e., go, stop, forward, or reverse. In normal tape movement, full torque is applied to the forward or leading motor and a reduced torque is applied to the reverse or trailing motor to keep proper tension on the tape. Since tape motion is bidirectional, each motor serves as either the leading or trailing drive for the tape, depending upon the forward or reverse control status of the TU55. A positive stop is achieved by an electromagnetic brake mounted on each motor shaft. When a stop command is given, the trailing motor brake latches to stop tape motion. Enough torque is then applied to the leading motor to take up slack in the tape.

Tape movement can be controlled by commands originating in the computer and applied to the TU55 through the TC02 DECtape Control, or can be controlled by commands generated by manual operation of rocker switches on the front panel of the transport. Manual control is used to mount new reels of tape on the transport, or as a quick maintenance check for proper operation of the control logic in moving the tape.

## DECtape Control Type TC02

A maximum of eight TU55 DECtape transports may be connected to one TC02. Of the four data channels available, DECtape is assigned to channel 0 (i.e., core memory locations 30 and 31).

C(30) = Word Count (in 2s complement form) - WC
C(31) = Current Address Register - CA

Data transfers may take place to or from only one transport at any given time at a rate of one word every 200 microseconds (1 block of $256_{10}$ words every 53 msec), after the desired block has been found (see DECtape summary for complete timing information).

Since the CA is incremented before the data transfer (except in search where the CA is not incremented), the 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 2s complement of the desired number of data transfer. In this way, the word transfer which causes the word count overflow is the last transfer to take place.

The number of IOTs required for the TC02 is minimized by the scheme of transferring all necessary DECtape control data (i.e., unit, function, mode, direction, etc.) from the AC to the control using one set of IOTs (refer to table 5-1). Similarly all status information (i.e., all above information plus status bits, error flags, etc.) can be read into the AC from the control unit via a second set of IOTs.

A 6-bit parity check character is computed (the XOR of the reverse parity check character and every 6 bits of every data word) and recorded by the DECtape control for every block of data recorded during the WRITE DATA function. It is used for automatic parity checking during the READ DATA function.

## Command and Status Bit Configuration

*Status Register A (Command Bits)*

| *Bit* | | *Assignment* |
|---|---|---|
| 0, 1, 2 | Unit selection 1 through 8 (1, 2, 3,...........7, 0) | |
| 3 | Motion: | Forward–0; Reverse–1 |
| 4 | Motion: | Stop–0; GO–1 |
| 5 | Transfer Mode: | Normal (NM)–0; Continuous Mode (CM)–1 |
| 6, 7, 8 | Function | 0 – MOVE<br>1 – SEARCH<br>2 – READ DATA<br>3 – READ ALL<br>4 – WRITE DATA<br>5 – WRITE ALL<br>6 – WRITE TIMING and MARK TRACK<br>7 – SELECT ERROR |

TABLE 5-1 TC02 CONTROL IOT INSTRUCTIONS

| Mnemonic | Octal Code | Description |
|---|---|---|
| DTCA | 707541 | Clear status register A. The DECtape control and error flags are undisturbed (DTF and EF). |
| DTRA | 707552 | Read status register A. The AC is cleared and the content of status register A is ORed into the accumulator. |
| DTXA | 707544 | XOR status register A. The exclusive OR of the content of bits 0 through 9 of the accumulator and status A is loaded into status register A, and bits 10 and 11 of the accumulator are sampled to control clearing of the error and DECtape flags, respectively. Any time this command is given with AC bits 0-4 set to 1, the select delay of 120 msec will be incurred. |
| DTLA | 707545 | Load status register A. Combines action of DTCA and DTXA to load AC0-9 into status register A. Bits 10 and 11 control clearing of error and DECtape flags, respectively. |
| DTEF | 707561 | Skip on error flag. The state of the error flag (EF) is sampled. If it is set to 1 the content of the PC is incremented by one to skip the next sequential instruction. |
| DTRB | 707572 | Read status B. The AC is cleared and the content of status B is ORed into the accumulator. |
| DTDF | 707601 | Skip on DECtape flag. The state of the DECtape flag (DTF) is sampled. If it is set to a 1, the content of the PC is incremented by one to skip the next sequential instruction. |

| | |
|---|---|
| 9 | Disable (0); Enable (1) DTF and EF to cause Program Int. |
| 10 | Error Flag – Clear (0); Undisturbed (1) |
| 11 | DECtape Flag – Clear (0); Undisturbed (1) |

*Status Register B (Flag and Error Status Bits)*

| *Bit* | *Assignment* |
|---|---|
| 0 | Error Flag |
| 1 | Mark Track Error |
| 2 | End of Tape |
| 3 | Select Error |
| 4 | Parity Error |
| 5 | Timing Error |
| 6-10 | Unused |
| 11 | DECtape Flag |

All 10 command bits (0-9) of status register A may be sensed, set or changed via IOTs. Bits 10 and 11 of the AC are not retained by status A, but enable or disable the clearing of the DECtape and ERROR flags. The bits in status register B may be sensed and cleared by IOTs. To issue a DECtape command, the command bits 0-9 of status register A are set as desired by bits 0-9 of the AC with bits 10 and 11 set to 0. Bit 11of register B is set when a DTF occurs and must be cleared before the next DTF to avoid a timing error. When any error occurs, bit 0 of register B and the corresponding bits 1-5 will be set depending on the error. This bit must be cleared to avoid further interrupts on the same condition. All error flags (i.e., status register B) are cleared by issuing a DTXA instruction with AC bit 10 set to 0.

## DECtape System Programming Information

The seven functions available with the TC02 and their octal numbers as specified by the bits 6-8 of the AC are as follows:

| *Function* | *Octal No.* |
|---|---|
| MOVE | 0 |
| SEARCH | 1 |
| READ DATA | 2 |
| READ ALL | 3 |
| WRITE DATA | 4 |
| WRITE ALL | 5 |
| WRITE TIMING and MARK TRACK | 6 |
| Unused at present (select error if given) | 7 |

All functions take place in either direction and in either normal mode (NM) or continuous mode (CM). NM differs from CM only in the fact that the DECtape flag (DTF) occurs at more frequent intervals in NM. The DTF settings which occur in NM are eliminated in the CM until word count overflow (WC) has occurred.

Move - The MOVE function simply sets the selected unit in motion (forward/reverse). NM and CM have no meaning and are ignored in this function alone. When the tape enters either end zone* (i.e., beginning of tape (BOT) and end of tape (EOT)), and the unit in question is selected:

1. the error flag (EF) is set.

2. the EOT bit (bit 2 of status register B) is set.

3. an interrupt occurs**

A program check on the forward/reverse motion bit (AC bit 3) of the status register will determine whether EOT or BOT occurred. However, if the unit is deselected, the tape runs off the reel with no flags raised and no interrupt. In order to stop a selected unit at any time, the GO bit (AC bit 4) must be set to 0.*** Once a unit is deselected, status information pertaining to that unit is no longer accessible unless it was saved by the program prior to deselection.

Search - The search function provides the capability of random access of data blocks on DECtape. This function is used to locate the number of the block to or from which data transfer will occur. In normal mode at each block mark until EOT occurs, the DTF is raised and an interrupt occurs. The block

*If either end zone is entered during turn around or during stopping of tape, the EOT bit is not set and no interrupt occurs.

**All references to the occurrence of interrupts assume both:

1. The program interrupt is on.
2. The DTF and EF have been enabled to the program interrupt or API (i.e., bit 9 of status register A is set to a 1). If either of these is not true, flags are raised and status bits are set (and may be sensed and/or cleared), but no interrupt occurs.

***When setting the GO bit to 0, the forward/reverse motion bit and unit selection bits should not be changed from their current status. The hardware accepts the change in the TC02 (i.e., status A bit 3 changes from its former state) without error indication, but does not pass this change on to the transport. Programming confusion can result.

number is automatically transferred by the hardware into the memory location specified by the CA. The CA must have been set previously by the program but the contents are not incremented. The WC is incremented at each DTF; the program must clear the DTF bit in the status register and check the block number until the desired one is found.

In continuous mode, the WC is set to the 2s complement of the number of blocks to skip. At each block mark, the block number is read into the memory location specified by the CA which is not incremented. The DTF is raised only at the block mark at which the WC overflows. At that time, an interrupt occurs. Continuous mode provides a virtually automatic DECtape search.

Read Data - READ DATA is used to transfer blocks of data into core memory with the transfer controlled by the standard tape format. The standard block length is 256 18-bit words. For this and all following functions, the CA register initially must be set to (the transfer memory location - 1) because the CA register is incremented just before each word transfer. The WC register is also incremented prior to each word transfer so must be set to the 2s complement of the number of words to be transferred prior to the transfer. Data may be transferred in forward or reverse.

Any number of words equal to or less than 1 block may be transferred in NM. The DTF is raised and an interrupt occurs at the end of each block. The DTF must be cleared before the beginning of the next block (i.e., 1.7 msec) to avoid an erroneous timing error, (see summary). When partial blocks are transferred data transmission will have been ended with WC overflow (i.e., the word which causes the WC overflow is the last one transferred). However, the remainder of the block is read and parity checked before the DTF and interrupt occur. Tape motion continues until the GO bit is reset to 0 by the program. If the GO bit is not reset to a 0 or a new function specified before the end of the next block, a timing error will occur. READ DATA in NM is intended primarily for single, 256-word, block transfers. If any other number of words is to be transferred, it is advantageous to use CM. However, if the programmer chooses to use NM for any other number of words, the program must check for WC overflow at each interrupt since there is no other way to determine when to stop the tape or change to another function. When the WC overflow occurs, it is essential that the function be changed or the GO bit set to 0. Otherwise transfer begins again (the IOT to clear the DTF implicitly specifies the same function again) at the next block (or next word for the ALL functions) since WC = $000000_8$ is valid.

Any number of words may be transferred in CM. However, the DTF and an interrupt occur only once after a WC overflow and an end of block. The comments concerning tape continuation apply in CM as well as NM.

Read All - The READ ALL function allows information to be read from an unusually formatted tape essentially reading all data channels recorded on DECtape regardless of the mark track value. During the READ ALL function the DECtape control does not distinguish between different marks recorded on the mark track – except to check for mark track errors (MKTK).

In normal mode (NM), the DTF is raised and causes an interrupt at the end of each 18-bit word transfer. Data transfer stops after WC overflow, but tape motion continues until the GO bit is set to 0 or a new function is specified (in both NM and CM). If the DTF is not cleared after each word transfer, a timing error occurs at the end of the next word (i.e., 200 microseconds later).

For continuous mode, the DTF is raised and causes an interrupt at WC overflow only. If this interrupt is ignored no more data transfers occur but tape motion continues to EOT.

Write Data - The write enable switch on the TU55 must be in write enable position for all WRITE functions. All the details of the READ DATA function description apply with the following exceptions.

In normal mode, the DTF is set to a 1 at the end of each block. If WC0 did not occur in the block just ended and a new function is specified, the next block will be written (provided the DTF has been cleared). If WC overflow did occur in the block just ended and no new function is specified, the tape continues to move but the writers are disabled. In both CM and NM, when partial blocks are written, data transfer from core to DECtape stops at WC overflow. 000000s are written in the remaining data words of the block and the parity check character is computed over the entire block and recorded.

In continuous mode, the DTF is set at the end of the block in which WC overflow occurred. Therefore, if no new function is specified, the tape continues to move but the writers are disabled.

Write All - All the details of the READ ALL function description apply. The WRITE ALL function is used to write an unusual format (such as block numbers on DECtape after timing and mark tracks have been recorded). The word which causes WC overflow is the last one written in NM or CM. The tape continues to move but the writers are disabled.

NOTE: Change of function must be delayed for 90 microseconds to insure recording of last word. Alternative method: set WC to 1 greater than desired number of word transfers and change function within 40 microseconds after WCO.

Write Timing and Mark Track - This function and only this function may be performed with the selector switch on write timing and mark track (WRTM) on the maintenance control panel. Whereas the timing track is actually hardware recorded during execution of this function, the mark track is generated and recorded by program. The value written in the mark track is determined by bits 0, 3, 6, 9, 12, and 15 of the 18-bit word being written (i.e., the same bits assigned to channel 1).

CM may be conveniently used for this function since the hardware WC provides an automatic counter and interrupt at WC overflow only; in NM, the DTF and interrupt occur at every word until WC overflow. In NM, after WC overflow, if the GO bit or DECtape flag are not cleared, a timing error occurs and no more data is recorded. After WC overflow in CM, if the GO bit is not set to 0, zeros are written down on tape.

Enable the Interrupt Feature - The enable-to-the-interrupt feature allows the program to remove DECtape from the program interrupt line (even if the interrupt is ON). This is primarily of value in the automatic priority interrupt system.

When command bit 9 in the status register is set to a 1, the TC02 is connected to the interrupt system. If this bit is 0, the DTF in the TC02 cannot cause an interrupt even if the interrupt facility in the PDP-9 is ON. Similarly, any of the five error conditions will cause an interrupt if bit 9 is set to 1 in the status register but cannot cause a program interrupt if bit 9 is a 0.

Whether this bit is set or not does not influence the setting of status bits 0-5 of the status register B upon receipt of an error flag (EF) or DTF. Similarly, the result of the I/P skip instruction is independent of the condition of this bit.

Error Conditions - Five types of errors can be detected in the use of DECtape:

Timing Error
Parity Error
Select Error
End of Tape
Mark Track Error

For all errors the EF is raised, a bit is set in the status register and an interrupt occurs (if the enable-to-interrupt bit has been set). The DTEF instruction skips on the inclusive OR of those error bits; hence, each status bit must be checked to determine the kind of error. For all but the parity error, the selected transport is stopped and the EF is raised at the time of error detection. No DTF occurs. For a parity error, the GO bit remains 1 (i.e., motion continues) and the EF is raised simultaneously with the DTF in NM. Only 1 interrupt occurs; hence the program must check the EF.

A parity error in CM raises the EF at the end of the block in which the parity occurs causing an interrupt (if enabled). If no program action is taken, e.g., stop transport or reverse and re-read, data transfer continues and the DTF is raised and causes an interrupt at WC overflow and end of final block read.

Timing Error - A timing error (program malfunction) is a 'data miss' or program failure to clear the DTF status bit. A timing error occurs also if the program switches to READ or WRITE DATA function while the DECtape is currently passing over a data area on tape.

Parity Error - A parity error occurs only during the READ DATA function for a hardware computed parity check character (PCC) failure.

Select Error* - A select error will result under any of the following conditions:

1. Selection of zero or > 1 unit.

2. Attempt to write on DECtape transport with WRITE ENABLE/WRITE LOCK switch in the WRITE LOCK position.

3. Attempt to select unit for any function with DECtape transport REMOTE/OFF/LOCAL switch in the OFF or LOCAL (off-line) position.

*No-tape or tape-run-off-reel conditions are not detectable.

4. Attempt to write timing and mark tracks with the DECtape controls switch in any position other than write timing and mark track.

5. Attempt to perform any function other than write timing and mark tracks with the DECtape control switch in the write timing and mark track position.

6. Attempt to perform any function other than read all with DECtape controls switch in the read mark track position.

7. Attempt to execute unused function (7).

End of Tape - An EOT error occurs when the DECtape enters either end zone with the GO bit = 1 and the forward/reverse direction bit set to continue in the same direction. In NM and CM data transfer stops at the last legitimate block, the EF is raised, the tape transport stops and an error interrupt occurs.

Mark Track Error - A mark track error occurs when the DECtape control fails to recognize a legitimate mark on the mark track. The error may occur in all but the move or write timing and mark track functions. In both CM and NM, the EF is raised, the tape transport stops and an interrupt occurs.

## DECtape Programming Examples

Illustrated below are a few examples of possible ways to code DECtape functions on the PDP-9/L. Some are intended to illustrate the obvious capabilities of DECtape. Others demonstrate some of the more obscure features. Assumed as part of the hardware configuration is the API option. The examples are written in PDP-9 Basic Symbolic Assembler language.

*Auto-Search* - The combined use of NM and CM (example 1) provides virtually automatic DECtape search for a desired block number with a minimum (2) of interrupts.

1. Search forward in NM to find next block number.

2. Compute difference between this and the desired block number.

3. Set WC=2s complement of this difference.

4. Switch to search in CM.

The next interrupt will be at the desired block number.

*Read Data (Continuous Mode)* - Assuming the correct block number has been found, example 2 illustrates a possible way to code a data transfer function in continuous mode. One interrupt will occur at the end of the transfer. This example continues from and relies on the preceding example.

*Read Data (Normal Mode)* - Normal mode provides a convenient tool for double buffering or processing large amounts of data on a block-by-block basis for economic use of core storage. (See example 3.) It also allows for transfer of non-contiguous blocks of data into contiguous locations or vice versa. This example also continues from and relies upon example 1.

*Bootstrap Loading Technique* - The data channel facility and the design of CM allow for linked loading of data from DECtape where the first two DECtape data words determine the core location and amount of data which follows. Note that the following technique will work only for a data channel device whose CA and WC registers are in core memory locations. The address into which data is loaded is specified by CA. Thus is the CA points to the WC-1, the first word transferred specifies the number of words to be transferred. After the first data word transfer, the CA points to itself and the second word transferred specifies where the following data is to be loaded. No program interrupts, timing or computation is required to locate the data. Only the TC02 and DCH features are used.

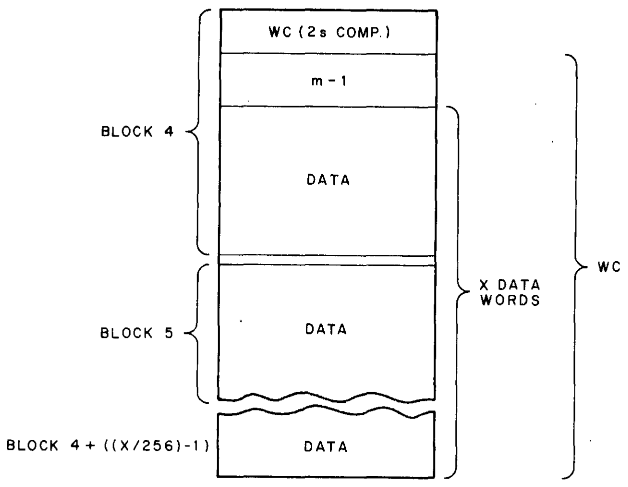
*Problem:*

Load x data words beginning at DECtape block 4 of unit 3 into core locations M to M+X-1, assuming tape has been positioned at block 4.

```
/BOOTSTRAP EXAMPLE

            .
            .
            DZM 30       /TO INSURE NO WC OVERFLOW
            LAC (27      /TO BEGIN LOADING AT REGISTER
                         /30
            DAC 31       /CA
            LAC IOTD     /IOT DATA
            DTLA         /LOAD STATUS REGISTER
            .
            .
IOTD,       332400       /READ ON UNIT  3, CM, FORWARD,
                         /GO (1)
            .            /WITH INTERRUPT ENABLED
            .
            .
```

The following represents the format of the data on the tape starting at block 4.



WHERE X IS AN INTREGRAL MULTIPLE OF 256

*Writing and Reading in Opposite Directions* - As mentioned earlier, it is possible (though non-trivial) to read data from a DECtape in the opposite direction from which the data was written via program manipulation. A re-ordering of both the entire block and individual words is required, however.

a. Block Re-ordering: A block of words $x_n$ $x_1$ recorded in one direction is loaded into core as $x_n$ $x_1$ when read in the opposite direction.

b. Word Unscrambling: Data read in backwards comes into core memory locations from the TC02 in the following 18-bit format:

Bits:

```
— — — — — — — — — — — — — — — — — —
15 16 17 12 13 14 9 10 11 6 7 8 3 4 5 0 1 2
```

In bit positions:

$0 \longleftrightarrow 17$

NOTE: If data is to be re-ordered on the fly, the routine is limited to 140 microseconds since the word transfer rate = 200 microseconds (±30%). The probability of such a routine not working is very high if interrupts from other devices are encountered.

*Miscellaneous Information* - Additional information concerning DECtape programming is provided in the subsequent paragraphs.

Scatter Read/Gather Write - By program manipulation in CM, it is possible to scatter read or gather write on DECtape. A separate programmed WC must be maintained and incremented as the hardware increments its WC. When program WC overflows, the CA may be reset to the beginning of another core area. With a 200 microsecond word transfer rate, (±30%) there is ample time to reset the CA. Note that this function is impossible if other interrupts are likely to occur.

Modification of Individual Data Words - This technique should not be used.

Data Transfer -- Upper Bound Protection - The WC controls are data transfers. After WC overflow, no more date transfers take place. Thus, to protect memory when reading a block of unknown length, the WC is set to the 2s complement of the difference between the initial address where data is transferred and the upper bound.

Similar action prevents writing beyond a predetermined point on tape when transferring an unknown number of words from core.

Special Formats on Tape - The user is cautioned to always specify an even number of words in his special format. If he does not, the control will indicate parity errors where none exist.

Programming Note: When a turn-around command is issued (i.e., complement the direction bit while the GO bit remains set to 1), the tape may not be up to speed when the point at which the command was issued is passed (in the new direction). The tape will be up to speed one standard 256 word block length after the turn-around point. Therefore, to find a block in the opposite direction it is sufficient to delay the turn around one block as shown in figure 5-2.

With this turn around specification finding blocks next to the end zones requires special handling. Block 0 forward may be found if the tape is backed into the end zone twice before turning around. To prevent this special end zone handling, a new formatting program must be written which provides one block length of inter-block zone marks (no-op marks)

*Example 1:*

```
/AUTO-SEARCH EXAMPLE
               X/
BEGIN,                      LAC (CBLK        /CBLK=TEMP STORAGE
                            DAC 31           /CA - DECTAPE
                            DZM 30           /INSURE NO WC OVERFLOW
                            LAC (JMP SEARCH
                            DAC SWITCH
                            LAC (321600      /SEARCH DECTAPE UNIT 3, IN FORWARD
                                             /DIRECTION, NM, GO (1), FLAGS ENABLED
                            DTLA             /LOAD STATUS REGISTER
                             .
                             .
                             .
               44/
                            JMS DECTAPE      /DECTAPE=API CHANNEL 44

               Y/
DECTAPE,                    O                /INTERRUPT SERVICE ROUTINE
                            DAC ACSAV
                            DTEF             /SKIP: EF
                            SKP
                            JMP DECER        /SERVICE DECTAPE ERROR TO BE WRITTEN BY USER
                            DTDF             /SKIP ON DTF - REQUIRED ONLY IF OTHER
                                             /DEVICES ON SAME API CHANNEL
                            SKP
SWITCH,                     JMP SEARCH       /OR JMP REDE
                             .
                             .
                             .
               Z/
SEARCH,                     LAC CBLK         /CURRENT BLOCK NUMBER
                            AND (007777
                            SAD RBLK         /DESIRED BLOCK NUMBER
                            JMP RBLKS        /SERVICE CORRECT BLOCK
                            CLC              /COMPUTE BLOCK NUMBER
                            TAD CBLK         /DIFFERENCE AND DIRECTION
                            DAC TEMP         /OF SEARCH
                            CLC              /53 MS ARE AVAILABLE TO
                            TAD RBLK         /SWITCH FROM SEARCH NM
                            CMA              /TO SEARCH CM
                            ADD TEMP
                            SMA
                            JMP REV          /REVERSE DIRECTION
                            DAC 30           /WC=2'S COMP OF DIFFERENCE
                            LAC (010000*     /CM
                            DTXZ             /XOR STATUS    AC AND
                             .               /LOAD STATUS REGISTER
                             .               /NEXT INTERRUPT WILL BE
                             .               /DESIRED BLOCK NUMBER
                            LAC ACSAV
                            DBR
                            JMP I DECTAP     /EXIT
```

*If the program were to clear and then load status register A, the control would cause the select delay of 120 msec. However, by simply setting the bit to a 1 in the AC for the corresponding status register bit, the change is made but no select delay occurs.

*Example 2:*

```
/READ DATA EXAMPLE
               A/
RBLKS,                      CLC              /LOAD AC WITH - 1
                            TAD ADDR         /SET CA=ADDRESS - 1
                            DAC31
                            CLC              /SET WC=2'S COMPLEMENT
                            TAD WDCNT        /OF NO. OF WORDS TO TRANSFER
                            CMA              /AND LOAD STATUS REGISTER
                            DAC 30
SELRD,                      LAC (003000      /READ DATA, FORWARD, GO, CM,
                            DTXA             /FLAGS ENABLED, UNIT 3
```

*Example 2 con't:*

```
                              LAC (JMP REDE         /RESET INTERRUPT CHAIN
                              DAC SWITCH            /JMP
                               .
                               .
                              LAC ACSAV
                              DBR                   /DEBREAK
                              JMP I DECTAP          /EXIT
ADDR,                         ADDRES
WDCNT,                        N
```

*Example 3:*

```
/DOUBLE BUFFER EXAMPLE
          A/                                        /CONTINUES FROM SEARCH EXAMPLE
RBLKS,                        CLC
                              TAD WDCNT             /WC=2'S COMPLEMENT OF
                              CMA                   /1 NO. OF WORDS TO TRANSFER
                              DAC 30
RESET1,                       LAC (BUF1
RESET2,                       DAC ADDR
                              CLC
                              TAD ADDR              /CA = ADDRESS - 1
                              DAC 31
                              LAC (013000           /LOAD STATUS REGISTER:
                              DTXA                  /READ DATA, FORWARD, GO, NM
                              LAC (JMP REDE         /FLAGS ENABLED, UNIT 3
                              DAC SWITCH            /RESET INTERRUPT CHAIN JMP
                               .
                               .
                              LAC ACSAV
                              DBR
                              JMP I DECTAP          /EXIT
REDE,                         LAC 30                /CHECK WC OVERFLOW
                              SMA
                              JMP STOP              /STOP DECTAPE
                              LAC (BUF2             /NO OVERFLOW
                              SAD ADDR              /RESET ADDR TO
                              JMP RESET1            /SWITCH BUFFER
                              JMP RESET2
STOP,                         LAC (020000           /SET GO = 0, CLEAR DTF AND EF
                              DTXA                  /LOAD STATUS REGISTER
                               .
                               .
                               .
```

*Example 4:*

```
/SUBROUTINE TO REORDER A DECTAPE BLOCK OF N WORDS WHERE N=EVEN NUMBER
/USES SUBROUTINE UNSCR TO UNSCRAMBLE INDIVIDUAL WORDS. ENTRANCE
/PARAMETERS: BUFFER LOCATIONS – HIGH AND LOW, HIGH LOC = C(DTHAD)
/LOW LOC = C(DTLAD) 23 DECIMAL REGISTERS, 37 MICROSECONDS FOR EVERY 2 WORDS OR
/24.2 MS FOR REORDERING AND UNSCRAMBLING 256 DECIMAL WORDS.

DTORD,                        O
DTBEG,                        LAC I DTLAD           /SAVE LOWEST UNSCRAMBLED
                              DAC DTTAM             /WORD OF BLOCK
                              LAC I DTHAD           /UNSCRAMBLE HIGH WORD
                              JMS UNSCR             /UNSCRAMBLE 1 WORD SUBR
                              DAC I DTLAD           /STORE IN FREE LOW LOC
                              LAC DTTAM             /UNSCRAMBLE LOW WORD
                              JMS UNSCR
                              DAC I DTHAD           /STORE IN FREE HIGH LOC
                              ISZ DTLAD             /INCREMENT LOW ADDRESS
                              LAC DTLAD             /WHEN DTLAD+1 = DTHAD
                                                    /REORDERING IS COMPLETE
                              SAD DTHAD
DTEXIT,                       JMP I DTORD           /EXIT
                              CLC
                              TAD DTHAD             /DECREMENT HIGH ADDRESS
                              DAC DTHAD
                              JMP DTBEG             /UNSCRAMBLE NEXT SET OF 2
DTLAD,                        O                     /LOW, HIGH BLOCK ADDRESSES
DTHAD,                        O
DTTAM,                        O                     /TEMPORARY STORAGE
```

*Example 4 con't:*

```
/UNSCR, 53 DECIMAL REGISTERS, 76 MICROSECONDS SUBROUTINE TO UNSCRAMBLE ONE 18-BIT
/DECTAPE WORD IN AC.  RESULT IN AC
UNSCR,                          O                       /RETURN ADDRESS
                                CMA                     /COMPLEMENT
                                DZM UNT                 /INITIALIZE INTERMED.  RESULT REG
                                RALVCLL
                                RTL
                                DAC UNT1                /ARG. TEMPORARY STORAGE
                                AND (007000             /BITS 6, 7, 8
                                XOR UNT                 /ASSEMBLE BITS IN
                                DAC UNT                 /INTERMED.  RESULT REGISTER
                                LAC UNT1                /RESUME CYCLING
                                RAL
                                DAC UNT1                /BITS 15, 16, 17
                                AND (000007
                                XOR UNT
                                DAC UNT
                                LAC UNT1
                                RAL
                                RTL
                                RTL
                                DAC UNT1                /BITS 3, 4, 5
                                AND (070000
                                XOR UNT
                                DAC UNT
                                LAC UNT1
                                RAL
                                DAC UNT1                /BITS 12, 13, 14
                                AND (000070
                                XOR UNT
                                DAC UNT
                                LAC UNT1
                                RAL
                                RTL
                                RTL
                                DAC UNT1                /BITS 0, 1, 2
                                AND (700000
                                XOR UNT
                                DAC UNT
                                LAC UNT1
                                RAL
                                DAC UNT1                /BITS 9, 10, 11
                                AND (000700
                                XOR UNT
                                JMP I UNSCR             /EXIT WITH AC = RESULT
UNT,                            0                       /INTERMED.  RESULT OF UNSCRAMBLING
UNT1,                           0                       /ARG TEMP. STOR.
```

so that program can bounce off the end zone and find block 0 (if the tape has the new format on it). The end zone problem is also solved for either format by not using the block next to the end zones (block 0, 1101).

When using non-standard format tape (i.e., not $1102_8$ blocks of $400_8$ words) a length of tape equal to one 18 bit, $256_{10}$ ($400_8$) word block must pass the head before the turn around command is issued. This is approximately five inches of tape. However, when calculating the required delay for a non-standard format tape it should be computed in equivalent standard block lengths.

*Example:* Turn around delay calculation for blocks of $94_{10}$ words.

$$\frac{256 \text{ words/block}}{92 \text{ words block}} \times 1 \text{ block delay} = \frac{2.7 \text{ block delay}}{\text{required}}$$

*DECtape Function Summary* - The DECtape function summary is provided in table 5-2.

*DECtape Error Summary* - The DECtape error summary is provided in table 5-3.

*DECtape Timing Data* - The DECtape timing data on standard format (certified) tape is provided in table 5-4.

## MAGNETIC TAPE CONTROL, TYPE TC59

The Type TC59 will control the operation of a maximum of eight digital magnetic tape transports manufactured by Digital Equipment Corporation. The Type TC59 interfaces and uses the PDP-9/L data channel (DCH) facility to execute data transfers between system core memory and magnetic tape. Transfers are governed by the in-memory word counter (WC) and current address (CA) register associated with

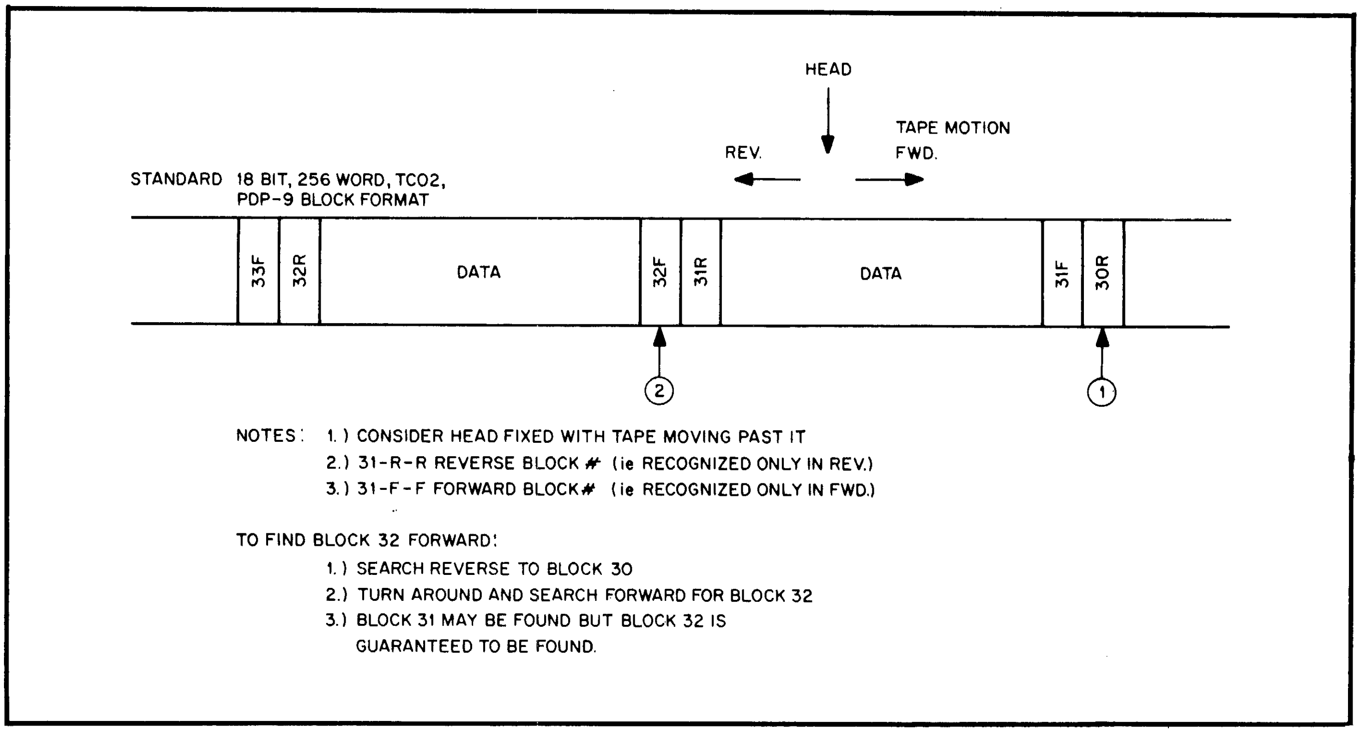

HEAD
TAPE MOTION
REV.
FWD.
STANDARD 18 BIT, 256 WORD, TC02, PDP-9 BLOCK FORMAT
33F
32R
DATA
32F
31R
DATA
31F
30R
2
1
NOTES: 1.) CONSIDER HEAD FIXED WITH TAPE MOVING PAST IT
2.) 31-R-R REVERSE BLOCK # (ie RECOGNIZED ONLY IN REV.)
3.) 31-F-F FORWARD BLOCK # (ie RECOGNIZED ONLY IN FWD.)
TO FIND BLOCK 32 FORWARD:
1.) SEARCH REVERSE TO BLOCK 30
2.) TURN AROUND AND SEARCH FORWARD FOR BLOCK 32
3.) BLOCK 31 MAY BE FOUND BUT BLOCK 32 IS GUARANTEED TO BE FOUND.

Figure 5-2 Location of Block in Opposite Direction

TABLE 5-2 DECTAPE FUNCTION SUMMARY

| Function | Normal Mode (NM) | | Continuous Mode (CM) | |
|---|---|---|---|---|
| 0. Move | DTF: | No Interrupt | Same as NM | |
| | CA: | Ignored | | |
| | WC: | Ignored | | |
| 1. Search | DTF: | Interrupt at each block mark | DTF: | Interrupt at block mark where WC overflows |
| | CA: | Not incremented | CA: | Not incremented |
| | WC: | Incremented at each block mark | WC: | Incremented at each block mark |
| 2. Read Data | DTF: | Interrupt at end of each block | DTF: | Interrupt at WC overflow and end of block |
| | CA: | Incremented at each word transfer | CA: | Incremented at each word transfer |
| | WC: | Incremented at each word transfer | WC: | Incremented at each word transfer |
| 3. Read All | DTF: | Interrupt at each word transfer | DTF: | Interrupt at WC overflow |
| | CA: | Incremented at each word transfer | CA: | Incremented at each word transfer |
| | WC: | Incremented at each word transfer | WC: | Incremented at each word transfer |
| 4. Write Data | Same as 2. Read Data | | | |
| 5. Write All | Same as 3. Read All | | | |
| 6. Write Timing & Mark Tracks | Same as 3. Read All | | | |
| 7. Unused* | | | | |

*If used by mistake, the control gives a Select Error (SE).

TABLE 5-3 DECTAPE ERROR SUMMARY

| Function | Normal Mode | Continuous Mode (CM) |
|---|---|---|
| Move | Select Error<br>EOT | Select Error<br>EOT |
| Search | Select Error<br>EOT<br>Timing Error<br>MK TRK Error | Select Error<br>EOT<br>Timing Error<br>MK TRK Error |
| Read Data | Select Error<br>EOT<br>Timing Error<br>Parity Error<br>MK TRK Error | Select Error<br>EOT<br>Timing Error<br>Parity Error<br>MK TRK Error |
| Read All | Select Error<br>EOT<br>Timing Error<br>MK TRK Error | Select Error<br>EOT<br>Timing Error<br>MK TRK Error |
| Write Data | Select Error<br>EOT<br>Timing Error<br>MK TRK Error | Select Error<br>EOT<br>Timing Error<br>MK TRK Error |
| Write All | Select Error<br>EOT<br>Timing Error<br>MK TRK Error | Select Error<br>EOT<br>Timing Error<br>MK TRK Error |
| Write Timing & Mark Tracks | Select Error<br>Timing Error | Select Error<br>Timing Error |

TABLE 5-4 DECTAPE TIMING DATA

| Operation | Time |
|---|---|
| Time to answer data channel request | Up to 66 microseconds* |
| Word Transfer Rate | 1 18-bit word every 200 microseconds* |
| Block Transfer Rate | 1 256 word block every 53 milliseconds* |
| Start Time | 375 milliseconds (± 20%) |
| Stop Time | 375 milliseconds (± 20%) |
| Turn Around Time (see programming note on page 5-20) | 375 milliseconds (± 20%) |
| Search ⟶ Read Data Function change for present block | Up to 400 microseconds* |
| Search ⟶ Write Data Function change for present block | Up to 400 microseconds* |
| Read ⟶ Search Function change for next block number | Up to 1000 microseconds* |

TABLE 5-4 DECTAPE TIMING DATA (continued)

| Operation | Time |
|---|---|
| Write ⟶ Search Function change for next block number | Up to 1000 microseconds* |
| DTF to beginning of next data block | 1.7 milliseconds* |
| DTF Occurrenct: Move: NM, CM | Never |
| Search: NM<br>Read Data: NM<br>Write Data: NM | Every 53 milliseconds* |
| Search: CM | (WC) X53 milliseconds* |
| Read Data: CM<br>Write Data: CM | (No. of blocks) X53 milliseconds* |
| Read All: NM<br>Write All: NM<br>Write Timing & Mark Tracks: NM | Every 200 microseconds* |
| Rea Read All: CM<br>Write All: CM<br>Write Timing & Mark Tracks: CM | (WC) X200 microseconds* |

*(± 30%)

the assigned data channel (memory locations 32 and $33_8$). Since the CA is incremented before each data transfer, its initial contents should be set to the desired initial address minus one. The WC is also incremented before each transfer and must be set to the 2's complement of the desired number of data words to be transferred. In this way the word transfer which causes the word count to overflow (WC becomes zero) is the last transfer to take place. The number of IOT instructions required for the Type TC59 is minimized by transferring all necessary control data (i.e., unit number, function, mode, direction, etc.) from the PDP-9/L accumulator (AC) to the control using IOT instructions. (Refer to Table 5-5.) Similarly, all status information (i.e., status bits, error flags, etc.) can be read into the AC from the control unit by IOT instructions.

During normal data reading, the control assembles 18-bit length computer words from successive frames read from the information channels of the tape. During normal data writing the control disassembles 18-bit words and distributes the bits so they are recorded on successive frames of the information channels. The control provides for selection of four recording densities: 200, 556, 800, and 800/9-channel.

Although any number of tapes may be simultaneously rewinding, data transfer may take place to or from only one transport at any given time. In this context, data transfer includes these functions: read or write data, write EOF (end of file), read/compare and space. When any of these functions are in process, the tape control is in the "not ready" condition. A transport is said to be "not ready" when tape is in motion, when transport power is off, or when it is off-line.

Data transmission may take place in either parity mode, odd-binary or even-BCD. When reading a record in which the number of characters is not a multiple of the number of characters per word, the final characters come into memory left-justified.

Ten bits in the magnetic tape status register retain error and tape status information. Some error types are combinations, such as lateral and longitudinal parity errors (parity checks occur after both reading and writing of data), or have a combined meaning, such as illegal command, to allow for the maximal use of the available bits.

The magnetic tape status register reflects the state of the currently selected tape unit. Inter-

TABLE 5-5 TC59 CONTROL IOT INSTRUCTIONS

| Mnemonic | Octal Code | Description |
|---|---|---|
| MTSF | 707341 | 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 to skip the next sequential instruction. |
| MTCR | 707321 | Skip on tape control ready (TCR). If the tape control is ready to receive a command, the contents of the PC are incremented by one to skip the next sequential instruction. |
| MTTR | 707301 | Skip on tape transport ready (TTR). The next sequential instruction is skipped if the tape transport is ready. |
| MTAF | 707322 | Clear the status and command registers, and the EF and MTF if tape control ready. If tape control not ready, clear MTF and EF flags only. |
| ----- | 707302 | Inclusively OR the contents of the command register into bits 0-11 of the AC. |
| MTCM | 707324 | Inclusively OR the contents of AC bits 0-5, 9-11 into the command register; jam transfer bits 6, 7, 8 (command function). |
| MTLC | 707326 | Load the contents of AC bits 0-11 into the command register. |
| ----- | 707342 | Inclusively OR the contents of the status register into bits 0-11 of the AC. |
| MTRS | 707352 | Read the contents of the status register into bits 0-11 of the AC. |
| MTRC | 707312 | Read the contents of the command register into bits 0-11 of the AC. |
| MTGO | 707304 | Set GO bit to execute command in the command register if command is legal. |

rupts may occur only for the selected unit. Therefore, other units which may be rewinding, for example, will not interrupt when done.

A special feature of this control is the "Write Extended Inter-Record Gap" capability. This occurs on a write operation when Command Register bit 5 is set. The effect is to cause a 3 inch inter-record gap to be produced before the record is written. The bit is automatically cleared when the writing begins. This is very useful for creating a 3 inch gap of blank tape over areas where tape performance is marginal.

## Magnetic Tape Functions

For all functions listed below, upon completion of the data operation (after the end-of-record character passes the read head), the MTF (magnetic tape flag) is set, an interrupt occurs (if enabled), and errors are checked.

*No Operation* - A NO OP command defines no function in the command register. A MTGO instruction with NO OP will cause an illegal command error (set EF).

*Space* - There are two commands for spacing records, space forward and space reverse. The number of records to be spaced (2's complement) is loaded into the WC. CA need not be set. MTF (magnetic tape flag) is set, and an interrupt occurs at WC overflow, EOF (end of file), or EOT (end of tape), whichever occurs first. When issuing a space command, both the density and parity bits must be set to the density and parity in which the records were originally written.

Load Point or Beginning of Tape (BOT) detection during a backspace terminates the function with the BOT bit set. If a space reverse command is given when a transport is at BOT, the command is ignored, the illegal command error and BOT bits are set, and an interrupt occurs.

*Read Data* - Records may be read into memory only in the forward mode. Both CA and WC must be set: CA, to the initial core address minus one; WC, to the 2's complement of the number of words to be read. Both density and parity bits must be set.

If WC is set to less than the actual record length, only the desired number of words are transferred into memory. If WC is greater than or equal to the actual record length, the entire record is read into memory. In either case, both parity checks are performed, the MTF is set, and an interrupt occurs when the end-of-record mark passes the read head. If either lateral or longitudinal parity errors or bad tape have been detected, or an incorrect record length error occurs (WC not equal to the number of words in the record), the appropriate status bits are set. An interrupt occurs only when the MTF is set.

To continue reading without stopping tape motion, MTAF (clear MTF) and MTGO instructions must be executed. If the MTGO command is not given before the shutdown delay terminates, the transport will stop.

*Write Data* - Data may be written on magnetic tape in the forward direction only. For the write data function, the CA and WC registers and density and parity bits must be set. Write data is controlled by the WC, such that when the WC overflows, data transfer stops, and the EOR (end of record) character and IRG (inter-record gap) are written. The MTF is set after the EOR has passed the read head. To continue writing, a MTGO command must be issued before the shutdown delay terminates. If any errors occur, the EF will be set when the MTF is set.

*Write EOF* - The Write EOF command transfers a single character ($17_8$) record to magnetic tape and follows it with EOR character. CA and WC are ignored for write EOF. The density bits must be set, and the command register parity bit should be set to even (BCD) parity. If it is set to odd parity, the control will automatically change it to even.

When the EOF marker is written, the MTF is set and an interrupt occurs. The tape transport stops, and the EOF status bit is set, confirming the writing of EOF. If odd parity is required after a write EOF, it must be specifically requested through the MTLC command.

*Read/Compare* - The read/compare function compares tape data with core memory data. It can be useful for searching and positioning a magnetic tape to a specific record, such as a label or leader, whose content is known in core memory, or to check a record just written. Read/compare occurs in the forward direction only; CA and WC must be set. If there is a comparison failure, incrementing of the CA ceases, and the read/compare error bit is set in the status register. Tape motion continues to the end of the record; the MTF is then set and an interrupt occurs. If there has been a read/compare error, examination of the CA reveals the word that failed to compare.

*Rewind* - The high-speed rewind command does not require setting of the CA or WC. Density and parity settings are also ignored. The rewind command rewinds the tape to load point (BOT) and stops. Another unit may be selected after the command is issued and the rewind is in process. MTF is set, and an interrupt occurs (if the unit is selected) when the unit is ready to accept a new command. The selected unit's status can be read to determine or verify that rewind is in progress.

### *Continued Operation*

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

   a. MTCM, if the command is to be changed.

   b. MTAF (will only clear MTF and EF flags since tape control will be in a Not Ready state).

   c. MTGO (if LCM requested an illegal condition, the EF will be set at this time).

2. To change modes of operation, either in the same or opposite direction, the MTCM command is given to change the mode and a MTGO command is given to request the continued operation of the drive. If a change in direction is ordered, the transport will stop, pause, and automatically start up again.

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

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

NOTE: No flags will be set when the control becomes ready or the transport becomes ready, except if the rewind command is present in the command register and the selected drive reaches BOT and is ready for a new command.

5. If a write (odd parity) command is changed to write EOF, the parity is automatically changed to even.

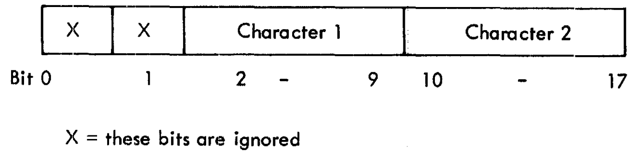
NOTE: Even parity will remain in the command register unless changed by a new command instruction, MTLC, which clears and loads the entire command register.

## 9-Track Operation

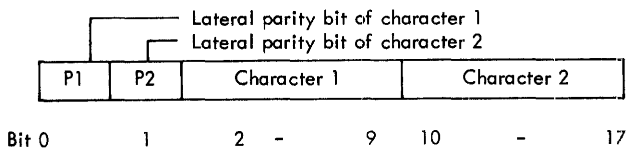
Nine- and seven-track transports may be intermixed on the Type TC59 control. When a transport is selected, it automatically sets the control for proper operation with its number of tracks.

Control of nine-track operation is identical to seven-track except as noted below.

*Write* - A word in memory is written on tape with the format shown below.



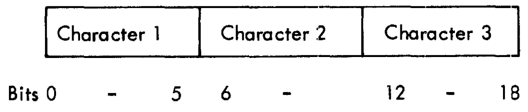
*Read* - A word is read into memory from tape with the format shown below.


Lateral parity bit of character 1
Lateral parity bit of character 2
P1
P2
Character 1
Character 2
Bit 0
1
2 - 9
10 - 17

*Read/Compare* - A direct comparison of the characters on tape is made with those in memory. The parity bits are ignored, as are bits 0 and 1 of each memory word.

*Core Dump Mode* - This mode is used only with nine-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 three six-bit characters. The format is as follows.



This is accomplished by only utilizing seven of the nine tracks on the tape.

Tape written in core dump mode must be read (read/compare) in the same mode. These operations are the same as for a seven-track transport.

### 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 as follows.

| *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.

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

2. When the selected transport becomes ready following a normal rewind function.

These functions will also set the EF if any errors are present. This flag sets bit 11 in the AC if an IORS instruction is issued.

*EOF (SR 5)* -End-of-file (EOF) is sensed and may be encountered for those functions which come under the heading of read status function, i.e., space, read data, or read/compare and write EOF. When EOF is encountered, the tape control sets EOF = 1. MTF is also set; hence, an interrupt* occurs and the EOF status bit may be checked.

*EOT (SR6) and BOT (SR2)* - End-of-tape (EOT) detection occurs during any forward command when the EOT reflective strip is sensed. When EOT is sensed, the EOT bit is set, but the function continues to completion. At this time the MTF is set (and EF is set), and an interrupt occurs.

Beginning-of-tape (BOT) detection status bit occurs only when the beginning-of-tape reflective strip is read on the transport that is selected.

When BOT detection occurs, and the unit is in reverse, the function terminates. If a tape unit is at load point when a reverse command is given, an illegal command error bit is set, causing an EF with BOT set. An interrupt then occurs.

*Illegal Command Error (SR3)* - The illegal command error bit is set under the following conditions.

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

2. A MTGO command is issued to a tape unit which is not ready, and the tape control is ready.

3. Any command which the tape control, although ready, cannot perform, for example.

   a. Write with write lock condition.

   b. Nine-channel tape and incorrect density.

   c. BOT and space reverse.

*Parity (SR4)* - Longitudinal and lateral parity checks will occur in both reading and writing. The parity bit is set for either lateral or longitudinal parity failure. A function is not interrupted, however, until MTF is set. Maintenance panel indicators are available to determine which type of parity error occurred.

*Read/Compare Error (SR7)* - When read/compare function is underway, SR7 is set to 1 for a read/compare error (see earlier section on read/compare for further details).

*Bad Tape (SR10)* - A bad tape error indicates detection of a bad spot on the tape. Bad tape is defined as three or more consecutive missing characters followed by data, within the period defined by the real shutdown delay. The error bit is set by the tape control when this occurs. MTF and interrupt do not occur until the end of the record in which the error was detected.

*Tape Rewinding (SR1)* - When a rewind command has been issued to a tape unit and the function is underway, the tape rewinding bit is set in the control. This is a transport status bit, and any selected transport which is in a high-speed rewind will cause this bit to be set.

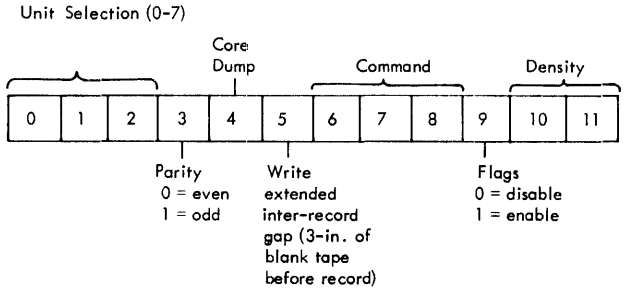
*Record Length Incorrect (SR8)* - During a read or read/compare, this bit is set when the WC overflow differs from the number of words in the record. The EF flag is set.

*Data Request Late (SR9)* - This bit can be set whenever data transmission is in progress. When the data flag causes a break cycle, the data must be transmitted before a write pulse or a real pulse occurs. If it does not, this error will occur, and data transmission will cease. The EF flag and bit 9 of the status register are set when the MTF is set.

*Error Flag (SR0)* - The error flag (EF) is set whenever any error status bit is present at the time that MTF is set. When an illegal command is given, however, the EF is set and the MTF is not set. This flag (as well as MTF) sets bit 11 in the AC if an IORS is issued.

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

## Command Register Contents



Unit Selection

| Unit | Unit Selection Bits 0 | 1 | 2 |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 |
| 2 | 0 | 1 | 0 |
| 3 | 0 | 1 | 1 |
| 4 | 1 | 0 | 0 |
| 5 | 1 | 0 | 1 |
| 6 | 1 | 1 | 0 |
| 7 | 1 | 1 | 1 |

Density Selection

| Density | Density Bits | |
|---|---|---|
| 200 bpi | 0 | 0 |
| 556 bpi | 0 | 1 |
| 800 bpi | 1 | 0 |
| 800 bpi 9 channel | 1 | 1 |

*Command Selection -*

| Command | Bits 6 | 7 | 8 |
|---|---|---|---|
| NO OP | 0 | 0 | 0 |
| Rewind | 0 | 0 | 1 |
| Read | 0 | 1 | 0 |
| Read/Compare | 0 | 1 | 1 |
| Write | 1 | 0 | 0 |
| Write EOF | 1 | 0 | 1 |
| Space Forward | 1 | 1 | 1 |
| Space Reverse | 1 | 1 | 1 |

## Magnetic Tape Function Summary

*LEGEND:* CA = Current Address Register = $32_8$
WC = Word Count Register = $33_8$
F = Forward
R = Reverse
DS = Density Setting
PR = Parity Setting
EN = Enable Interrupt

| Function | Characteristics | Status or Error Type |
|---|---|---|
| NO OP | CA: Ignored | Illegal |
| | WC: Ignored | BOT |
| | DS: Ignored | Tape Rewinding |
| | PR: Ignored | |
| | EN: Ignored | |
| SPACE FORWARD | CA: Ignored | Illegal |
| | WC: 2s complement of number of records to skip | EOF |
| | DS: Must be set | Bad Tape |
| | EN: Must be set | MTF, BOT, EOT |
| SPACE REVERSE | CA: Ignored | Illegal |
| | WC: 2s complement of number of records to skip | EOF |
| | DS: Must be set | Bad Tape |
| | EN: Must be set | BOT, MTF |
| READ DATA | CA: Core address -1 | Illegal |
| | WC: 2s complement of number of words to be transferred | EOF |
| | DS: Must be set | Parity |
| | PR: Must be set | Bad Tape |
| | EN: Must be set | MTF |
| | | EOT |
| | | Request Late |
| | | Record Length Incorrect |
| WRITE DATA | CA: Core address -1 | Illegal |
| | WC: 2s complement of number of words to be transferred | EOF |
| | DS: Must be set | Parity |
| | PR: Must be set | MTF |
| | EN: Must be set | Bad Tape |
| | | Data Request Late |
| WRITE EOF | CA: Ignored | Illegal |
| | WC: Ignored | EOF |
| | DS: Must be set | Parity |
| | PR: Must be set | MTF |
| | EN: Must be set | Bad Tape |
| | | Data Request Late |
| READ/COMPARE | CA: Core Address -1 | Illegal |
| | WC: 2s complement of number of words to be transferred | EOF |
| | DS: Must be set | Read/Compare Error |
| | PR: Must be set | Bad Tape |
| | EN: Must be set | MTF |
| | | EOT |
| | | Data Late |
| | | Record Length Incorrect |
| REWIND | CA: Ignored | Illegal |
| | WC: Ignored | Tape Rewinding |
| | DS: Ignored | MTF |
| | PR: Ignored | BOT |
| | EN: Must be set | |

## MAGNETIC TAPE TRANSPORT, TYPE TU20 (7–CHANNEL)

The Type TU20 is a digital magnetic tape transport designed to be compatible with the Type

TC59 Magnetic Tape Control. The transport operates at a speed of 45 in./sec and has three selectable densities: 200, 556, and 800 bpi. The maximum transfer rate is 36,000 six-bit characters per second. Standard seven-channel IBM-compatible tape format is used. The specifications for the unit are as follows.

Format - NRZI. Six data bits plus one parity bit. End and loadpoint sensing compatible with IBM 729 I-VI.

Tape - Width of 0.5 in. Length of 2400 ft. (1.5 mil.). Reels are 10.5 in., IBM-compatible with file protect (write lock) ring.

Heads - Write-read gap of 0.300 in. Dynamic and static skew is less than 14 microseconds.

Tape Specifications - 45 ips speed. Start time is less than 5 msec. Rewind time for 2400 ft. is less than 3 min. Start distance is 0.080 in. (±0.035, -0.025 in.). Stop time is less than 1.5 msec. Stop distance is 0.045 in. (±0.015 in.).

Density - 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, AND RESET.

Physical Specifications - Width of 22-1/4 in., depth of 27-1/6 in., height of 69-1/8 in., weight of 600 lb.

Read (Read/Compare) Shutdown Delay - 3.6 msec.

Write Shutdown Delay - approximately 4.5 msec.

## MAGNETIC TAPE TRANSPORT, TYPE TU20A (9–CHANNEL)

The Type TU20A is a digital magnetic tape transport designed to be compatible with the Type TC59 Magnetic Tape Control. The transport operates at a speed of 45 in./sec and has three selectable densities: 200, 556, and 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 of 0.5 in. Length of 2400 ft., (1.5 mil.). Reels are 10.5 in., IBM-compatible, with file protect (write lock) ring.

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

Tape Specifications - 45 ips speed. Start time is less than 5 msec. Rewind time for 2400 ft. is less than 5 min. Start distance is 0.080 in. (+0.035, -0.025 in.). Stop time is less than 1.5 msec. Stop distance is 0.045 in. (±0.015 in.).

Density - 200, 556, and 800 bpi. Maximum transfer rate is 36 kHz.

Transport Mechanism - Pinch roller drive; vacuum column tension.

Controls - ON/OFF, ON LINE, OFF LINE, FORWARD, REVERSE, REWIND. LOAD, RESET.

Physical Specifications - Width of 22-1/4 in., depth of 27-1/6 in., height of 69-1/8 in., weight of 600 lb.

Read (Read/Compare) Shutdown Delay - 3.6 msec.

Write Shutdown Delay - approximately 4.5 msec.

# CHAPTER 6
# ADDRESSING

## GENERAL

The PDP-9/L can directly address up to 8192 locations (a location consists of an 18-bit word register) and indirectly address up to 32,768 locations in system core memory. Locations are addressed, octally, as 00000 through 77777 with the following allocations per separate memory modules or banks.

Memory bank 0 – 00000 through 17777
Memory bank 1 – 20000 through 37777
Memory bank 2 – 40000 through 57777
Memory bank 3 – 60000 through 77777

4096 words of bank 0 are included with the basic PDP-9/L configuration; other banks are appended with expansion of the system.

PDP-9/L also offers autoindexing and extend mode addressing. Eight explicitly addressed locations (10-17) in the basic memory module provide efficient indexed addressing of up to 32,768 memory locations. An indirect reference to these locations increments the existing contents by one and takes the result as the effective address for the operand. Extend mode addressing, implemented by the optional memory extension control, sets up the required parameters for addressing across the memory module boundaries of an expanded PDP-9/L system.

## DIRECT ADDRESSING

The instruction word format for PDP-9/L memory reference instructions includes an operation code field of four bits, and indirect address indicator field of one bit, and a 13-bit address field, as shown below.

For direct addressing, the indirect address indicator (bit 4) is 0. In this case, the machine action defined by the operation code field considers the contents of the address field as being the "effective" address for the instruction. This address specifies a location for direct retrieval, entering, or modification of the contents. For example, the directly addressed instruction

```
LAC 100
```

directs the central processor unit (CPU) to load its accumulator register (AC) with the 18-bit contents of memory location 100.

The 13-bit address field allows direct addressing of any location of the 8192-word basic memory module.

## INDIRECT ADDRESSING

For indirect addressing, the indirect address indicator (bit 4) is 1. In this case, the contents of the address field reflect not the effective address but the address of the location at which the effective address is expressed. For example, the indirectly addressed instruction

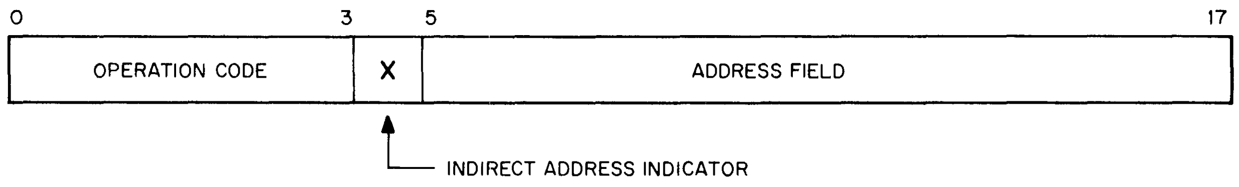
```
LAC I 100
```

(where I is the PDP-9 symbolic representation for indirect addressing) directs the CPU to load the AC, not with the contents of location 100, but the contents of the location that is addressed by the contents of location 100. If location 100 contained 000077, the instruction

```
LAC 100
```

would cause the quantity 000077 to be loaded into the AC. The instruction

```
LAC I 100
```



however, would not enter 000077 in the AC but the contents of location 00077 (considering the address field limitation of 15 bits for addressing up to 32,768 locations).

Indirect addressing adds one machine cycle to an instruction's execution time. During this interval, execution is deferred while the effective address is established. An instruction can have only one level of indirect addressing.

## AUTOINDEXING

Eight locations, 00010 through 00017, of memory bank 0 serve as auto-index registers. Indirect addressing of an auto-index register causes its contents to be automatically incremented by one and then taken as the effective address for the instruction. Thus, addressing of sequential memory locations can be easily achieved by loading an auto-index register with the initial address minus one, and then indirectly addressing the auto-index register until the required operation is completed.

The incrementation of an auto-index register's contents does not add to the instruction execution time. Autoindexing occurs only upon an indirect address reference of an auto-index register. When directly addressed, an auto-index register functions in the same manner as all other memory locations.

Assume that four memory locations are initialized as follows:

| *Location* | *Contents* |
|---|---|
| 0010 | 100 |
| 0040 | 050 |
| 100 | 040 |
| 101 | 041 |

The following four instructions to load the accumulator illustrate, by comparison, direct, indirect, and autoindexed addressing.

```
LAC       100    Places the number 40 into the AC.
LAC   I   100    Places the number 50 into the AC.
LAC       010    Places the number 100 into the
                 AC.
LAC   I   010    By autoindexing, the contents of
                 location 10 become 101, then
                 the number 41 is placed into the
                 AC.
```

Autoindexing can be used to process a block of numbers without the need for address arithmetic. The following three examples demonstrate typical programming techniques (the mnemonics refer to PDP-9 instructions of the memory referencing and augmented classes. (Refer to chapter 7, Instructions, for descriptions of their actions.)

*Example 1:* Sum of a series of numbers;

$$Y = \sum_{i=1}^{N} Xi$$

```
10/        FIRST -1     /FIRST WORD'S LOCATION -1
100/
COUNT,     -N + 1       /NUMBER OF ITERATIONS
                        /(2s COMPLEMENT)
ENTRY,     CLA!CLL      /CLEAR AC AND LINK
LOOP,      ADD I 10     /PARTIAL SUM
           ISZ COUNT    /TEST FOR COMPLETION
           JMP LOOP     /MORE IN TABLE, GO BACK
           HALT         /SUM IN AC
```

*Example 2:* Ci = Ai + Bi; i = 1, 2, ...N

Three autoindexing locations are used to simplify the addressing.

```
10/         L(A) -1       /(THE FIRST LOCATION
                          /OF THE A ARRAY)-1
11/         L(B) -1       /(THE FIRST LOCATION
                          /OF THE B ARRAY)-1
12/         L(C) -1       /(THE FIRST LOCATION
                          /OF THE C ARRAY)-1
100/
COUNT,      -N + 1        /(NUMBER OF ITERA-
                          /TIONS (2s COMPLE-
                          /MENT)
BEGIN,      CLA!CLL       /CLEAR AC AND LINK
LOOP,       LAC I 10      /GET ADDEND
            ADD I 11      /FORM SUM
            DAC I 12      /STORE SUM
            ISZ COUNT     /TEST FOR COMPLETION
            JMP LOOP      /MORE IN TABLE, GO
                          /BACK
```

*Example 3:* Cj = Cj + K; j = 1, 2, ...N

This example demonstrates the modification of a list of numbers by adding a constant to each of them. In this case, the autoindexing memory register contains an instruction rather than just an address.

```
10/        DAC FIRST -1 /DEPOSIT INTO (FIRST LOCATION
                        /IN TABLE) -1
100/
COUNT,     -N + 1       /TWOS COMPLEMENT OF NUMBER
                        /OF WORDS IN TABLE
CONST/,K                /THE CONSTANT
GO,        CLA!CLL      /CLEAR AC AND LINK
LOOP,      LAC I 10     /PICK UP INITIAL VALUE FROM
                        /TABLE
           ADD CONST    /ADD CONSTANT
           XCT 10       /REPLACE IN TABLE
           ISZ COUNT    /TEST FOR COMPLETION
           JMP LOOP     /MORE IN TABLE, GO BACK
           HALT
```

All indirectly addressed reference to locations 10-17 refer to the absolute locations of memory bank 0, regardless of the memory bank location of the instruction making the reference or the condition of the extend mode. A directly addressed reference to one of these locations refers to the absolute location, only if the instruction making the reference is also present in a location of memory bank 0. If the instruction is located in any bank other than memory bank 0, the direct reference is made to the relative location of that memory bank. Thus, the use of the instruction

```
    DAC 10
```

to load auto-index register 10 with a specified contents for indexing purposes would be effective only if the loading instruction is in memory bank 0 after the program was loaded. Users instead are advised to adhere to the following procedure for loading auto-index registers from any memory bank other than 0. The extend mode must be enabled.

```
        DAC I A
           .
           .
    A,10   .
```

This indirect loading sequence accomplishes the loading regardless of the memory bank storage of the DAC I A instruction.

## EXTEND MODE ADDRESSING

Addition of the optional memory extension control, Type KG09A, required for expansion of PDP-9/L core memory (up to 32,768 words in banks of 8192 words), implements the extend mode for addressing any memory location in the expanded system. The mode is enabled and disabled by programmed instructions and it can be entered through use of the EXD switch on the control console (refer to chapter 10, Controls and Indicators).

The memory extension control adds four instructions to the PDP-9/L order code. (Refer to table 6-1.)

Execution of the EEM instruction enables the extend mode, and LEM disables the mode. If the mode is enabled, execution of the SEM instruction causes the PC to be incremented by one to effect a skip of the next instruction in sequence.

While the extend mode is disabled, all instructions to be executed during this interim, and their operands, must be stored in the same memory bank, and this bank will be addressed by the Extended Program Counter (EPC). The EPC consists of standard 13-bit PC and the 2-bit extension, added by the option to the high-order end.

With the exception of the PI and API traps to bank 0, it is impossible to access any instruction or operand from other memory banks with the extend mode disabled. Regardless of the extend mode status, however, an instruction in another bank can always indirectly address auto-index registers (locations 10 through 17) of bank 0 to make use of that facility. When the extend mode is disabled, the auto-index register is used as an address pointer to the memory bank from which the reference was made (i.e., the 2-bit extension of the PC is unchanged and only 13 bits of address in the auto register are taken). To load the contents of an auto-index register from any memory bank except 0, the extend mode must be enabled.

Program interrupts and API interrupts trap to their proper locations in bank 0. In trapping to location 00000 of bank 0, a program interrupt stores the existing status of the extend mode (on or off) and then disables the mode. Through use of the DBR (debreak and restore) instruction, the interrupt-accessed subroutine can restore the mode to its interrupted state at the end of the routine. (For a description of the DBR instruction, refer to the API discussion in chapter 9.) An API request transfers program control to the appropriate channel entry in memory (always in bank 0). The instruction present at the channel location should be a JMS I Y, which stores the EPC and the status of the extend mode. (The JMS should be indirect to permit reloca-

TABLE 6-1 MEMORY EXTENSION CONTROL INSTRUCTIONS

| Mnemonic Symbol | Octal Code | Operation |
|---|---|---|
| SEM | 707701 | Skip next instruction if extend mode is active |
| EEM | 707702 | Enter extend mode |
| LEM | 707704 | Leave extend mode |
| ERIR | 707742 | Enter extend mode interrupt restore |

tion to any bank.) The mode status is not disturbed.

While the extend mode is enabled (the normal state), any location in the memory system may be indirectly addressed through extension of the effective address from the normal 13 bits (bits 5 through 17) to 15 bits (bits 3 through 17). Bits 3 and 4 indicate the memory bank that is to be addressed, and bits 5 through 17 address a location in that bank. The PC extension indicates by its contents which of the memory banks is currently addressed by the PC. Because the extension cannot count, the PC functions as a modulo 8192 counter and therefore does not increment across memory bank boundaries (e.g., the location addressed after 17777 is 00000, not 20000). To effect a change in memory banks, the program must include a jump instruction with indirect address (JMP I Y, or JMS I Y if the exit point is to be preserved for subsequent return). Execution of this instruction enters a 15-bit address in the extended program counter to select a new memory bank and the starting location in this bank.

When extend mode is enabled an effective address for an indirectly addressed location must be a 15-bit address. This requirement prohibits the use of an instruction word (such as LAW Y) as an indirect address when the extend mode is enabled.

Note that execution of a CAL instruction results in the addressing of location 20 of memory bank 0, when the extend mode is enabled; and location 20 (relative) of the currently addressed memory bank, when the extend mode is disabled.

XCT instruction always function as if the referenced instructions were fetched. Thus, XCT I reference of a skip instruction in another memory bank effects a skip of the instruction immediately following the XCT I instruction, if the skipping condition is satisfied (i.e., the EPC is incremented by one). Similarly, XCT I reference of a JMS or CAL instruction in another memory bank effects the appropriate storing of the EPC contents, which in turn represent the address of the location following the XCT I instructions and not the location following the referenced instruction.

## RESERVED ADDRESSES

Programs prepared for the PDP-9/L should not make use of locations addressed 00000 through 00077 for data or instruction storage as they are reserved for the pusposes listed in table 6-2.

TABLE 6-2 RESERVED ADDRESSES

| Address | Purpose |
|---|---|
| 0 | Stores the contents of the extended PC, link, extend mode status, and memory protection status during a program interrupt. |
| 1 | Stores the first instruction to be executed following a program interrupt. |
| 2-6 | Reserved for PDP-9 system programs. |
| 7 | Stores real-time clock count. |
| 10-17 | Autoindex registers. |
| 20 | Stores the contents of the extended PC, link, extend mode status, and memory protection status upon execution of a CAL instruction. |
| 21 | Stores the first instruction to be executed following a CAL instruction. |
| 22-27 | Reserved for PDP-9 system programs. |
| 30-37 | Four pairs of word counter-current address registers for use with data channels 0, 1, 2, and 3. |
| 40-77 | Store unique entry instructions for each $32_{10}$ automatic priority interrupt channels. |

# CHAPTER 7
# INSTRUCTIONS

## GENERAL

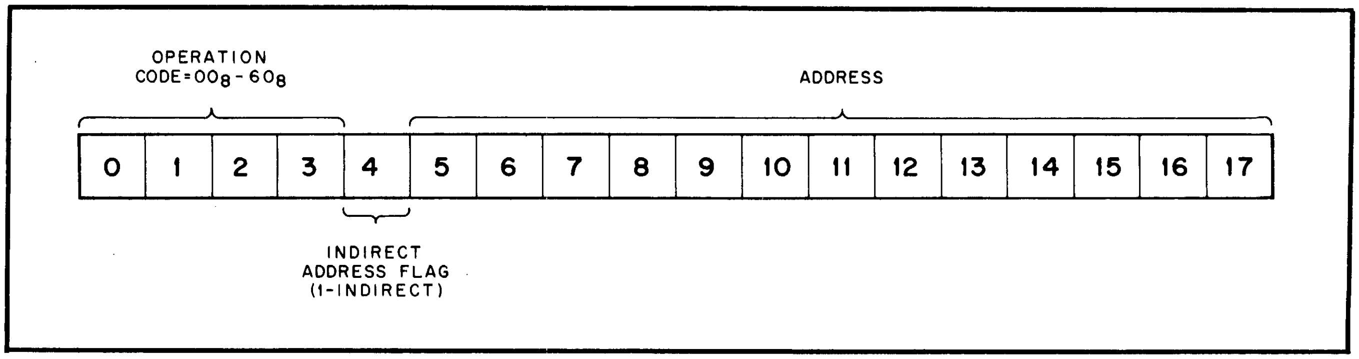
The PDP-9/L instruction set is subdivided into two groups: those which address system core memory and those which do not. An instruction of the former group addresses, either directly or indirectly, a location in memory for the purpose of retrieving, entering, or modifying the contents. These instructions are known as "memory referencing" instructions. The instructions which do not address memory are known as "augmented" instructions in that the entire 18-bit instruction word serves as an expanded operation code to specify a specific action or actions to be executed. The augmented instruction group has three subclasses: operate (which provides for skip, rotate, clear, complement, etc. operations involving the accumulator and/or the link register); IOT (input/output transfer of data, status, and command information between the central processor and peripheral devices); and EAE (optional extended arithmetic element implementation of hardware multiply, divide, shift, normalize, etc.).

## MEMORY REFERENCE INSTRUCTION FORMAT

The memory reference instruction word (figure 7-1) consists of three parts: operation code, indirect address bit, and address. The operation code, bits 0 through 3, indicates which one of PDP-9/L's 13 memory reference instructions is specified. The indirect address bit indicates whether the 13-bit address (bits 5 through 17) is to be taken as the direct address (bit 4 is 0) or indirect address (bit 4 is 1). If direct addressing is indicated, the addressed memory location is taken to contain the required operand. If indirect addressing is called for, the contents of the addressed memory location are taken not as the operand but the address at which the operand is located. In either case, the address specifying the memory location that contains the operand is taken as the "effective address" for the instruction.

## AUGMENTED INSTRUCTION FORMAT

The augmented instruction word (figure 7-2) has two parts: an operation code and an instruction code. The operation code, bits 0 through 3, denotes the type of instruction specified by the instruction code. Operation codes for the three types are: $70_8$ for input/output transfer (IOT), $74_8$ for operate, and $64_8$ for the optional extend arithmetic element (EAE). The instruction code, bits 4 through 17, specifies the action to be executed. An important and useful feature of the PDP-9/L augmented instruction is its microprogramming capability. Multiple instruction codes having the same operation code can be combined to form one instruction word. Execution of all the microprogrammed functions occurs during the time allocated to the type of instruction (operate instructions require one machine cycle, IOTs require four cycles, and EAEs require two cycles plus a variable time interval to complete their functions). Thus, microprogramming decreases program running time, lessens the number of instruction words required, and simplifies programming efforts.



OPERATION CODE=$00_8$-$60_8$
ADDRESS
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
INDIRECT ADDRESS FLAG (1-INDIRECT)

Figure 7-1. Memory Reference Instruction Format

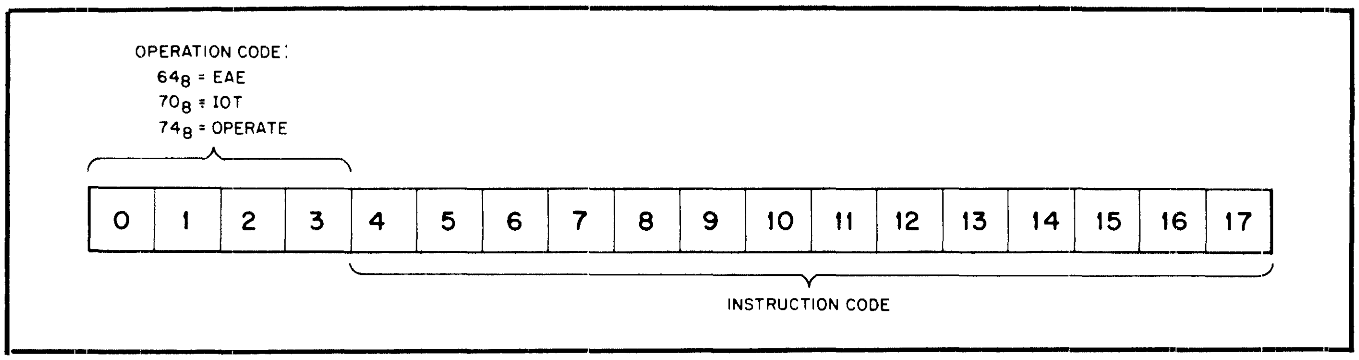


Figure 7-2. Augmented Instruction Format

MEMORY REFERENCE INSTRUCTIONS

The following information applies to each memory reference instruction:

1. Instruction time is expressed in machine cycle units, where one cycle equals 1.0 microsecond.

2. An instruction with indirect addressing (bit 4 is 1) requires one additional machine cycle. PDP-9/L memory referencing instructions can take only one level of indirect addressing; i.e., the indirectly addressed memory location must contain the address of the operand.

3. The term "effective address" applies to the address that specifies the memory location containing the operand for the instruction.

4. Numerical memory location addresses are expressed octally.

5. Subscript notations identify specific bit positions of the respectively identified register or location. Numerical subscripts are expressed decimally.

6. Except for the CAL instruction, all memory referencing instructions must include the address (direct or indirect) of an operand. The CAL instruction takes the hardware-fixed address of 20; it ignores the address field in the instruction word.

7. In the symbolic representations for the JMP, JMS, and CAL instructions, the quotation marks enclosing "Y" or "21" for CAL, indicate that Y, or 21, rather than their contents, enters the PC, as shown.

Mnemonic: LAC (Load the Accumulator)
Octal Code: 20
Time: 2 cycles
Operation: The contents of the effectively addressed memory register, Y, are read into the AC. The contents of Y are unchanged; the previous contents of the AC are lost.
Symbolic: Y AC

Mnemonic: DAC (Deposit the Accumulator)
Octal Code: 04
Time: 2 cycles
Operation: The contents of the AC are deposited (written) into the effectively addressed memory register, Y. The contents of the AC are unchanged; the previous contents of Y are lost.
Symbolic: $AC \longrightarrow Y$

Mnemonic: DZM (Deposit Zero in Memory)
Octal Code: 14
Time: 2 cycles
Operation: An all-zeros data word is deposited (written) in the effectively addressed memory register, Y. The previous contents of Y are lost; the contents of the AC are unchanged.
Symbolic: $0 \longrightarrow Y$

Mnemonic: ADD (Add, 1s Complement)
Octal Code: 30
Time: 2 cycles
Operation: The contents of the effectively addressed memory register, Y, are added to the contents of the AC, following the rules of 1s complement arithmetic (end around carry)*. The result is left in the AC. An arithmetic overflow sets the link to the binary 1 state. The contents of Y are unchanged; the previous contents of the AC are lost. The previous content of the link is lost. Overflow occurs if the magnitude (absolute) of the algebraic sum of the operands exceeds $2^{17}$-1; i.e., if the operands were of like sign and the result is signed differently, overflow has occured to set the link. Overflow cannot occur if the operands are of different sign.
Symbolic: Y + AC AC

LV Overflow $\longrightarrow$ L

Mnemonic: TAD (Add, 2s Complement)
Octal Code: 34
Time: 2 cycles
Operation: The contents of the effectively addressed memory register, Y, are added to the contents of the AC, following the rules of 2s complement arithmetic*. The result is left in the AC. An arithmetic carry from $AC_0$ complements the link. The contents of Y are unchanged; the previous contents of the AC are lost.
Symbolic: $Y + (L,AC) \longrightarrow (L,AC)$

Mnemonic: AND (Boolean AND)
Octal Code: 50
Time: 2 cycles
Operation: The contents of the effectively addressed memory register, Y, are logically ANDed with the contents of the AC on a bit-by-bit basis. The result is left in the AC. If corresponding Y and AC bits (i) are in the 1 state, the AC bit remains a 1; otherwise the AC bit is cleared to the 0 state. The contents of Y are unchanged; the previous contents of the AC are lost.
Symbolic: $Y \wedge AC \longrightarrow AC$

| AND | | $AC_i$ | |
|---|---|---|---|
| | | 0 | 1 |
| $Y_i$ | 0 | 0 | 0 |
| | 1 | 0 | 1 |

Mnemonic: XOR (Boolean Exclusive OR)
Octal Code: 24
Time: 2 cycles
Operation: The contents of the effectively addressed memory register, Y, are exclusively ORed with the contents of the AC on a bit-by-bit basis. The result is left in the AC. If corresponding Y and AC bits (i) are in the same binary state (i.e., 1 or 0), the AC bit is cleared to the 0 state. If the corresponding bits are not in the same binary state, the AC bit is set to the 1 state. The contents of Y are unchanged; the previous contents of the AC are lost.
Symbolic: $Y \veebar AC \quad AC$

| Exclusive OR | | $AC_i$ | |
|---|---|---|---|
| | | 0 | 1 |
| $Y_i$ | 0 | 0 | 1 |
| | 1 | 1 | 0 |

*Refer to chapter 8 for discussion of 1s and 2s complement notations and arithmetic.

Mnemonic: SAD (Skip if AC Differs)
Octal Code: 54
Time: 2 cycles
Operation: The contents of the effectively addressed memory register, Y, are compared with the contents of the AC. If they differ, the PC is incremented by one to effect skipping the next instruction. If they have the same binary quantity, the next instruction is executed. The contents of Y and the contents of the AC are unchanged.
Symbolic: If $Y \neq AC$, $PC + 1 \longrightarrow PC$

Mnemonic: ISZ (Increment and Skip if Zero)
Octal Code: 44
Time: 2 cycles
Operation: The contents of the effectively addressed memory register, Y, are incremented by one (in 2s complement arithmetic) and tested. If Y now contains an all-zero word, the PC is incremented by one to effect skipping the next instruction. If the contents of Y, after being incremented, are other than all zeros, the next instruction is executed. The previous contents of Y are lost; the contents of the AC are unchanged.
Symbolic: If $Y + 1 = 0$, $PC + 1 \longrightarrow PC$
$Y + 1 \longrightarrow Y$.

Mnemonic: JMP (Unconditional Jump)
Octal Code: 60
Time: 1 cycle
Operation: The next instruction is read from the contents of the effectively addressed memory register, Y, thereby breaking the existing program sequence and starting a new sequence from Y. The previous contents of the PC are lost when the effective address enters the PC. The contents of the AC are unchanged.
Symbolic: "Y" (bits 5-17) $\longrightarrow$ PC

Mnemonic: JMS (Jump to Subroutine)
Octal Code: 10
Time: 2 cycles
Operation: The contents of the PC and of the link, and the status (on or off) of the extend mode and of the memory protect mode are deposited in the effectively addressed memory register, Y. The next instruction is read from the contents of memory register Y + 1, breaking the previous program sequence and starting a new sequence from Y + 1. The contents of the AC are unchanged.

Symbolic: $L \longrightarrow Y_0$
$EM \longrightarrow Y_1$
$MP \longrightarrow Y_2$
$PC \longrightarrow Y_{3\text{-}17}$
"Y" + 1 (bits 5-17) $\longrightarrow$ PC

Mnemonic: CAL (Call (jump to) Subroutine)
Octal Code: 00
Time: 2 cycles
Operation: The CAL instruction is the equivalent of a JMS 20 instruction. The contents of the PC and of the link, and the status (on or off) of the extend memory mode and of the memory protect mode are deposited in memory register 20. The next instruction is read from the contents of memory register 21, breaking the previous program sequence and starting a new sequence from 21. The contents of the AC are unchanged. If the API option is present and enabled, priority level 4 will be activated after the execution of a CAL instruction.
Symbolic: $L \longrightarrow 20_0$
$EM \longrightarrow 20_1$
$MP \longrightarrow 20_2$
$PC \longrightarrow 20_{3\text{-}17}$
$\text{“21”} \longrightarrow PC$

Mnemonic: XCT (Execute the Instruction at Y)
Octal Code: 40
Time: 1 cycle plus time of instruction at Y
Operation: The computer executes the instruction located at the effectively addressed memory register, Y. The contents of the PC are unchanged unless Y contains a JMS, CAL, JMP, or skip instruction, each of which changes the contents of the PC to alter the program sequence. (XCT can be thought of as a single-instruction subroutine causing a quasi-jump to Y, execution of the instruction specified there, and return to the program sequence (i.e., execution of the instruction following XCT) if the instruction at Y has not changed the PC.)
Symbolic: $Y \longrightarrow IR$

## OPERATE INSTRUCTIONS

Operate instructions have an operation code of $74_8$ and are used to sense and alter the contents of the AC and link. Typical functions are: conditional or unconditional skips and complementing, setting, clearing, or rotating the contents of the two registers jointly or independently. A HLT instruction is included. Operate instructions are fetched and executed in one machine cycle; the actions are specified by the microprogramming of the instruction code. Each of the 14 bits (figure 7-2) can effect a unique response; hence, these bits are microinstructions to the computer. The important feature of the operate instruction is its microprogramming capability because two or three microinstructions can be combined to form one instruction word and, therefore, they can be executed during one cycle. Microinstructions that logically conflict and occur during the same event time should not be microprogrammed. Figure 7-3 lists the sequential event times and the microinstructions associated with each (the instructions are indicated by their mnemonics).

The nature of rotate operations precludes the microprogramming of other microinstructions during the same event times.

When noninverted skip actions are microprogrammed (bit 8 is 0), the conditions to be met are inclusively ORed. For example: if SZA (741200) and SZL (741400) are specified in a microprogrammed instruction (741600), the skip occurs only if both conditions are present (the contents of the AC are other than 0, the content of the link is 0).

Mnemonic: NOP (No Operation)
Octal Code: 740000
Time: 1 cycle
Operation: The program is delayed for one cycle before the next instruction is fetched. As a “do nothing” cycle, NOP can be used to synchronize program timing to peripheral timing by delaying execution of an instruction until the appropriate time.
Symbolic: Not applicable

Mnemonic: CMA (Complement Accumulator)
Octal Code: 740001
Time: 1 cycle
Operation: Each bit of the AC is set or cleared to the inverse of its current state. The previous contents of the AC are lost.
Symbolic: $\overline{AC} \longrightarrow AC$

Mnemonic: CML (Complement Link)
Octal Code: 740002
Time: 1 cycle
Operation: The link is set or cleared to the inverse of its current state. Its previous content is lost.
Symbolic: $\overline{L} \longrightarrow L$

Mnemonic: OAS (Inclusive OR ACCUMULATOR Switches)
Octal Code: 740004
Time: 1 cycle
Operation: The word set up by manual positioning of the ACCUMULATOR switches is inclusively ORed with the contents of the AC on a bit-by-bit basis. The result is left in the AC. If corresponding AC and AC switch bits (i) are in the binary 0 state, the AC bit remains 0. If either or both of the corresponding bits (i) are in the binary 1 state, the AC bit is set to 1. The previous contents of the AC are lost. The switch settings are not affected.
Symbolic: $AC \vee AC\ Switch \longrightarrow AC$

Operation Code = $74_8$

| CLA | CLL | Additional Rotate | 0 = OR of / 1 = AND of | SNL / SZL | SZA / SNA | SMA / SPA | HLT | Bit 7 = 0: RAR RAL / Bit 7 = 1: RTR RTL | | OAS | CML | CMA |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |

| | | 0-2 | 3-5 | 6-8 | 9-11 | 12-14 | 15-17 | Event Time | |
|---|---|---|---|---|---|---|---|---|---|
| | OPR / NOP | 7 | 4 | 0 | 0 | 0 | 0 | – | |
| | CMA | 7 | 4 | 0 | 0 | 0 | 1 | 3 | No other operation |
| | CML | 7 | 4 | 0 | 0 | 0 | 2 | 3 | may take place at |
| | OAS | 7 | 4 | 0 | 0 | 0 | 4 | 5 | the same event time |
| | RAL | 7 | 4 | 0 | 0 | 1 | 0 | 4 | as rotates. |
| | RAR | 7 | 4 | 0 | 0 | 2 | 0 | 4 | RAR, RAL may not be |
| | *HLT / *XX | 7 | 4 | 0 | 0 | 4 | 0 | 6 | combined with: |
| | SMA | 7 | 4 | 0 | 1 | 0 | 0 | 1 | OAS, CML, CMA |
| | SZA | 7 | 4 | 0 | 2 | 0 | 0 | 1 | |
| | SNL / SML | 7 | 4 | 0 | 4 | 0 | 0 | 1 | RTR, RTL may not be |
| | SKP | 7 | 4 | 1 | 0 | 0 | 0 | 1 | combined with: |
| | SPA | 7 | 4 | 1 | 1 | 0 | 0 | 1 | CLA, CLL, OAS, CML, CMA. |
| | SNA | 7 | 4 | 1 | 2 | 0 | 0 | 1 | |
| | SZL / SPL | 7 | 4 | 1 | 4 | 0 | 0 | 1 | |
| | RTL | 7 | 4 | 2 | 0 | 1 | 0 | 4 | |
| | RTR | 7 | 4 | 2 | 0 | 2 | 0 | 4 | |
| | CLL | 7 | 4 | 4 | 0 | 0 | 0 | 2 | |
| (CLL-CML) | STL / CCL | 7 | 4 | 4 | 0 | 0 | 2 | 2,3 | |
| (CLL-RAL) | RCL | 7 | 4 | 4 | 0 | 1 | 0 | 2,4 | |
| (CLL-RAR) | RCR | 7 | 4 | 4 | 0 | 2 | 0 | 2,4 | |
| | CLA | 7 | 5 | 0 | 0 | 0 | 0 | 2 | |
| (CLA-CMA) | CLC | 7 | 5 | 0 | 0 | 0 | 1 | 2,3 | |
| (CLA-OAS) | LAS / LAT | 7 | 5 | 0 | 0 | 0 | 4 | 2,5 | |
| (CLA-RAL) | GLK | 7 | 5 | 0 | 0 | 1 | 0 | 2,4 | |

*Programming Note: The PDP-9/L Symbolic Assembler accepts either HLT or XX as a valid mnemonic for the operate class instruction to stop program execution. The latter facilitates visual scanning of a program listing to determine the occurrence of program halts.

Figure 7-3 Operate Instructions

| Inclusive OR | | $ACS_i$ | |
|---|---|---|---|
| | 0 | 0 | 1 |
| $AC_i$ | 1 | 1 | 1 |

Mnemonic: RAL (Rotate AC and Link Left)
Octal Code: 740010
Time: 1 cycle
Operation: The contents of the AC and the link are rotated one bit position to the left with $AC_0$ entering the link and the link entering $AC_{17}$.
Symbolic: $AC_i \longrightarrow AC_{i-1}$; i = 1, 17
$AC_0 \longrightarrow L$
$L \longrightarrow AC_{17}$

Mnemonic: RAR (Rotate AC and Link Right)
Octal Code: 740020
Time: 1 cycle
Operation: The contents of the AC and the link are rotated one bit position to the right with $AC_{17}$ entering the link and the link entering $AC_0$
Symbolic: $AC_i \longrightarrow AC_{i+1}$; i = 0, 16
$AC_{17} \longrightarrow L$
$L \longrightarrow AC_0$

Mnemonic: HLT (Halt Program) (see programming footnote, figure 7-3)
Octal Code: 740040
Time: 1 cycle
Operation: Program execution stops at completion of the current machine cycle. The PGRM STOP indicator is lighted.
Symbolic: $0 \longrightarrow$ RUN flip-flop

Mnemonic: SMA (Skip on Minus Accumulator)
Octal Code: 740100
Time: 1 cycle
Operation: Test the contents of the sign bit, $AC_0$, of the data word in the AC. If the bit is in the 1 state, the contents of the PC are incremented by one to effect skipping the next instruction. If $AC_0$ is in the 0 state, the next instruction is executed. The contents of the AC are unchanged.
Symbolic: If $AC_0 = 1$, $PC + 1 \longrightarrow PC$

Mnemonic: SZA (Skip on Zero Accumulator)
Octal Code: 740200
Time: 1 cycle
Operation: Test the contents of the word in the AC. If all bits are 0s, the quantity is taken to be zero (2s complement notation), and the contents of the PC are incremented by one to effect skipping the next instruction. If any bit is in the 1 state, the next instruction is executed. The contents of the AC are unchanged.
Symbolic: If AC = 0, $PC + 1 \longrightarrow PC$

Mnemonic: SNL (Skip on Non-zero Link)
Octal Code: 740400
Time: 1 cycle
Operation: Test the content of the link. If the link is in the 1 state, the contents of the PC are incremented by one to effect skipping the next instruction. If the link is a 0, the next instruction is executed. The content of the link is unchanged.
Symbolic: If L = 1, $PC + 1 \longrightarrow PC$

Mnemonic: SKP (Unconditional Skip)
Octal Code: 741000
Time: 1 cycle
Operation: The contents of the PC are incremented by one to effect an unconditional skip of the next instruction.
Symbolic: $PC + 1 \longrightarrow PC$

Mnemonic: SPA (Skip on Positive Accumulator)
Octal Code: 741100
Time: 1 cycle
Operation: Test the contents of the sign bit, $AC_0$, for a data word in the AC. If the bit is in the 0 state, the quantity in the AC is taken to be positive. Therefore, the contents of the PC are incremented by one to effect skipping the next instruction. If the bit is in the 1 state, the next instruction is executed. The contents of the AC are unchanged.
Symbolic: If $AC_0 = 0$, $PC + 1 \longrightarrow PC$

Mnemonic: SNA (Skip on Non-zero Accumulator)
Octal Code: 741200
Time: 1 cycle
Operation: Test the contents of the data word in the AC. If any bit is in the 1 state, the quantity is not equal to zero (2s complement notation only), and the contents of the PC are incremented by one to effect skipping of the next instruction. If all bits are in the 0 state, the quantity is zero and the next instruction is executed. The contents of the AC are unchanged.
Symbolic: If $AC \neq 0$, $PC + 1 \longrightarrow PC$

Mnemonic: SZL (Skip on Zero Link)
Octal Code: 741400
Time: 1 cycle
Operation: Test the contents of the link. If the link is in the 0 state, the contents of the PC are incremented by one to effect skipping the next instruction. If the link is a 1, the next instruction is executed. The content of the link is unchanged.
Symbolic: If L = 0, $PC + 1 \longrightarrow PC$

Mnemonic: RTL (Rotate AC and Link Two Left)
Octal Code: 742010
Time: 1 cycle
Operation: The contents of the AC and the link are rotated two bit positions to the left with $AC_0$ entering $AC_{17}$, $AC_1$ entering the link, and the link entering $AC_{16}$.
Symbolic: $AC_i \longrightarrow AC_{i-2}$; i = 2, 17
$L \longrightarrow AC_{16}$
$AC_0 \longrightarrow AC_{17}$
$AC_1 \longrightarrow L$

Mnemonic: RTR (Rotate AC and Link Two Right)
Octal Code: 742020
Time: 1 cycle
Operation: The contents of the AC and the link are rotated two bit positions to the right with the link entering $AC_1$, $AC_{17}$ entering $AC_0$, and $AC_{16}$ entering the link.

Mnemonic: CLL (Clear the Link)
Octal Code: 744000
Time: 1 cycle
Operation: The content of the link is cleared to the 0 state.
Symbolic: $0 \longrightarrow L$

Mnemonic: STL (Set the Link)
Octal Code: 744002
Time: 1 cycle
Operation: A microcoded instruction equivalent to CLL+CML. The link is first cleared to binary 0; it is then complemented to binary 1.
Symbolic: $1 \longrightarrow L$

Mnemonic: RCL (Clear Link, Then Rotate AC and L Left)
Octal Code: 744010
Time: 1 cycle
Operation: A microcoded instruction equivalent to CLL+RAL. The link is first cleared to 0; then the contents of the AC and the link are rotated one bit position to the left.
Symbolic: $AC_i \longrightarrow AC_{i-1}$; i=1, 17
$AC_0 \longrightarrow L$
$0 \longrightarrow AC_{17}$

Mnemonic: RCR (Clear Link, Then Rotate AC and L Right)
Octal Code: 744020
Time: 1 cycle
Operation: A microcoded instruction equivalent to CLL+RAR. The link is first cleared to 0; then the contents of the AC and the link are rotated one bit position to the right.
Symbolic: $AC_i \longrightarrow AC_{i+1}$; i=0, 16
$AC_{17} \longrightarrow L$
$0 \longrightarrow AC_0$

Mnemonic: CLA (Clear the Accumulator)
Octal Code: 750000
Time: 1 cycle
Operation: Each bit of the AC is cleared to 0. The previous contents are lost.
Symbolic: $0 \longrightarrow AC$

Mnemonic: CLC (Clear and Complement Accumulator)
Octal Code: 750001
Time: 1 cycle
Operation: A microcoded instruction equivalent to CLL+CMA. Each bit of the AC is cleared to 0. Then each bit is set to 1. The previous contents of the AC are lost.
Symbolic: $777777 \longrightarrow AC$

Mnemonic: LAS (Load AC from ACCUMULATOR Switches)
Octal Code: 750004
Time: 1 cycle
Operation: A microcoded instruction equivalent to CLA+OAS. Each bit of the AC is cleared to 0. Then the word set up by manual positioning of the ACCUMULATOR switches is entered in the AC. The previous contents of the the AC are lost. The switch settings are not affected.
Symbolic: $ACS \longrightarrow AC$

Mnemonic: GLK (Get the Link)
Octal Code: 750010
Time: 1 cycle
Operation: A microcoded instruction equivalent to CLA+RAL. Each bit of the AC is cleared to 0. Then the contents of the AC and the link are rotated one bit position left with the link contents entering $AC_{17}$. The previous contents of the AC are lost.
Symbolic: $L \longrightarrow AC_{17}$
$0 \longrightarrow AC_{0\text{-}16}$
$0 \longrightarrow L$

Mnemonic: LAW (Load AC with "n")
Octal Code: 760000 + n
Time: 1 cycle
Operation: A single-cycle instruction that loads itself into the AC for the purpose of generating a number, n, of the range $0 \leq n \leq 17777_8$. Following the fetch, the computer enters the contents of the MB (the LAW instruction word) in the AC. The previous contents of the AC are lost. (Refer to figure 7-4.)
Symbolic: $MB \longrightarrow AC$

Some applications of the LAW instruction are:

1. Loading an address for use in establishing indirect operand addressing (if extend mode off).

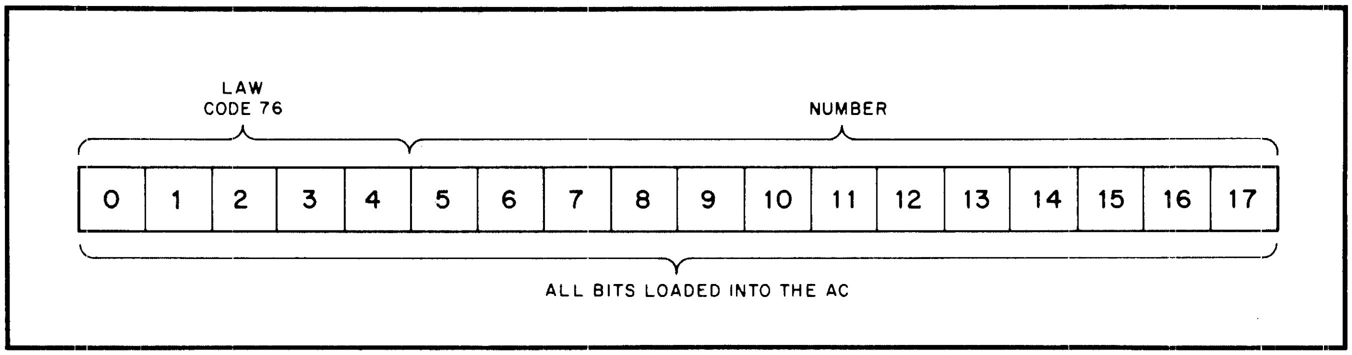


Figure 7-4 LAW Instruction

2. Loading alphanumeric character codes in the AC for use with peripherals.

3. Initializing word counters and/or current address registers.

4. Presetting the real-time clock counter.

Although the entire LAW instruction word is contained in the AC, only bits 5 through 17 can be employed to formulate the required quantity. Bits 0 through 4 serve as the operation code of the instruction.

PROGRAMMING NOTE: The PDP-9/L Symbolic Assembler includes in its permanent symbol table the code for the mnemonic LAM (load minus $177777_8$. To generate a negative (1s complement) number in the AC, the user need only program a LAM-n instruction. After execution of the instruction, the quantity, n, will be present in 1s complement form in the AC.

## INPUT/OUTPUT TRANSFER INSTRUCTIONS

Input/Output transfer (IOT) instructions initiate transmission of signals via the I/O bus to control peripheral devices, sense their status, and effect information transfers between them and the processor. A PDP-9/L IOT instruction contains the following information (figure 7-5):

1. An operation code of $70_8$.

2. An 8-bit device selection code to discriminate one of 256 peripheral devices (selection logic in the I/O bus of a device interface responds only to its preassigned code). In normal practice, bits 6 through 11 perform the primary device discrimination among up to 64 devices with bits 12 and 13 coded to select an operational mode or subdevice.

3. A command code (bits 14 through 17) capable of being microprogrammed to clear the AC and issue up to three pulses via the I/O bus.

The four machine cycles required to execute an IOT instruction consist of the IOT fetch from core memory (memory is not accessed thereafter until completion of the IOT), and there sequential cycles of 1.0 microseconds duration each, designated event times 1, 2, and 3 (figure 7-6). Bits 14 and 17 can be coded to initiate clearing of the AC and generation of an IOP1 pulse for testing a device status flag. IOP2 pulses are normally used to effect programmed transfers of information from a device to the processor. Because the AC serves as the data register for both "in" and "out" transfers, the "clear AC" microinstruction (bit 4) is usually microprogrammed with the IOP2 microinstruction; this combination effects clearing the AC during event time 1 prior to the IOP2 strobe of information into the AC during event time 2. IOP4 pulses are normally used to effect programmed transfers of information from the AC to a selected device. These conventions do not, however, preclude use of the IOP pulses to effect other external functions if the following restrictions are observed.

The <u>usual</u> use of IOPs is:

| | |
|---|---|
| IOP1 - | normally used in an I/O skip instruction to test a device flag. May be used as a command pulse but not to initiate either a "load of" or a "read from" de- a device. |
| IOP2 - | usually used to transfer data from the device to the computer, or to clear a device's information register. May not be used to determine a "skip" condition. |
| IOP4 - | usually used to transfer data from the computer to the device. May not be used to determine a "skip" condition. |

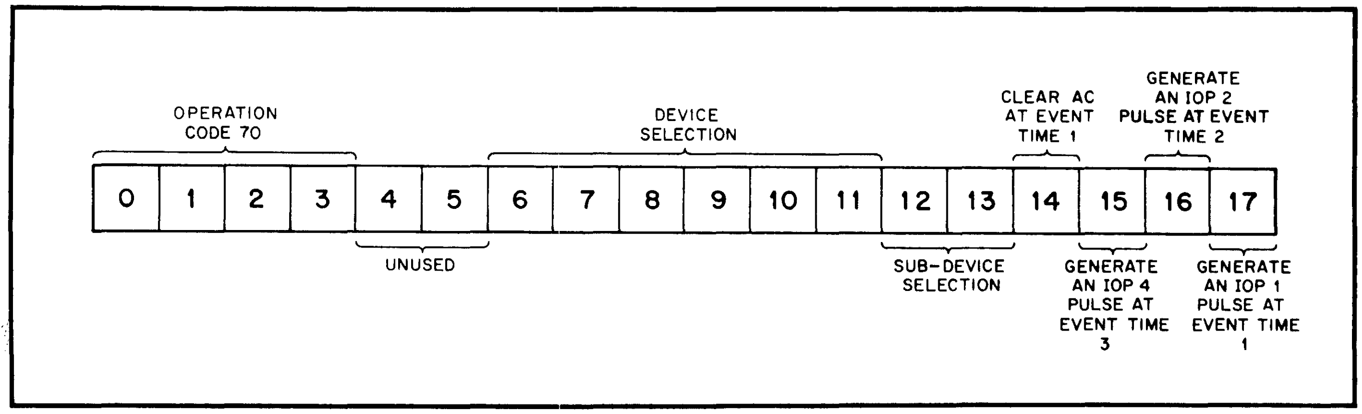

OPERATION CODE 70
DEVICE SELECTION
CLEAR AC AT EVENT TIME 1
GENERATE AN IOP 2 PULSE AT EVENT TIME 2
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
UNUSED
SUB-DEVICE SELECTION
GENERATE AN IOP 4 PULSE AT EVENT TIME 3
GENERATE AN IOP 1 PULSE AT EVENT TIME 1

Figure 7-5 IOT Instruction Format

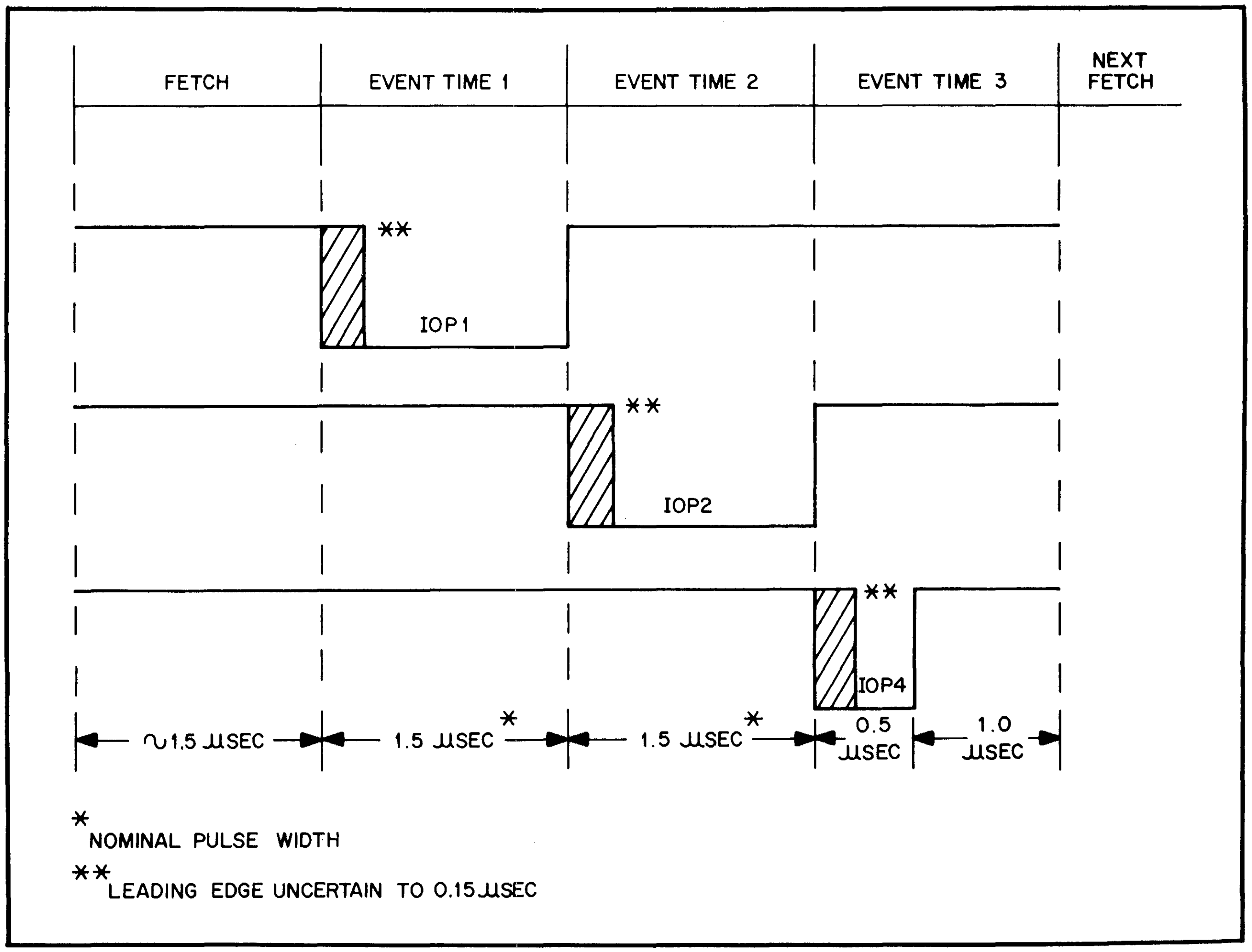


Figure 7-6 IOT Instruction Timing

Programming Note:

Execution of an IOT instruction and the next instruction in sequence cannot be interrupted; i.e., the PDP-9/L will not grant an interrupt request until the instruction following an IOT (and which is not an IOT itself) has completed its function.

**Clear All Flags**

This IOT (703302) is implemented on all PDP-9/L's. Its purpose is to clear the flags of any device that can call for I/O interrupt service. When the CAF instruction is issued, a pulse goes out on the I/O PWR CLR line of the I/O bus.

Customer-installed equipment should also make use of this pulse for flags and registers which should be cleared for system initiation. (This pulse is also issued at power-on time and when the I/O RESET key is depressed.)

## EAE INSTRUCTIONS

The extended arithmetic element option adds the hardware necessary to implement the EAE instructions. This class of instructions, identified by an operation code of $64_8$. performs high-speed data manipulation and multiply-divide operations as specified by microprogramming of individual instructions. Figure 7-7 illustrates the microinstruction capabilities for, respectively, register setup, data shift, normalize, multiply, and divide.

The time required to execute an EAE instruction is a function of the operation and/or the shift. or step, count specified by microprogramming (see table 7-2). In general, the following considerations apply to the different types of EAE operations.

1. All set-up instructions require two machine cycles (3.0 microseconds).

2. Long register shift instructions require a time equal to 2.5 microseconds plus 0.4 microseconds per "n" bit-position shifts, quantized up to next whole time integer. This count is specified by the addition of n (octal) to the instruction code. For example: the input of the symbolic instruction LLS+14 to the PDP-9/L assembler would result in an instruction code that specified a long left shift of the AC and MQ (taken as a 36-bit register) $12_{10}$ bit positions to the left. This instruction would require 8.5 microseconds.

3. The ASL and ALSS instruction, respectively, AC left shift and AC left shift signed, also require the specification of "n".

4. The normalizing instructions, NORM and NORMS, require an execution time equal to 2.5 microseconds plus 0.4 microseconds per number of bit positions shifted to normalize ($AC_0 \neq AC_1$) quantity. These instructions are microprogrammed to set the six-bit step count to $44_8(36_{10})$. Hence, $-44+n_8$ (the step count is entered in 2s complement notation at execution) equals the biased scale factor of a normalized quantity.

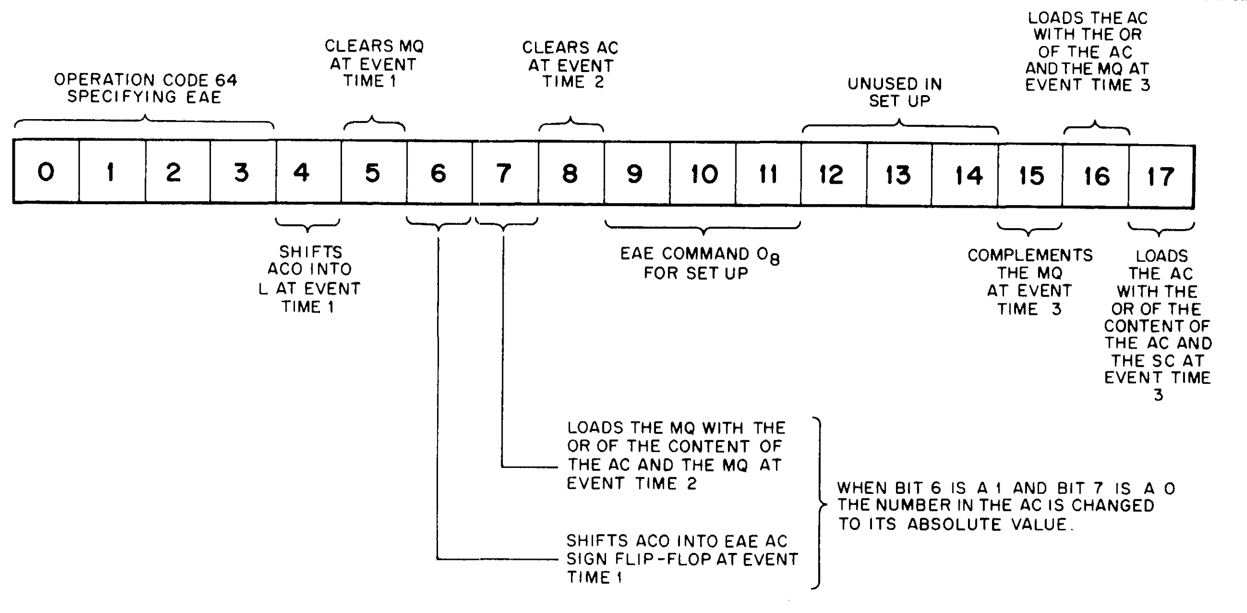
5. Multiply instructions are microprogrammed to set the step count to $22_8(18_{10})$, representing the multiplication of one 18-bit quantity (sign bit and 17 magnitude bits for signed quantities) by another to produce a 36-bit product. The execution time is 12.0 microseconds. Where such precision is not required, the microprogrammed step count can be decreased by subtracting the appropriate number "n" (octal) from the instruction code. The product is always left justified in the AC, MQ. If "-n" is appended to a multiply instruction, the "n" low order bits in the long register are meaningless.

6. Divide instructions are microprogrammed to set the step count to $23_8(19_{10})$, representing division of a 36-bit dividend (actual or implied) by an 18-bit divisor. The execution time is 13.0 microseconds. Where such precision is not required, the microprogrammed step count can be decreased by subtracting the appropriate number "n" (octal from the instruction code. For example, the symbolic instruction DIV-12 would result in a right-justified quotient with the most significant bit in $MQ_9$. The execution time is decreased in correspondence to the decrease in the step count.

Figure 7-8 and table 7-2, which follow the discussions describing the EAE instructions, illustrate the microinstructions of the EAE instructions. Should an existing instruction not suffice, the programmer may combine the appropriate microinstructions to achieve the required result.

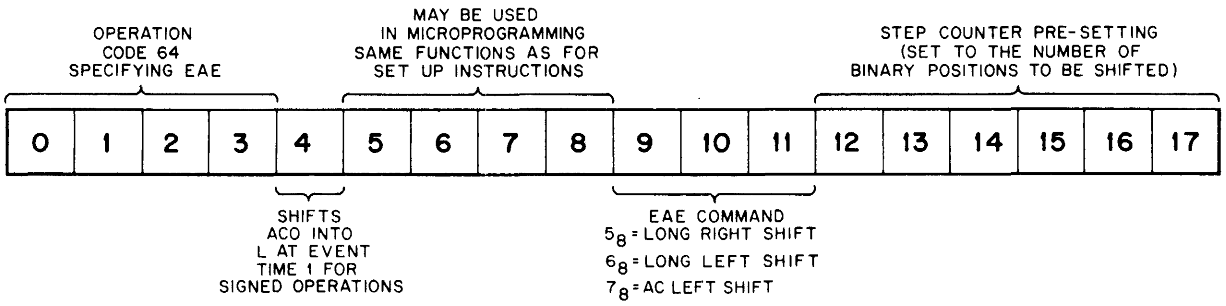
## EAE SETUP

Mnemonic: OSC (Inclusive OR SC with AC)
Octal Code: 640001
Time: 3 microseconds



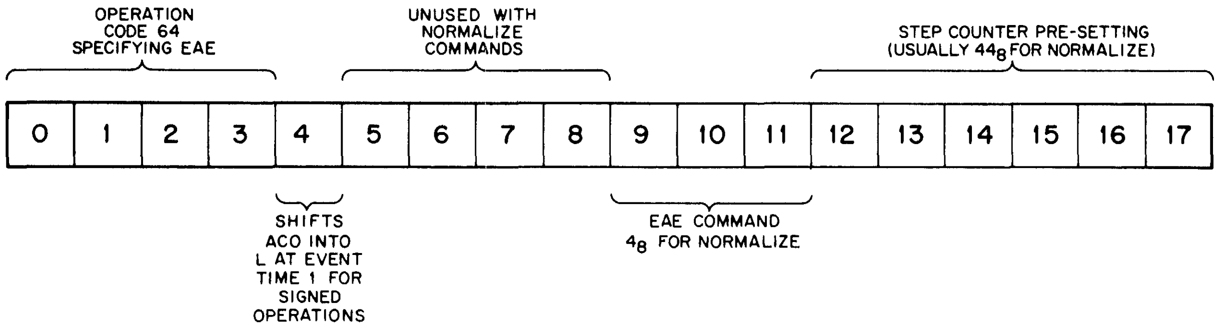
OPERATION CODE 64 SPECIFYING EAE
CLEARS MQ AT EVENT TIME 1
CLEARS AC AT EVENT TIME 2
UNUSED IN SET UP
LOADS THE AC WITH THE OR OF THE AC AND THE MQ AT EVENT TIME 3
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
SHIFTS ACO INTO L AT EVENT TIME 1
EAE COMMAND 0₈ FOR SET UP
COMPLEMENTS THE MQ AT EVENT TIME 3
LOADS THE AC WITH THE OR OF THE CONTENT OF THE AC AND THE SC AT EVENT TIME 3
LOADS THE MQ WITH THE OR OF THE CONTENT OF THE AC AND THE MQ AT EVENT TIME 2
SHIFTS ACO INTO EAE AC SIGN FLIP-FLOP AT EVENT TIME 1
WHEN BIT 6 IS A 1 AND BIT 7 IS A 0 THE NUMBER IN THE AC IS CHANGED TO ITS ABSOLUTE VALUE.

a. EAE Setup Microinstructions



OPERATION CODE 64 SPECIFYING EAE
MAY BE USED IN MICROPROGRAMMING SAME FUNCTIONS AS FOR SET UP INSTRUCTIONS
STEP COUNTER PRE-SETTING (SET TO THE NUMBER OF BINARY POSITIONS TO BE SHIFTED)
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
SHIFTS ACO INTO L AT EVENT TIME 1 FOR SIGNED OPERATIONS
EAE COMMAND
5₈ = LONG RIGHT SHIFT
6₈ = LONG LEFT SHIFT
7₈ = AC LEFT SHIFT

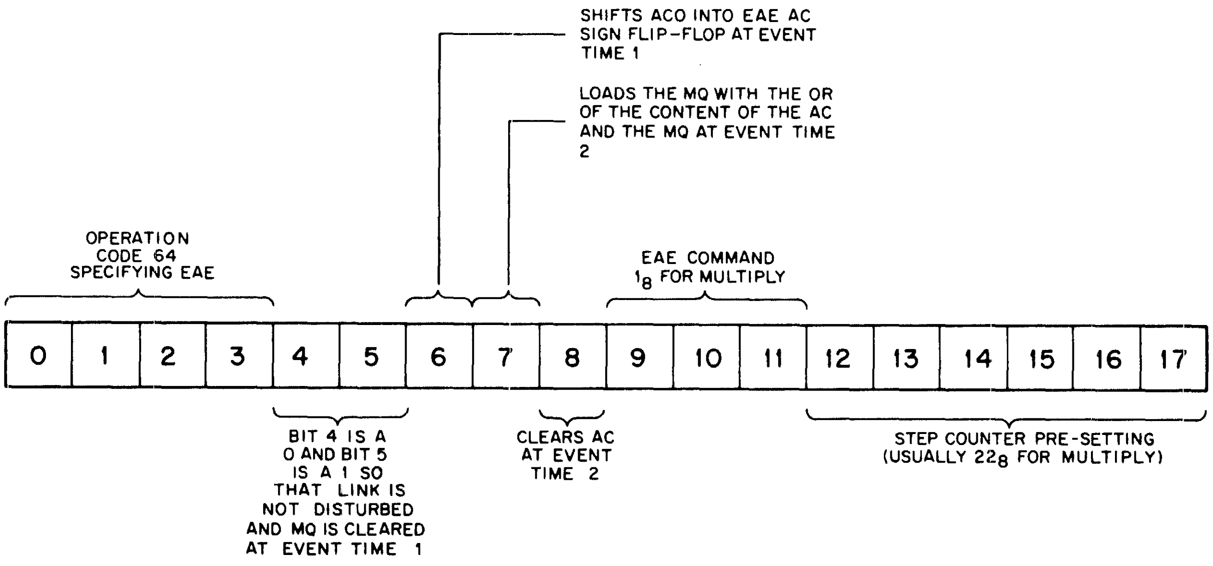
b. EAE Shift Microinstructions



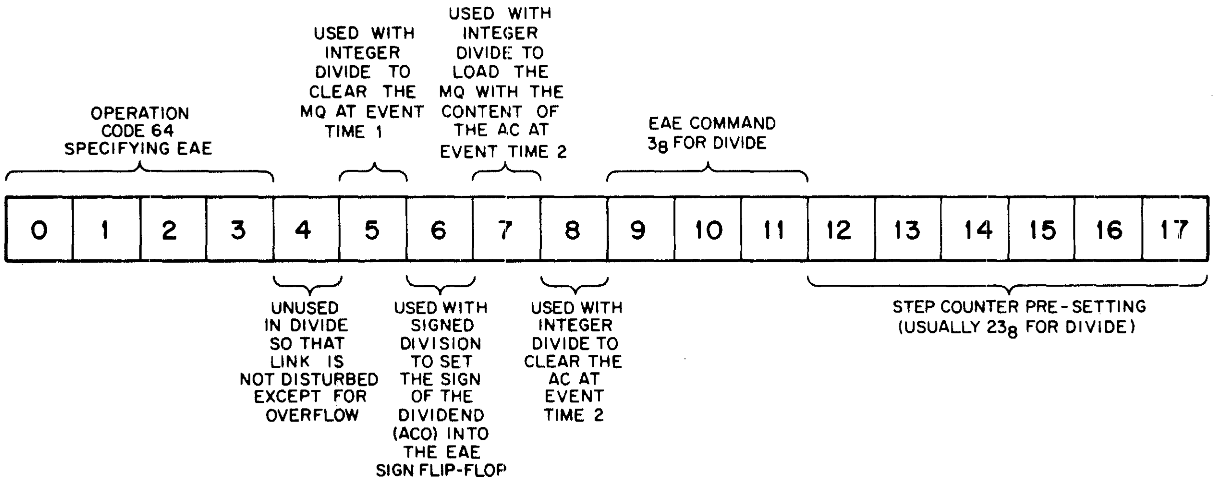
OPERATION CODE 64 SPECIFYING EAE
UNUSED WITH NORMALIZE COMMANDS
STEP COUNTER PRE-SETTING (USUALLY 44₈ FOR NORMALIZE)
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
SHIFTS ACO INTO L AT EVENT TIME 1 FOR SIGNED OPERATIONS
EAE COMMAND 4₈ FOR NORMALIZE

c. EAE Normalize Microinstructions

Figure 7-7. EAE Instruction Formats (Sheet 1)


SHIFTS ACO INTO EAE AC SIGN FLIP-FLOP AT EVENT TIME 1
LOADS THE MQ WITH THE OR OF THE CONTENT OF THE AC AND THE MQ AT EVENT TIME 2
OPERATION CODE 64 SPECIFYING EAE
EAE COMMAND $1_8$ FOR MULTIPLY
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
BIT 4 IS A 0 AND BIT 5 IS A 1 SO THAT LINK IS NOT DISTURBED AND MQ IS CLEARED AT EVENT TIME 1
CLEARS AC AT EVENT TIME 2
STEP COUNTER PRE-SETTING (USUALLY $22_8$ FOR MULTIPLY)

d. EAE Multiplication Microinstructions


OPERATION CODE 64 SPECIFYING EAE
USED WITH INTEGER DIVIDE TO CLEAR THE MQ AT EVENT TIME 1
USED WITH INTEGER DIVIDE TO LOAD THE MQ WITH THE CONTENT OF THE AC AT EVENT TIME 2
EAE COMMAND $3_8$ FOR DIVIDE
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
UNUSED IN DIVIDE SO THAT LINK IS NOT DISTURBED EXCEPT FOR OVERFLOW
USED WITH SIGNED DIVISION TO SET THE SIGN OF THE DIVIDEND (ACO) INTO THE EAE SIGN FLIP-FLOP
USED WITH INTEGER DIVIDE TO CLEAR THE AC AT EVENT TIME 2
STEP COUNTER PRE-SETTING (USUALLY $23_8$ FOR DIVIDE)

e. EAE Division Microinstructions

Figure 7-7. EAE Instruction Formats (Sheet 2)

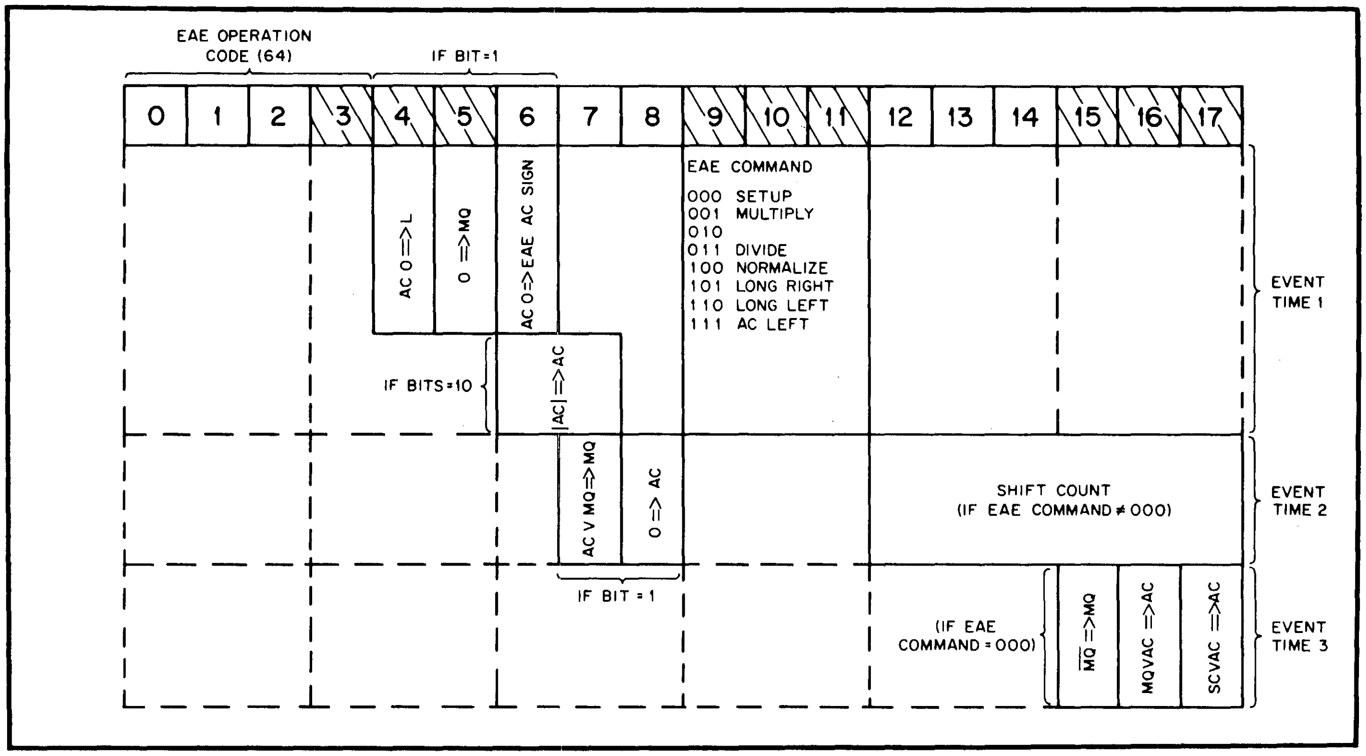


Figure 7-8 EAE Microinstructions

Operation: The contents of the AC are inclusively ORed with the 6-bit contents of the step counter (SC) on a bit-by-bit basis. The result is left in $AC_{12-17}$. If corresponding SC and AC bits are in the binary 1 state, the AC bit is set to 1. The previous contents of the AC are lost. The contents of the SC are unchanged.
Symbolic: SC V AC $\longrightarrow$ AC

| Inclusive OR | | $SC_i$ 0 | 1 |
|---|---|---|---|
| | 0 | 0 | 1 |
| $AC_i$ | 1 | 1 | 1 |

Mnemonic: OMQ (Inclusive OR MQ with AC)
Octal Code: 640002
Time: 3 microseconds
Operation: The contents of the MQ are inclusively ORed with the contents of the AC on a bit-by-bit basis. The result is left in the AC. If corresponding MQ and AC bits are in the binary 0 state, the AC bit is cleared to 0. If either of the corresponding bits is in the binary 1 state, the AC bit is set to 1. The previous contents of the AC are lost. The contents of the MQ are unchanged.
Symbolic: MQ V AC $\longrightarrow$ AC

| Inclusive OR | | $MQ_i$ 0 | 1 |
|---|---|---|---|
| | 0 | 0 | 1 |
| $AC_i$ | 1 | 1 | 1 |

Mnemonic: CMQ (Complement MQ)
Octal Code: 640004
Time: 3 microseconds
Operation: Each bit of the MQ is set or cleared to the inverse of its current state. The previous contents of the MQ are lost.
Symbolic: $\overline{MQ} \longrightarrow$ MQ

Mnemonic: LACQ (Load AC from MQ)
Octal Code: 641002
Time: 3 microseconds
Operation: This microcoded instruction clears each bit of the AC to 0 and then enters the contents of the MQ in the AC. The previous contents of the AC are lost. The contents of the MQ are unchanged.
Symbolic: MQ $\longrightarrow$ AC

Mnemonic: LACS (Load AC from SC)
Octal Code: 641001
Time: 3 microseconds
Operation: This microcoded instruction clears each bit of the AC to 0 and then enters the contents of the SC in $AC_{12-17}$. The previous contents of the AC are lost. The contents of the SC are unchanged
Symbolic: SC $\longrightarrow$ AC

Mnemonic: CLQ (Clear MQ)
Octal Code: 650000
Time: 3 microseconds
Operation: Each bit of the MQ is cleared to 0.

The previous contents of the MQ are lost.
Symbolic: $0 \longrightarrow MQ$

Mnemonic: ABS (Load AC with Absolute Value to AC)
Octal Code: 644000
Time: 3 microseconds
Operation: A microcoded instruction which complements the contents of the AC (1s complement notation) if the contents of $AC_0$ are 1.
Symbolic: If $AC_0 = 1$, $\overline{AC} \longrightarrow AC$

Mnemonic: GSM (Get Sign and Magnitude of AC)
Octal Code: 664000
Time: 3 microseconds
Operation: A microcoded instruction which enters the contents of $AC_0$ in the link, and then complements the contents of the AC (1s complement notation) if $AC_0$ is a 1. The previous content of the link is lost.
Symbolic: $AC_0 \longrightarrow L$
If $AC_0 = 1$, $\overline{AC} \longrightarrow AC$

Mnemonic: LMQ (Load MQ)
Octal Code: 652000
Time: 3 microseconds
Operation: A microcoded instruction which clears each bit of the MQ to 0 and then enters the contents of the AC in the MQ. The previous contents of the MQ are lost. The contents of the AC are unchanged.
Symbolic: $AC \longrightarrow MQ$

Mnemonic: EAE+n (Basic EAE Instruction)
Octal Code: 640000
Time: 3 microseconds
Operation: The addition of "n" (octal) to the mnemonic converts the basic instruction into a microcoded instruction to accomplish a setup, shift, or arithmetic operation not already in the instruction repertoire. Refer to table 7-2 for descriptions of the functional use of the individual bits of an EAE instruction. The sole restriction for the development of "n" is that the microcoded operations must not occur during the same event time if they logically conflict.
Symbolic: not applicable.

## EAE SHIFTING INSTRUCTIONS

Mnemonic: LRS n (Long Right Shift)
Octal Code: (640500)+n
Time: see table
Operation: The AC and the MQ function as a 36-bit register to permit serial shifting of their contents "n" bit positions to the right, "n" being specified by the contents of the six low order bits of the instruction word. Shifting halts when the contents of the step counter, initialized to the 2s complement of "n", reach zero. For each shift step, the contents of $AC_{17}$ enter $MQ_0$ and the contents shifted out of $MQ_{17}$ are lost. The contents of the link, usually initialized to zero, remains unchanged and enters $AC_0$ at each step to replace the contents of vacated bits.

TABLE 7-1 EAE OPERATION TIMES

| Bit Positions Shifted | Multiply or Divide, Unsigned | Multiply or Divide, Signed* | All Shifts or Normalize |
|---|---|---|---|
| 0 | 4 microseconds | 6 microseconds | 2.5 microseconds |
| 1,2,3 | 6 microseconds | 8 microseconds | 4.5 microseconds |
| 4,5,6 | 7 microseconds | 9 microseconds | 5.5 microseconds |
| 7,8,9 | 8 microseconds | 10 microseconds | 6.5 microseconds |
| 10,11,12 | 10 microseconds | 12 microseconds | 8.5 microseconds |
| 13,14,15 | 11 microseconds | 13 microseconds | 9.5 microseconds |
| 16,17,18 | 12 microseconds | 14 microseconds | 10.5 microseconds |
| 19,20,21 | 13 microseconds (divide) | 15 microseconds (divide) | 11.5 microseconds |
| 22,23,24 | | | 12.5 microseconds |
| 25,26,27 | | | 14.5 microseconds |
| 28,29,30 | | | 15.5 microseconds |
| 31,32,33 | | | 16.5 microseconds |
| 34,35,36 | | | 17.5 microseconds |

*Signed multiply producing a positive signed produce requires the number of cycles equivalent to unsigned multiply. Signed divide producing a negative signed quotient with positive signed remainder, if any, requires the number of cycles equivalent to unsigned divide.

Symbolic: $(AC\text{-}MQ)_i \longrightarrow (AC\text{-}MQ)_{i+n}$, i=0, 35-n
$(AC\text{-}MQ)_i \longrightarrow$ Lost; i=36-n, 35
$L \longrightarrow (AC\text{-}MQ)_i$; i=0, n-1
$0 \longrightarrow SC$

Mnemonic: LRSS n (Long Right Shift, Signed)
Octal Code: (660500) + n
Time: see table
Operation: The content of $AC_0$ is entered in the link. Then, the AC and the MQ function as a 36-bit register to permit serial shifting of their contents "n" bit positions to the right, "n" being specified by the contents of the six low order bits of the instruction. Shifting halts when the contents of the step counter, initialized to the 2s complement of "n" reach zero. For each shift step, the contents of $AC_{17}$ enter $MQ_0$ and the contents shifted out of $MQ_{17}$ are lost. The content of the link remain unchanged and enters $AC_0$ at each step to replace the contents of vacated bits.

Symbolic: $AC_0 \longrightarrow L$
$(AC\text{-}MQ)_i \longrightarrow (AC\text{-}MQ)_{i+n}$; i=0, 35-n
$(AC\text{-}MQ)_i \longrightarrow$ Lost; i=36-n, 35
$L \longrightarrow (AC\text{-}MQ)_i$; i=0, n-1
$0 \longrightarrow SC$

Mnemonic: LLS n (Long Left Shift)
Octal Code: (640600) + n
Time: see table
Operation: The AC and the MQ function as a 36-bit register to permit serial shifting of their contents "n" bit positons to the left, "n" being specified by the contents of the six low order bits of the instruction word. Shifting halts when the contents of the step counter, initialized to the 2s complement of "n", reach zero. For each shift step, the contents of $MQ_0$ enter $AC_{17}$ and the contents shifted out of $AC_0$ are lost. The content of the link, usually initialized to zero, remains unchanged and enters $MQ_{17}$ at each step to replace the contents of vacated bits.

Symbolic: $(AC\text{-}MQ)_i \longrightarrow (AC\text{-}MQ)_{i-n}$; i=n, 35
$(AC\text{-}MQ)_i \longrightarrow$ Lost; i=0, n-1
$L \longrightarrow (AC\text{-}MQ)_i$; i=36-n, 35
$0 \longrightarrow SC$

Mnemonic: LLSS n (Long Left Shift, Signed)
Octal Code: ( 660600) + n
Time: see table
Operation: The content of AC is entered in the link. Then, the AC and MQ function as a serial 36-bit register to permit serial shifting to their contents "n" bit positions to the left, "n" being specified by the contents of the six low order bits of the instruction word. Shifting halts when the contents of the step counter, initialized to the 2s complement of "n" reach zero. For each shift step, the contents of $MQ_0$ enter $AC_{17}$ and the contents shifted out of $AC_0$ are lost. The content of the link remains unchanged and enters $MQ_{17}$ at each step to replace the contents of vacated bits.
Symbolic: $AC_0 \longrightarrow L$
$(AC\text{-}MQ)_i \longrightarrow (AC\text{-}MQ)_{i-n}$; i=n, 35
$(AC\text{-}MQ)_i \longrightarrow$ Lost; i=0, n-1
$L \longrightarrow (AC\text{-}MQ)_i$; i=36-n, 35
$0 \longrightarrow SC$

Mnemonic: ALS n (Accumulator Left Shift)
Octal Code: (640700) + n
Time: see table
Operation: The contents of the AC are shifted "n" bit positions to the left, "n" being specified by the contents of the six low order bits of the instruction word. Shifting halts when the contents of the step counter, initialized to the 2s complement of "n", reach zero. For each shift step, the content of the link, usually initialized to zero, enters $AC_{17}$ to replace the contents of vacated bits. The contents shifted out of $AC_0$ are lost.
Symbolic: $AC_i \longrightarrow AC_{i-n}$; i=n, 17
$AC_i \longrightarrow$ Lost; i=0, n-1
$L \longrightarrow AC_i$; i=18-n, 17
$0 \longrightarrow SC$

Mnemonic: ALSS n (Accumulator Left Shift, Signed)
Octal Code: (660700) + n
Time: see table
Operation: The content of $AC_0$ enters the link. Then, the contents of the AC are shifted "n" bit positions to the left, "n" being specified by the contents of the six low order bits of the instruction word. Shifting halts when the contents of the step counter, initialized to the 2s complement of "n" reach zero. For each shift step, the content of the link remains unchanged and enters $AC_{17}$ to replace the contents of vacated bits. The contents shifted out of $AC_0$ are lost.

Symbolic: $AC_0 \longrightarrow L$
$AC_i \longrightarrow AC_{i-n}$; i=n, 17
$AC_i \longrightarrow$ Lost; i=0, n-1
$L \longrightarrow AC_i$; i=18-n, 17
$0 \longrightarrow SC$

Mnemonic: NORM (Normalize)
Octal Code: 640444
Time: see table
Operation: The contents of the AC and the MQ are shifted left (i.e., leading zeros are shifted out) with the AC and MQ functioning as a serial 36-bit register until:

Either the content of $AC_0$ does not agree with the content of $AC_1$; i.e., the bits differ in their binary states.

Or the contents of the step counter reach zero. This six-bit counter is initialized to the 2s complement of $44_8(36_{10}$ steps). The contents of the six low order bits of the NORM instruction word specify the step count. For each shift step, the contents of $MQ_0$ enter $AC_{17}$ and the contents shifted out of $AC_0$ are lost. The content of the link, usually initialized to zero, enters $MQ_{17}$ to replace the contents of vacated bits. If shifting halts because $AC_0$ does not equal $AC_1$, the contents of the step counter reflect the number of steps executed to reach the condition. The counter's contents (2s complement of the step count plus the steps executed) are accessible through use of the OSC or LACS instruction.

Symbolic: n = number of steps required to normalize = $36_{10}$ maximum

$(AC\text{-}MQ)_i \longrightarrow (AC\text{-}MQ)_{i-n}$; i=n, 35

$(AC\text{-}MQ)_i \longrightarrow$ Lost; i=0, n-1

$L \longrightarrow (AC\text{-}MQ)_i$; i=36-n, 35

$(-44_8 + n) \longrightarrow SC$

Programming Note:

Although a NORM or NORMS (normalize, signed) instruction cannot be interrupted by a program or automatic priority interrupt request, the intent of its execution can be interrupted before completion; i.e., the central processor can grant an interrupt request after the normalization of a quantity but before the quantity and its representative step count have been completely extracted from the EAE registers and processed. Hence, if interrupt-accessed subroutines are to make use of the EAE, the following instruction sequences can be included to preserve the register contents during the interrupt and to restore them in the EAE at completion of the interrupt service.

```
/SAVE EAE REGISTERS DURING INTERRUPT
          JMS SUBENTR
SUBENTR,   0
          DAC ACSAVE        /SAVE AC CONTENTS
          LACQ              /MOVE MQ TO AC
          DAC MQSAVE        /SAVE MQ CONTENTS
          LASC              /MOVE STEP COUNT TO
                            /AC
          DAC SCSAVE        /SAVE STEP COUNT
             .
             .
             .
          LAC SCSAVE
          XOR (77           /COMPLEMENT STEP
                            /COUNT
          TAD (640402       /DEVELOP PSUEDO NORM
          AND (640477       /DELETE POSSIBLE STEP
                            /COUNT OVERFLOW
          DAC.+1            /PLACE NORM IN SE-
                            /QUENCE
          HLT*              /STEP COUNT TO SC
          LAC MQSAVE
          LMQ               /LOAD MQ
          LAC ACSAVE        /LOAD AC
          DBR               /RESTORE PC' LINK'
                            /ETC.
          JMP I SUBENTR
```

Restoration of the step count to the step counter requires that the 2s complemented quantity, taken from the SC at the time of interrupt, be complemented and then combined with the psuedo NORM instruction. The step count at this point is one less than the actual value produced by the previous normalization. Execution of the NORM instruction, however, transfers the 2s complemented step count to the SC and in shifting the AC and the MQ left one bit position (the 6404XX present in the AC shifts to become 501XXX) adds the necessary one to the SC to produce the correctly stored step count.

The DBR instruction (which precedes the JMP I subroutine termination) primes the computer for restoration of the interrupted program. Such restoration occurs during the JMP I instruction. During this time, the PC and the Link are restored to the contents existing at the time of h the interrupt, and the options of memory protection and extended memory, if in the system, are restored to their status (on or off) at the time of interrupt. The DBR instruction is fully described in chapter 8 under the discussion of the optional multilevel automatic priority interrupt system. DBR is a basic machine instruction.

Mnemonic: NORMS (Normalize, Signed)
Octal Code: 660444
Time: see table
Operation: The contents of $AC_0$ enter the link. Then, the contents of the AC and the MQ are shifted left (i.e., leading zeros are shifted out) with the AC and MQ functioning as a serial 36-bit register until:

Either the contents of $AC_0$ do not agree with the contents of $AC_1$; i.e., the bits differ in their binary states.

Or the contents of the step counter reach zero. This counter is initialized to the 2s complement of $44_8$ ($36_{10}$ steps). The contents of the six low order bits of the NORMS instruction word

*Good programming practice dictates that instructions to be developed at "run" time be represented by HLT instructions in the source program. If the development does not occur, the HLT will facilitate debugging of the program.

specify the step count. For each shift step, the content of $MQ_0$ enters $AC_{17}$ and the content shifted out of $AC_0$ are lost. The content of the link enters $MQ_{17}$ to replace the contents of vacated bits. If shifting halts because $AC_0$ does not equal $AC_1$, the contents of the step counter reflect the number of steps executed to reach the condition. The counter's contents (2s complement of the step count plus the steps executed) are accessible through use of the OSC or LACS instruction.

Symbolic: $AC_0 \longrightarrow L$
n=number of steps required to normalize = 36
$(AC\text{-}MQ)_i \longrightarrow (AC\text{-}MQ)_{i-n}$; i=n, 35
$(AC\text{-}MQ)_i \longrightarrow$ Lost; i=0, n-1
$L \longrightarrow (AC\text{-}MQ)_i$; i=36=n, 35
$(-44_8 + n) \longrightarrow SC$

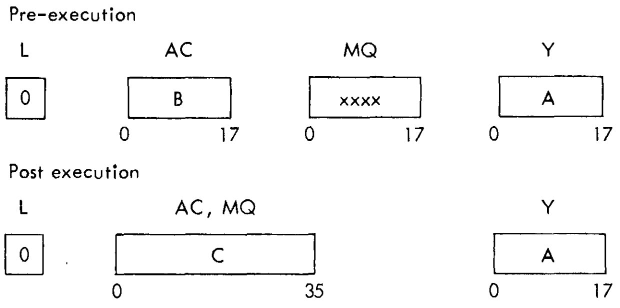
Programming Note:

Refer to the note below the description of the NORM instruction for restoration of the step counter contents following an interrupt.

## EAE ARITHMETIC INSTRUCTIONS

Mnemonic: MUL (Multiply (unsigned)
Octal Code: 653122
Time: 12 microseconds
Operation: Multiply the contents of memory register Y (the multiplicand) by the contents of the MQ (the multiplier), and place the resulting 36-bit product in the AC and the MQ with the more significant half appearing in the AC. The address of Y is taken to be sequential to the address of the MUL instruction word. Prior to this instruction, the contents of the link must be zero and the multiplier must be entered in the AC. During the set-up phase of MUL, the multiplier is transferred to the MQ, the AC is cleared to zero, and the step counter is initialized to the 2s complement of $22_8$ ($18_{10}$ steps); the six low order bits of the instruction word specify the step count. The arithmetic phase, executed as multiplication of one unsigned quantity by another (18 bits, binary point of no consequence), halts when the step counter counts up to zero. The content of the link remains zero. The contents of Y are unchanged. The program resumes at the next instruction (memory register Y + 1).
Symbolic: $0 \longrightarrow SC$
$Y \cdot MQ \longrightarrow (AC, MQ)$
$0 \longrightarrow L$
$PC+2 \longrightarrow PC$

Data Structure: $C = A \cdot B$



Instruction Sequence

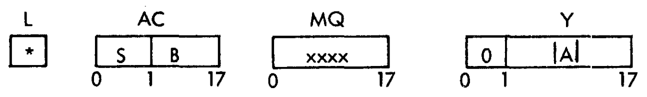
| Register | Contents |
|---|---|
| Y - 2 | LAC Multiplier |
| Y - 1 | MUL |
| Y | Multiplicand |
| Y + 1 | Next Instruction |

Mnemonic: MULS (Multiply, Signed)
Octal Code: (657122)
Time: 14 microseconds
Operation: Multiply the contents of memory register Y (the multiplicand) by the contents of the MQ (the multiplier), and place the signed product in the AC and MQ with the sign notation and more significant portion in the AC. Bits $AC_0$ and $AC_1$ each receive the sign of the product; the remaining AC and MQ bits represent the magnitude of the product in 1s complement form. The address of Y is taken to be sequential to the address of the MULS instruction word. The contents of Y are taken to be the absolute value of the multiplicand; the contents of the link are taken to be the original sign of the multiplicand (MULS assumes previous execution of an EAE GSM instruction, q.v.). Just prior to this MULS instruction, the multiplier must be entered in the AC. During the setup phase of the MULS instruction the multiplier is transferred to the MQ and 1s complemented if negative, the AC is cleared to zero, and the step counter is initialized to the 2s complement of $22_8$ ($18_{10}$ steps); the six low order bits of the MULS instruction word specify the step count. The arithmetic phase, executed as multiplication of one signed quantity by another (sign bit plus 17 magnitude bits, binary point position of no consequence), halts when the step counter counts up to zero. The link is cleared to zero. The contents of Y are unchanged. The program resumes at the next instruction (memory register Y + 1).
Symbolic: $0 \longrightarrow SC$
$Y \cdot MQ \longrightarrow (AC, MQ)$
$0 \longrightarrow L$

$PC+2 \longrightarrow PC$

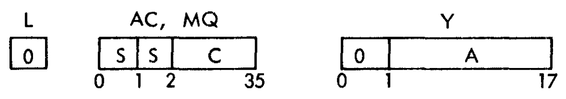
Data Structure: $C = |A|\ B$

Pre-execution



*original sign of A

Post Execution



S = L ⊻ Sign B

Instruction Sequence:

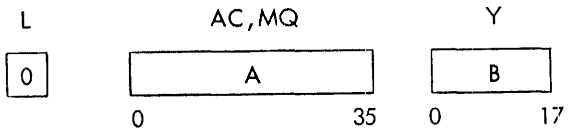
| *Register* | *Contents* |
|---|---|
| Y - 5 | LAC Multiplicand |
| Y - 4 | GSM (take absolute value and save sign in link) |
| Y - 3 | DAC Y |
| Y - 2 | LAC Multiplier |
| Y - 1 | MULS |
| Y | Multiplicand (absolute value) |
| Y + 1 | Next Instruction |

Mnemonic: DIV (Divide (unsigned)
Octal Code: (640323)
Time: 13 microseconds
Operation: Divide the contents of the AC and the MQ (an unsigned 36-bit dividend) by the contents of memory register Y (the divisor). The resulting quotient appears in the MQ; the remainder is in the AC. The address of Y is taken to be sequential to the address of the DIV instruction word. Prior to this, the contents of the link must be zero, and the dividend must be entered in the AC and MQ (LAC least significant half, LMQ, LAC most significant half). If the divisor is not greater than the AC portion of the dividend, divide overflow occurs (magnitude of quotient exceeds the 18-bit capacity of the MQ), and the link is set to one to signal the overflow condition; data in the AC and the MQ are of no value. A valid division halts when the step counter, initialized to the 2s complement of $23_8$ ($19_{10}$ steps), counts up to zero (the six low order bits of the DIV instruction word specify the step count). The contents of the Y are unchanged. The program resumes at the next instruction (memory register Y + 1).
Symbolic: If $Y \leq AC$, $1 \longrightarrow L$

If $Y > AC$,
$0 \longrightarrow SC$
$(AC, MQ)/Y \longrightarrow MQ$ (quotient,
AC (remainder)
$0 \longrightarrow L$
$PC + 2 \longrightarrow PC$

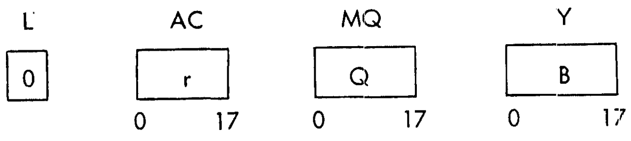
Data Structure: $A = BQ + r$

Pre-execution

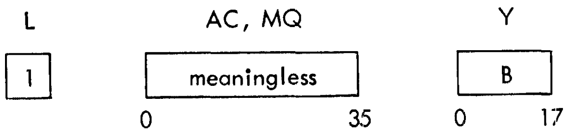


Post execution

(no overflow)



(overflow)



Instruction Sequence:

| *Register* | *Contents* |
|---|---|
| Y - 4 | LAC Dividend (least significant half) |
| Y - 3 | LMQ |
| Y - 2 | LAC Dividend (most significant half) |
| Y - 1 | DIV |
| Y | Divisor |
| Y + 1 | Next instruction |

Mnemonic: DIVS (Divide, Signed)
Octal Code: (644323)
Time: 15 microseconds
Operation: Divide the contents of the AC and MQ (a 36-bit signed dividend with the sign in bits $AC_0$ and $AC_1$ and the remaining 34 bits devoted to magnitude) by the contents of memory register Y (the divisor). The resulting quotient appears in the MQ with the algebraically determined sign in bit $MQ_0$ and the magnitude (1s complement) in bits $MQ_{1\text{-}17}$. The remainder is in the AC with bit $AC_0$ containing the sign of the dividend and bits $AC_{1\text{-}17}$ containing the magnitude (1s complement). The address of Y is taken to be sequential to the address of the DIVS instruction word. The contents of Y are taken to be the absolute value of the divisor;

the contents of the link are taken to be the original sign of the divisor (DIVS assumes previous execution of an EAE GSM instruction, q.v.). Prior to this DIVS instruction, the dividend must be entered in the AC and MQ (LAC of least significant half, LMQ, and LAC of most significant half). The MQ portion of a negative dividend is 1s complemented prior to the division. If the divisor is not greater than the AC portion of the dividend, divide overflow occurs (magnitude of the quotient exceeds the 17-bit plus sign capacity of the MQ), and the link is set to one to signal the overflow condition; data in the AC and the MQ are of no value. A valid division halts when the step counter, initialized to the 2s complement of $23_8$ ($19_{10}$ steps), counts up to zero (the six low order bits of the DIVS instruction word specify the step count). The content of the link is cleared to zero. The contents of Y are unchanged. The program resumes at the next instruction (memory register Y + 1).

Symbolic: If $Y \leq |AC|$, $1 \longrightarrow L$ (divide overflow)

If $Y > |AC|$,

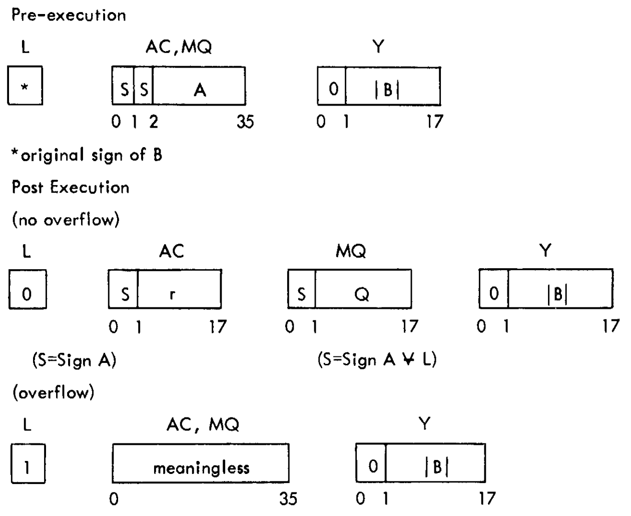
$0 \longrightarrow SC$

$(AC,MQ)/Y \longrightarrow MA$ (quotient), AC (remainder)

$0 \longrightarrow L$

$PC + 2 \longrightarrow PC$

Data Structure: $|A| = B \quad Q + r$



Instruction Sequence:

| *Register* | *Contents* |
|---|---|
| Y - 7 | LAC Divisor |
| Y - 6 | GSM |
| Y - 5 | DAC Divisor in Y |
| Y - 4 | LAC Dividend (least significant half) |
| Y - 3 | LMQ |
| Y - 2 | LAC Dividend (most significant half) |
| Y - 1 | DIVS |
| Y | Divisor (absolute value) |
| Y + 1 | Next Instruction |

Mnemonic: IDIV (Integer Divide (unsigned))
Octal Code: 653323
Time: 13 microseconds
Operation: Divide the contents of the AC and the MQ (AC is zero, MQ contains a 18-bit integer dividend) by the contents of memory register Y (the divisor). The resulting quotient appears in the MQ; the remainder is in the AC. The address of Y is taken to be sequential to the address of the IDIV instruction word. Prior to this instruction, the contents of the link must be zero, and the dividend must be entered in the AC (the setup phase of IDIV transfers the dividend to the MQ and clears the AC). Division overflow occurs only if division by zero is attempted, i.e., the quotient's magnitude will not exceed the 17-bit plus sign capacity of the MQ. The division halts when the step counter, initialized to the 2s complement of $23_0$ ($19_{10}$ steps), counts up to zero (the six low order bits of the IDIV instruction word specify the step count). The content of the link is cleared to zero. The contents of Y are unchanged. The program resumes at the next instruction (memory register Y + 1).

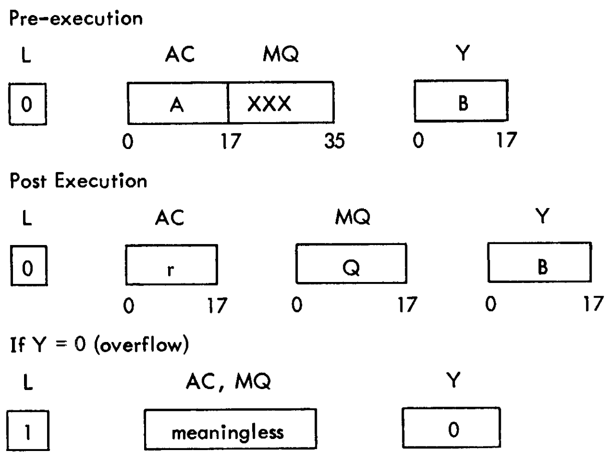
Symbolic: $0 \longrightarrow SC$

$MQ/Y \longrightarrow MQ$ (quotient), AC (remainder)

$0 \longrightarrow L$

$PC+2 \longrightarrow PC$

Data Structure: $A = BQ + r$



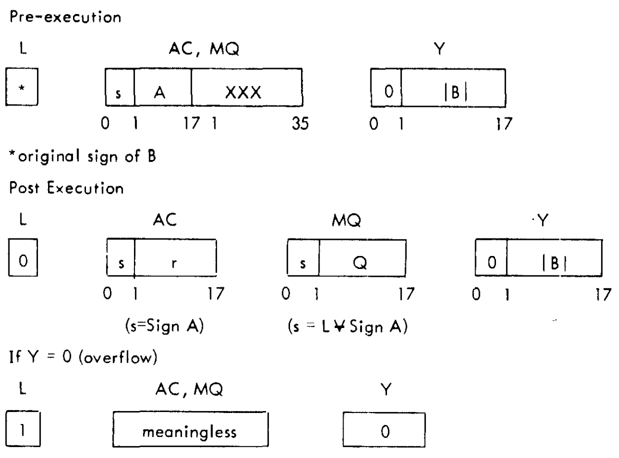
Instruction Sequence:

| *Register* | *Contents* |
|---|---|
| Y - 2 | LAC Dividend |
| Y - 1 | IDIV |
| Y | Divisor |
| Y + 1 | Next Instruction |

Mnemonic: IDIVS (Integer Divide, Signed)
Octal Code: 657323
Time: 15 microseconds
Operation: Divide the contents of the AC and the MQ (AC is zero, MQ contains a signed integer dividend) by the contents of memory register Y (the divisor). The resulting quotient appears in the MQ with the algebraically determined sign in bit $MQ_0$ and the magnitude (1s complement) in bits $MQ_{1-17}$. The remainder is in the AC with bit $AC_0$ containing the sign of the dividend and bits $AC_{1-17}$ containing the magnitude (1s complement). The address of Y is taken to be sequential to the address of the IDIVS instruction word. The contents of Y are taken to be the absolute value of the divisor; the contents of the link are taken to be the original sign of the divisor (IDIVS assumes previous execution of an EAE GSM instruction, q.v.). Prior to this IDIVS instruction, the dividend must be entered in the AC (the setup phase of IDIVS transfers the dividend to the MQ, clears the AC, and 1s complements the MQ if the dividend is negative). Divide overflow occurs only if division by zero is attempted; i.e., the quotient's magnitude will not exceed the 17-bit plus sign capacity of the MQ. The division halts when the step counter, initialized to the 2s complement of $23_8$ ($19_{10}$ steps), counts up to zero (the six low order bits of the IDIVS instruction word specify the step count). The contents of the link are cleared to zero. The contents of Y are unchanged. The program resumes at the next instruction (memory register Y + 1).

Symbolic: $0 \longrightarrow SC$
$MQ/Y \longrightarrow MQ$ (quotient), AC (remainder)
$0 \longrightarrow L$
$PC + 2 \longrightarrow PC$

Data Structure: $A = |B|\ Q + r$



Instruction Sequence:

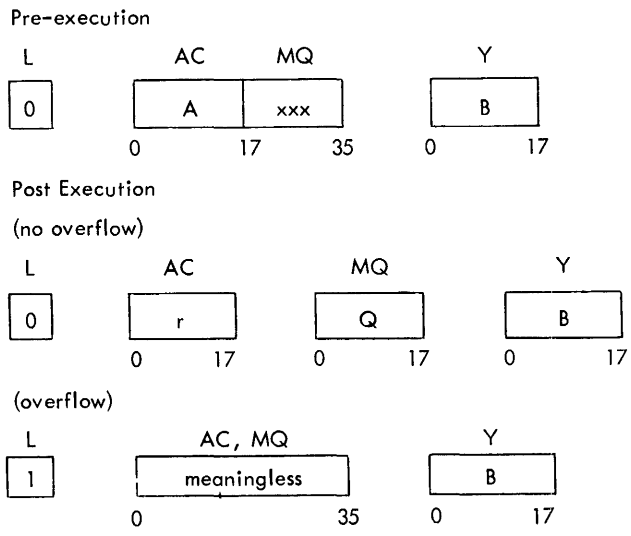
| *Register* | *Contents* |
|---|---|
| Y - 5 | LAC Divisor |
| Y - 4 | GSM |
| Y - 3 | DAC Divisor (absolute value) in Y |
| Y - 2 | LAC Dividend |
| Y - 1 | IDIVS |
| Y | Divisor (absolute value) |
| Y + 1 | Next Instruction |

Mnemonic: FRDIV (Fraction Divide (unsigned)
Octal Code: 650323
Time: 13 microseconds
Operation: Divide the contents of the AC and the MQ (AC contains an 18-bit fractional dividend, MQ is zeroed at steup) by the contents of memory register Y (the divisor). The binary point is assumed at the left of $AC_0$. The quotient appears in the MQ; the remainder is in the AC. The address of Y is taken to be sequential to the address of the FRDIV instruction word. Prior to this instruction, the contents of the link must be zero, and the dividend must be entered in the AC (the set-up phase of FRDIV clears the MQ). If the divisor is not greater than the dividend, divide overflow occurs (magnitude of quotient exceeds the 18-bit capacity of the MQ), and the link is set to one to signal the overflow condition; data in the AC and the MQ are of no value. A valid division halts when the step counter, initialized to $23_8$ ($19_{10}$ steps), counts up to zero (the

six low order bits of the FRDIV instruction word specify the step count). The contents of the link remain zero. The contents of Y are unchanged. The program assumes at the next instruction (mem ory register Y + 1).

Symbolic: If $Y \leq |AC|$, $1 \longrightarrow L$ (divide overflow)
If $Y > AC$ ,
$0 \longrightarrow SC$
$AC/Y \longrightarrow MQ$ (quotient), AC (remainder)
$0 \longrightarrow L$
$PC + 2 \longrightarrow PC$

Data Structure: A = BQ + r



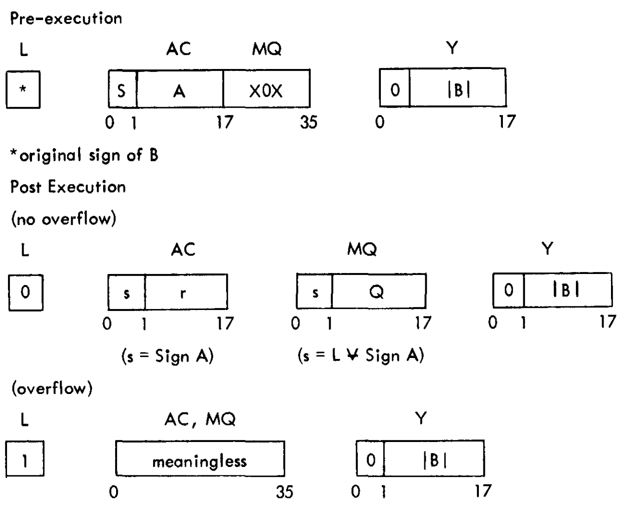
Instruction Sequence:

| *Register* | *Contents* |
|---|---|
| Y - 2 | LAC Dividend |
| Y - 1 | FRDIV |
| Y | Divisor |
| Y + 1 | Next Instruction |

Mnemonic: FRDIVS (Fraction Divide, Signed)
Octal Code: 654323
Time: 15 microseconds
Operation: Divide the contents of the AC and the MQ (AC contains a signed fractional dividend, MQ is zeroed at setup) by the contents of memory register Y (the divisor). The binary point is assumed between $AC_0$ and $AC_1$. The resulting quotient appears in the MQ with the algebraically determined sign in bit $MQ_0$ and the magnitude (1s complement) in bits $MQ_{1-17}$ . The remainder is in the AC with bit $AC_0$ containing the original sign of the dividend and bits $AC_{1-17}$ containing the magnitude (1s complement). The address of Y is taken to be sequential to the address of the FRDIVS instruction word. The contents of Y are taken to be the absolute value of the divisor; the contents of the link are taken to be the original sign of the divisor (FRDIVS assumes previous execution of an EAE GSM instruction, q.v.). Prior to this FRDIVS instruction, the dividend must be entered in the AC (the setup phase of FRDIVS clears the MQ and 1s complements the dividend, if negative, prior to the division). If the divisor is not greater than the dividend, divide overflow occurs (magnitude of the quotient exceeds the 18-bit capacity of the MQ) and the link is set to one to signal the overflow condition. Data in the AC and the MQ are of no value. A valid division halts when the step counter, initialized to the 2s complement of $23_8$ ($19_{10}$ steps), counts up to zero (the six low order bits of the FRDIVS instruction word specify the step count). The contents of the link are cleared to zero. The contents of Y are unchanged. The program resumes at the next instruction (memory register Y + 1).

Symbolic: If $Y \leq |AC|$, $1 \longrightarrow L$ (divide overflow)
If $Y > AC$,
$0 \longrightarrow SC$
$AC/Y \longrightarrow MQ$ (quotient), AC (remainder)
$0 \longrightarrow L$
$PC + 2 \longrightarrow LC$

Data Structure: A = |B| Q + r



Instruction Sequence:

| *Register* | *Contents* |
|---|---|
| Y - 5 | LAC Divisor |
| Y - 4 | GSM |
| Y - 3 | DAC Divisor (absolute value) in Y |
| Y - 2 | LAC Dividend |
| Y - 1 | FRDIVS |
| Y | Divisor (absolute value) |
| Y + 1 | Next Instruction |

TABLE 7-2 EAE MICROINSTRUCTIONS

| Bit Positions | Binary Code | Function |
|---|---|---|
| 4 | 1 | Enter the content of $AC_0$ in the link for signed operations. |
| 5 | 1 | Clear the MQ. |
| 6 | 1 | Read the content of $AC_0$ into the EAE AC sign register prior to carrying out a signed multiply and divide operation. |
| 6,7 | 10 | Take the absolute value of the AC. Takes place after the content of $AC_0$ is read into the EAE AC sign register. |
| 7 | 1 | Inclusive OR the AC with the MQ and read into MQ. |
| 8 | 1 | Clear the AC. |
| 9, 10, 11 | 000 | Setup. Accompanies code in bits 15, 16, and 17. |
| 9, 10, 11 | 001 | Multiply. Causes the number in the MQ to be multiplied by the number in the memory location following this instruction. If the EAE AC sign register is 1, the MQ is complemented prior to multiplication. The exclusive OR of the EAE AC sign and the link is entered in the EAE sign register. The product is in the AC and MQ, with the lowest order bit in MQ bit 17. At completion of this instruction the link is cleared and if the EAE sign is 1, the AC and MQ are complemented. |
| 9, 10, 11 | 010 | Unused operation code. |
| 9, 10, 11 | 011 | Divide. Causes the 36-bit number in the AC and MQ to be divided by the 18-bit number in the memory register following the instruction. If the EAE AC sign is 1, the MQ is complemented prior to starting the division. The exclusive OR of $AC_0$ and the link is placed in the EAE sign register. The AC portion of the dividend must be less than the divisor or divide overflow occurs. In such cases, the link is set and divide does not occur. Otherwise, the link is cleared. At completion of this instruction, if the EAE sign was a 1, the MQ is complemented. Thus, the remainder has the sign of the dividend. |
| 9, 10, 11 | 101 | Long right shift. Causes the AC and MQ to be shifted right together as a 36-bit register the number of times specified in the instruction. On each step the link fills AC bit 0, AC bit 17 fills MQ bit 0, and MQ bit 17 is lost. The link remains unchanged. |
| 9, 10, 11 | 110 | Long left shift. Causes the AC and MQ to be shifted left together the number of times specified in the in struction. On each step, MQ bit 17 is filled by the link; the link remains unchanged. MQ bit 0 fills AC bit 17, and AC bit 0 is lost. |

TABLE 7-2 EAE MICROINSTRUCTIONS (continued)

| Bit Positions | Binary Code | Function |
|---|---|---|
| 9, 10, 11 | 100 | Normalize. Causes the AC and MQ to be shifted left together until the step count is equaled or AC bit 0 ≠ AC bit 1. MQ bit 17 is filled by the link; the link is not changed. The step count of this instruction is normally 44 (octal). When the step counter is read into the AC, it contains the number of shifts minus the initial shift count as a 2s complement 6-bit number. |
| 9, 10, 11 | 111 | Accumulator left shift. Causes the AC to be shifted left the number of times specified in the shift count. AC bit 17 is filled by the link, but the link is unchanged. |
| 12-17 | | Specify the step count for all EAE commands (9-11) except the setup command. |
| 15 | 1 | On the setup command only, causes the MQ to be complemented. |
| 16 | 1 | On the setup command only, causes the MQ to be inclusively ORed with the AC and the result placed in AC. |
| 17 | 1 | On the setup command only, causes the AC to be inclusively ORed with the SC and the results placed in AC bits 12-17. |

# CHAPTER 8
# DATA FORMATS AND ARITHMETIC INFORMATION

## GENERAL

This chapter defines the possible formats for PDP-9/L data words, and presents information basic to the accomplishment of arithmetic operations by the PDP-9/L. The information presented includes: explanations of the three possible notations for signed data (sign and magnitude, 1s complement, and 2s complement); a discussion of scaling considerations for fixed-point arithmetic; and descriptions of the addition and subtraction processes.

The multiply and divide processes are presented in the "Mathematical Subroutines" section of the Software Reference Manual, DEC-9B-GSAA-D. This reference source also includes descriptions of single- and multi-precision arithmetic, plus use of the Basic Software floating-point arithmetic system. Subroutines in the PDP-9/L arithmetic library take full advantage of the computing power of the Extended Arithmetic Element (EAE) when this option is included in a PDP-9/L system.

## SIGNED DATA NOTATIONS

The PDP-9/L uses three notations to represent signed data. They are: sign and magnitude, 1s complement, and 2s complement. In each, bit 0, or the most significant bit of a single- or multi-precision data word, serves as the sign indicator: a 0 for a positive quantity, a 1 for a negative quantity.

### Sign and Magnitude Notation

In the sign and magnitude notation, quantities of equal magnitude but opposite in sign differ only in the content of the sign indicator bit; i.e., the positive number will contain a 0 in bit 0, and the negative number will contain a 1 in bit 0. For example:

```
              S
+131 071₁₀    011 111 111 111 111 111₂
              S
-131 071₁₀    111 111 111 111 111 111₂
```

Observe that conversion of a positive number to a negative number, and vice versa, requires only the sign bit be complemented; the magnitude bits are not affected.

### Complement Notations

In both complement notation (1s and 2s), the sign indicator (bit 0) is 0 for positive quantities and 1 for negative quantities. The 1s complement of a quantity is equivalent to the logical complement of its magnitude and sign; i.e., all binary 1s are replaced by 0s and all binary 0s are replaced by 1s. The 2s complement of a quantity is equivalent to its 1s complement, plus the addition of one to the lowest order, or least significant, bit. Positive quantities in either notation have identical representations. For example: $+15_{10}$ is represented in a PDP-9/L data word as

```
s
000 000 000 000 001 111
```

in either 1s or 2s complement notation. The 1s complement of $-15_{10}$ is represented by

```
s
111 111 111 111 110 000
```

The 2s complement of $-15_{10}$ appears as

```
s
111 111 111 111 110 001
```

A quantity of zero has two representations in 1s complement notation:

```
            S
+0₁₀        000 000 000 000 000 000₂
```

and

```
            S
-0₁₀        111 111 111 111 111 111₂
```

The 2s complement notation has one representation for zero:

```
            S
+0₁₀        000 000 000 000 000 000₂
```

Minus zero in 2s complement notation likewise appears in binary form as

```
              S
-0   -0₁₀     000 000 000 000 000 000₂
```

since complementing each bit and then adding one to the low order bit results in the propogation of an arithmetic carry through the entire word, as follows:

```
          s
+0₁₀      000 000 000 000 000 000₂
```

is complemented to be

```
          s
          111 111 111 111 111 111₂
                               +1
```

with plus one

equals

```
          s
-0₁₀      000 000 000 000 000 000₂
```

The binary 1 carried out of the sign bit "overflows" the 18-bit capacity of the PDP-9/L word. Since two identical representations of 0 would be ambiguous to the computer, convention has adopted one representation of 0 in 2s complement notation, namely +0.

Typical PDP-9/L instruction sequences for forming the 2s complement of any number are:

```
LAC Y        CLC
CMA          TAD Y
TAD (1       CMA
DAC Y        DAC Y
```

The TAD (2s complement add) instruction must be used rather than the ADD (1s complement add) instruction as ADD permits an end-around carry into the low order bit.

The following list indicates the representations in 1s and 2s complement notations of a range of numbers from +5 to −5; a five-bit word is used for simplicity.

| Number | 1s Complement | 2s Complement |
|---|---|---|
| +5 | 00101 | 00101 |
| +4 | 00100 | 00100 |
| +3 | 00011 | 00011 |
| +2 | 00010 | 00010 |
| +1 | 00001 | 00001 |
| +0 | 00000 | 00000 |
| -0 | 11111 | Not considered |
| -1 | 11110 | 11111 |
| -2 | 11101 | 11110 |
| -3 | 11100 | 11101 |
| -4 | 11011 | 11100 |
| -5 | 11010 | 11011 |

## DATA WORDS

PDP-9/L hardware and software capabilities include add, subtract, multiply, divide, etc., of data in single- and multi-precision formats. For signed data words, bit 0 serves as the sign indicator, with a 0 for a positive quantity, and a 1 for a negative quantity.

### Data Word Formats

A signed single-precision data word includes a sign bit and 17 magnitude bits (figure 8-1a).

A signed double-precision data word consists of two computer words for a total of 36 bits (figure 8-1b). The first word contains the sign bit and the 17 most significant bits; the second word contains the 18 least significant bits. The words are normally stored in consecutively addressed core memory locations for ease of programming.

### Magnitudes of Data Words

For 1s complement signed and sign-and-magnitude notations, the permissible magnitude of any quantity, X, is in the range of:

$$-(2^{n-1}-1) \leq X \leq 2^{n-1}-1$$

where n is the number of bits allocated to the storage of data in a data word. For a single-precision data word (sign bit and 17 data bits), this relationship becomes:

$$-(2^{17}-1) \leq X \leq 2^{17}-1$$

or

$$-131\ 071_{10} \leq X \leq +131\ 071_{10}$$

For 2s complement signed notation, the permissible magnitude of any quantity, X, is in the range of:

$$-2^{n-1} \leq X \leq 2^{n-1}-1$$

where n is again the number of bits allocated to the storage of data. A single-precision data word has the range:

$$-2^{17} \leq X \leq 2^{17}-1$$

or

$$-131\ 072_{10} \leq X \leq +131\ 071_{10}$$

The position of the decimal point is implied in the above ranges.

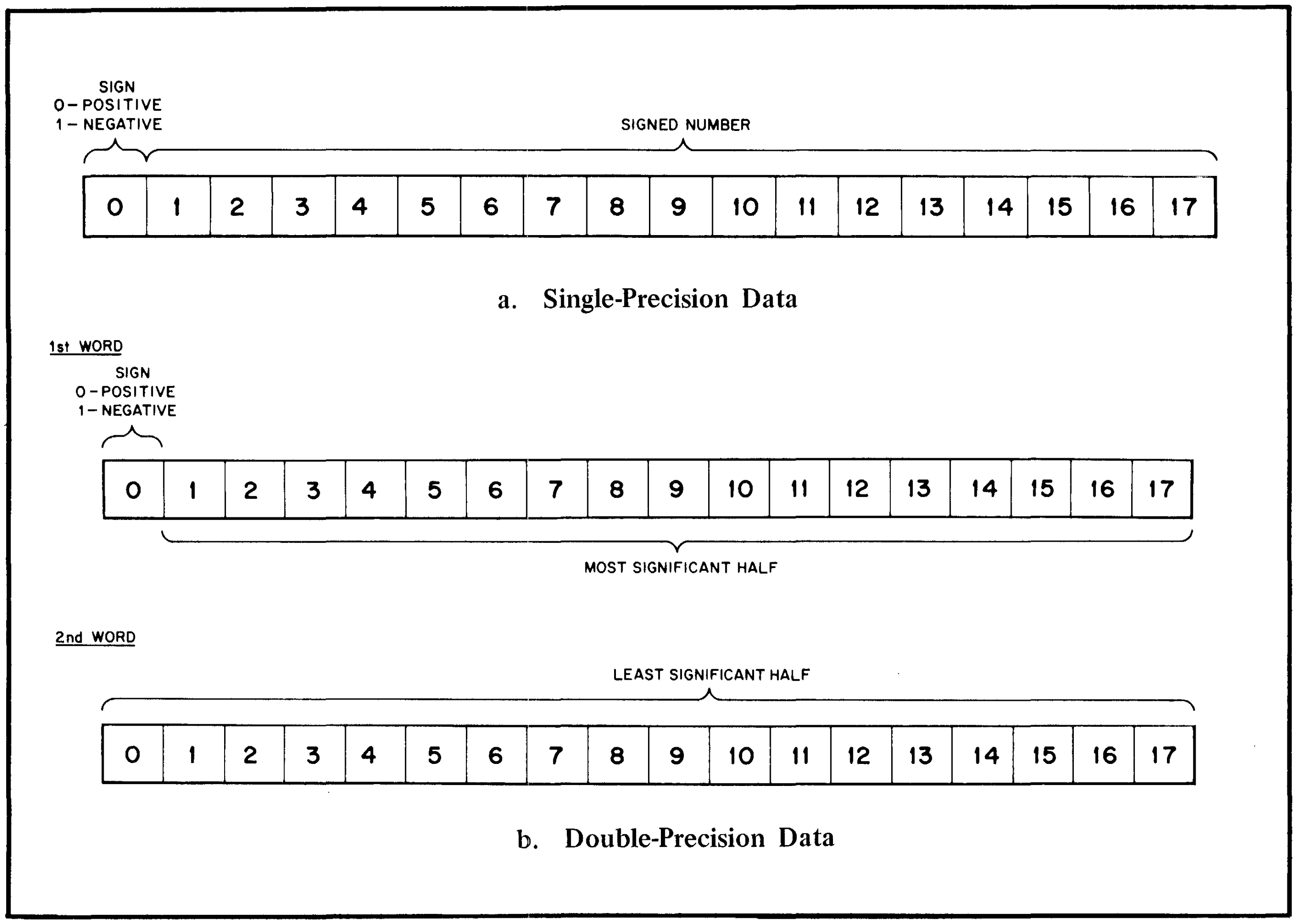


Figure 8-1 Data Word Formats

**Basic Software Floating-Point Formats**

Floating-point representation of a binary number consists of two parts: the exponent and the mantissa. The mantissa is a fraction with the binary point assumed to be positioned between the sign bit and the most significant data bit. The mantissa is always stored in a normalized state; i.e., leading 0s are eliminated from the binary representation so that the high order bit is always a 1. The exponent as stored represents the power of 2 by which the mantissa is multiplied to obtain the number's value for use in computation.

The PDP-9/L floating-point software system offers two modes for storage of floating-point numbers: three-word mode and two-word mode.

The three-word mode requires three memory locations for storage of a floating-point binary number (figure 8-2a). The exponent, a signed 17-bit integer in 2s complement notation, occupies the first word, or memory location. The mantissa, a 35-bit quantity in sign and magnitude notation, is stored in the second and third words. The sign of the mantissa is stored in the high-order bit of the second word.

The two-word mode requires two memory locations for storage of a floating-point binary number, (figure 8-2b). The exponent, an eight-bit integer in 2s complement notation, and its sign occupy the nine high-order bits of the first word. The mantissa, a 26-bit quantity in sign and magnitude notation, is stored in the nine low-order bits of the first word and in the 17 low-order bits of the second word. The sign of the mantissa is stored in the high-order bit of the second word.

## SCALING FOR FIXED-POINT ARITHMETIC

In the programming of arithmetic operations on a fixed-point computer, the position of the scale point (i.e., the binary point in a binary number) must be kept track of by the programmer. Once numbers have been entered in the computer, there is no hardware or movable machine point to

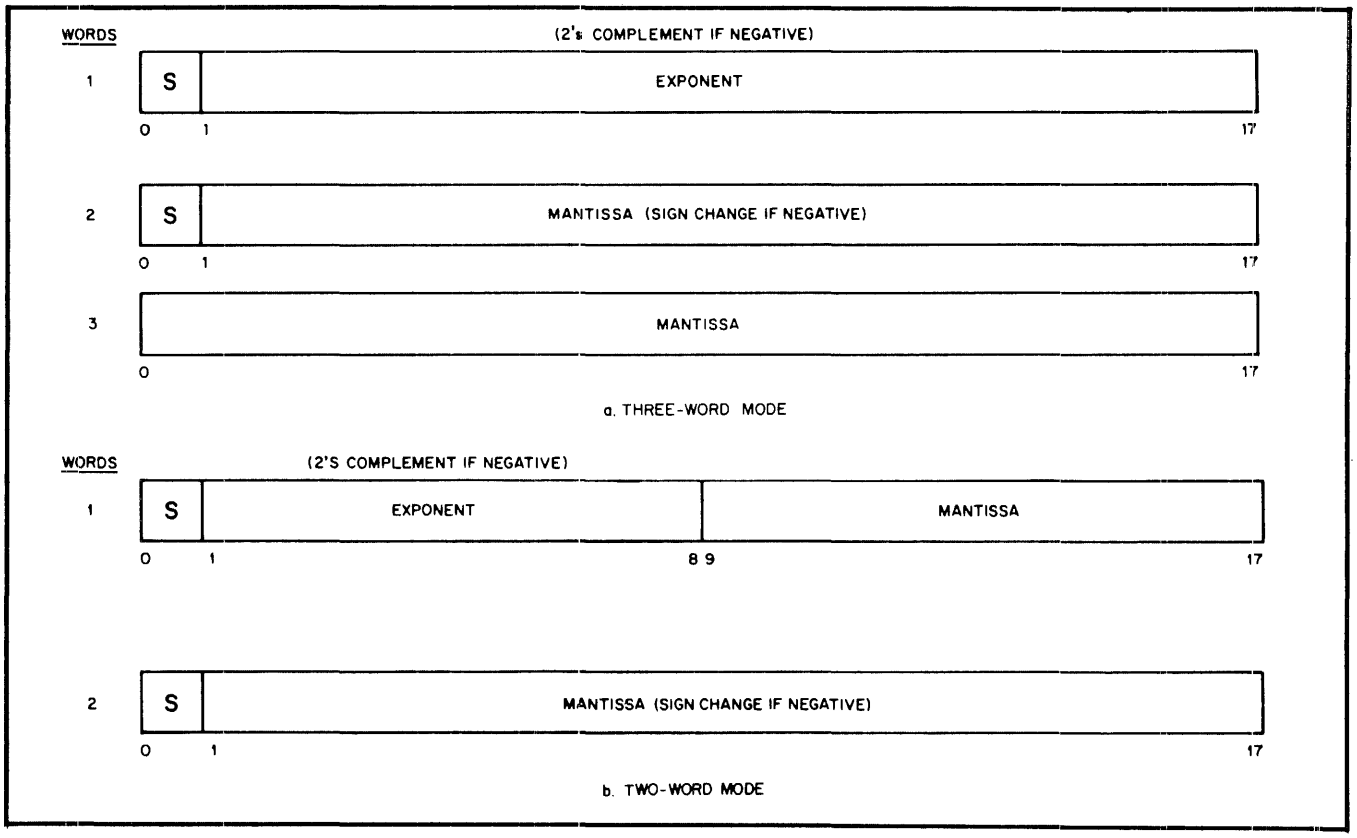

WORDS
(2's COMPLEMENT IF NEGATIVE)
1
S
EXPONENT
0
1
17
2
S
MANTISSA (SIGN CHANGE IF NEGATIVE)
0
1
17
3
MANTISSA
0
17
a. THREE-WORD MODE
WORDS
(2'S COMPLEMENT IF NEGATIVE)
1
S
EXPONENT
MANTISSA
0
1
8 9
17
2
S
MANTISSA (SIGN CHANGE IF NEGATIVE)
0
1
17
b. TWO-WORD MODE

Figure 8-2. Floating Point Formats

represent the scale points. The scale point exists only in the mind of the programmer, and only by keeping track of its imaginary position is he able to correctly interpret the machine's calculated results.

The fundamental properties of scaled numbers can be simply explained by considering a hypothetical decimal machine capable of manipulating numbers consisting of a sign and five decimal digits. As in real computers, this machine acts as if there were a scale point between the sign and the leftmost decimal digit. It is called the machine point. Thus, every number processed by the computer can be thought of as a signed decimal fraction.

Example: $+_{\wedge}12345$

machine point

However, the programmer is free to assign a decimal point at any position in the number. For example, the above number could represent +123.45 if the scale point were thought of as being three places to the right of the machine point. In that case the number would be written $+_{\wedge}12345$ D3, where D3 indicates that the decimal point is three places to the right of the machine point. In other words, D3 is the decimal scale factor.

Other numbers with different scale factors can have the same representation in the machine.

Examples: $+_{\wedge}12345$ D2 = +12.345
$+_{\wedge}12345$ D4 = +1234.5
$+_{\wedge}12345$ D0 = +.12345

The scale factor need not be restricted by the size of the machine word. Numbers in the hypothetical computer can be assigned scale factors that exceed five, or the scale factors can even be negative.

Examples: $+_{\wedge}12345$ D7 = +1234500.
$+_{\wedge}12345$ D-4 = +.000012345

Of course, these are merely programmer's representations; the machine number is always restricted to a sign and five digits.

## Addition and Subtraction

In addition and subtraction, the scale factors of the numbers to be combined must be identical. Thus, $+_{\wedge}42204$ D3 added to $+_{\wedge}23332$ D3 gives a sum of $+_{\wedge}65536$ D3 (422.04 + 233.32 = 655.36). This rule has the same basis as in ordinary arithmetic. If the scale factors differ, one number must be shifted until the scale points are aligned.

```
Examples:  +∧14271 D1 (+1.4271)
           +∧38496 D3 (+384.96)
```

The number +∧14271 D1 is brought into the accumulator and shifted right two decimal places before addition or subtraction. The scale point changes when the number shifts.

```
+001.42    (+.00142 D3)
+384.96    (+.38496 D3)
-------    ------------
+386.38    (+.38638 D3)
```

Notice that if the number of the higher scale factor had been shifted left instead, its two most most significant digits would have been lost and the resulting sum would have been seriously in error.

**Multiplication**

When two numbers are multiplied together, the scale factor of the product is the algebraic sum of the scale factors of the multiplier and multiplicand.

Example:

```
(+ 00200 D3) x (+ 06000 D2) = + 0001200000 )
                                D5, or 12
```

Normally the most significant part of the product is in the AC and the least significant part is in the MQ. Thus, the product in the above example would appear in the computer with the machine point between the sign and leftmost digit in the AC. The machine point for the least significant portion of the product is ignored.

It is important to remember that the two decimal numbers, + 00200 and + 06000, when multiplied in the computer will result in the machine product of + 0001200000 regardless of the positions of the scale points in the multiplier and multiplicand. The scale points must be kept track of by the programmer. Thus fractions, as well as integers, can be multiplied in exactly the same way.

Examples:

```
+∧20000 D5 x +∧00060 D3 = +∧0001200000 D8
                            (+20,000 x +.6 = +12,000)

+∧00200 D0 x +∧06000 D0 = +∧0001200000 D0
                            (+.002 x +.06 = +.00012)
+∧00200 D-2 x +∧60000 D-4 = +∧0012000000 D-6
                              (+.00002 x +.00006 =
                              .0000000012)
```

**Division**

The remarks above for multiplication apply to division, with the following exception. The scale point of the result is the algebraic difference of the scale points of the operands.

## SCALING ON A BINARY COMPUTER

The fundamental properties of scaled numbers in a computer as outlined previously can now be applied to the binary and octal numbering systems as used in a fixed-point, binary computer. The decimal scale factor becomes the binary scale factor and is indicated by a B in front of the scale factor. The machine point is still positioned between the sign and the leftmost digit, but in this case the sign is a binary digit. In a 18-bit computer such as the PDP-9, the leftmost bit is the sign bit and there are 17 bits for the number.

Because the number system used by the machine is now different from that customarily used by the programmer, conversion is a new consideration. The programmer may be dealing with decimal or octal numbers, but because the machine is binary, the scale factors must be determined from the binary equivalents. As will be explained below, a scaling analysis is performed on each problem so that the binary scale factors chosen result in the most efficient use of the 18-bit word. Having selected the appropriate scale factor for a given number, it is expressed in decimal or octal form followed by the binary scale factor. For example, the combination 975 B10 means that the decimal number 975, when converted to binary form, has a binary point ten places to the right of the machine point.

The decimal number 975 B10 can be converted to binary to appear in the machine as follows:

```
(decimal)        975.0        B10
(octal)         1717.0        B10
(binary) 1111001111.0         B10
```

Shift binary point left 10-bit positions

```
        .1111001111 0
```

Add sign bit and trailing 0 bits

```
   S
   0∧11 110 011 11∧0 000 000
machine point    binary point
```

In octal, the machine word is

363600

Negative numbers may be expressed in either the 1s complement notation or the 2s complement notation. For example, consider the octal number:

-3.2 B6

As a positive number, 3.2 B6 would be stored in the computer as

```
 s
 0,00 001 101 000 000 000
  ^
machine point   binary point
```

As a negative number, it would be stored as:

```
 1,11 110 0,10 111 111 111 (1s complement)
  ^M       ^B
```

or

```
 1,11 110 0,11 000 000 000 (2s complement)
  ^        ^
  M        B
```

## OVERFLOW

Suppose we are working with signed quantities and we add the numbers:

| Decimal Value | Binary Representation | Octal Equivalent |
|---|---|---|
| | S | |
| 18 B5 | 0,10 010 000 000 000 000 | 220000 B5 |
| | S | |
| 5 B5 | 0,00 101 000 000 000 000 | 50000 B5 |
| | S | |
| 23 B5 | 0,10 111 000 000 000 000 | 270000 B5 |

Notice that there was no carry to the left of the first machine position (i.e., into the sign bit). However, if we try to add the numbers:

| Decimal Value | Binary Representation | Octal Equivalent |
|---|---|---|
| | S | |
| 28 B5 | 011 100 000 000 000 000 | 340000 B5 |
| | S | |
| 5 B5 | 000 010 000 000 000 000 | 50000 B5 |
| | S | |
| 33 B5 | 100 001 000 000 000 000 | 410000 B5 |

The result as given in the machine would be erroneous because the magnitude portion of the AC is not large enough to hold the sum. This situation is described as overflow.

Overflow is something which must be avoided in all normal circumstances. To accomplish this, the programmer working with fixed-point data must have some knowledge of the magnitude of the numbers in the program and, accordingly, must locate each number at such a scale that overflow cannot occur even in the "worst case".

In this connection, the concept of "minimum binary scale" is helpful. At a binary scale of 5, the largest positive integer than can be contained is $2^5$-1 which in decimal is 31.

The programmer may not always be successful in his attempts to arrange numbers so that overflow will not occur. If a programmer suspects that overflow may occur as a result of an addition or division, he should follow such an operation by a program sequence that would correct the error or at least indicate that such an overflow took place.

The proper location of the binary point and the avoidance of overflow, at best, takes some effort on the part of the programmer.

## PROGRAMMING TECHNIQUES FOR SCALING

One technique used in scaling is to express numbers in a symbolic form that would clearly imply the position of the binary point. The general form is:

$$X2^{-q} = X'$$

where: X is the value of the number.
$2^q$ is the factor such that $q$ is the smallest integer that makes $2^q$ greater than the maximum value of X.
$q$ corresponds to the minimum binary scale factor which was previously discussed.
$X'$ is the scaled form of X (i.e., X is $X'$ with the binary point $q$ places to the right of the machine point).

A scaling analysis should be performed on each problem to insure maximum accuracy (i.e., the most efficient use of the binary word so that there are no leading insignificant bits). At the same time, the programmer must insure that there will be no loss of the most significant bits by overflow at any step in the calculation. These are the two bounds within which the programmer must keep the numbers as they are stored and manipulated in the machine.

### Analysis

In the programming of any given problem or equation, there are three steps prior to the actual coding which should be taken to insure maximum accuracy and to prevent error due to overflow.

Step 1: Determine the limits of the values of all numbers to be used in the problem (maximum and minimum).

Step 2: Determine the scale factors and set up the relationships between the true numbers and the scaled fractions.

Step 3: Substitute the scaled quantities into the original equation and cancel where possible. The scale factors that do not cancel specify the number of required shift operations. If the scale factor of a term is negative, the number must be shifted right before manipulation is performed. If the scale factor is positive, the number must be shifted left if it is to be stored at minimum binary scale.

## Addition Scaling

As emphasized before, quantities to be added (or subtracted) must have the same scale factors. However, in order to prevent an overflow in the summing process, it is not enough to scale the final sum according to its limit. Generally the program must be scaled by the largest limit which applies to any element in the sum or partial sum generated during the summing process.

*Example 1*

Program the operation specified by:

$$A=\sum_{i=1}^{n} a_i$$

where $a_i \leq K$ for $i = 1, 2, 3, \ldots . n$. The maximum value of A is $\leq K \cdot_n$, that is, the maximum value of the sum is obtained by multiplying the upper limit of any element in the list to be summed by the number of elements in the list.

1. Statement

Solve the above problem for n = 10 and K = 100.0;

2. Analysis

Step 1: $a_i \leq 100.0$ for $i = 1, 2, 3, \ldots 10$
Therefore, $A \leq K \cdot n = 100.0 \cdot 10 = 1000$

Step 2: $A = 2^{10} \cdot A^l$
$a_i = 2^7 \cdot a_i^l$

Step 3: $2^{10} A^l \; 2^7 a^l_1 + 2^7 a^l_2 + \ldots \ldots + 2^7 a^l_{10}$

$A^l = 2^{-3} a^l_1 + 2^{-3} a^l_2 \ldots \ldots + 2^{-3} a^l_{10}$

3. Machine Instruction Coding

Assume that the ten values of the numbers are stored in consecutive locations starting at location A1 as $a_i$ B7 and that the sum is stored in A.

```
ADD UP,   DZM  A          /INITIALIZE
          LAM -12+1
          DAC CNTR        /INITIALIZE COUNTER FOR NUM-
                          /BER OF TERMS
          LAC (A1-1       /INITIALIZE AUTOINDEX RERIS-
                          /TER
          DAC AUT 1
LOOP,     LAC I AUT 1     /PICK UP TERM
          CLL
          LRS  3          /SHIFT RIGHT
          TAD  A
          DAC  A
          ISZ CNTR
          JMP LOOP
          JMP DONE        /EXIT, SUMMATION COMPLETED
A1,                       /a1
                          /a2
                          /a3
                           .
                           .
                           .
```

Programming Note:

If the numbers were signed, the instruction sequence of CLL and LRS 3 would be replaced by LRSS 3.

## Multiplication Scaling

When multiplication is performed by the PDP-9/L, the product of two "n" bit numbers is one "2n" bit number. Usually the high-order portion is stored in the MQ.* The fundamental rule, again, is:

Scale factor of multiplier + scale factor of multiplicant = scale factor of product.

1. Statement

Program the multiplication operation:

x = a.b

2. Analysis

| | | |
|---|---|---|
| Step 1: | a | 400.0 |
| | b | 1000.0 |
| Therefore | x | 400,000 |

Step 2: $a = 2^9 a^l$
$b = 2^{10} b^l$
$x = 2^{17} x^l$

Step 3: $2^{17} x^l = 2^9 a^l \cdot 2^{10} b^l$
$x^l = 2^2 a^l \cdot b^l$

*MQ if the EAE option is present; otherwise a memory location.

3. Machine Instruction Coding

Assume that the values of a and b are stored in locations A and B. Assume that they are scaled B9 and B10, respectively.

```
MULT,    LAC B     /PICK UP B
         DAC .+3
         LAC A     /PICK UP A
         MUL
         HLT
         LLS       /SHIFT PRODUCT LEFT 2 PLACES
```

NOTE: This example assumes use of the EAE.

After the multiplication, the shift brings two more significant bits into the high-order protion of the product. Knowing the maximum value of y more definitely (i.e., if a and b could never be maximum at the same time) would allow for even more accuracy. In this example, the limit of y was not known so it was assumed to be 400,000 as calculated in Step 1 of the analysis.

### Division Scaling

When division is performed in digital computers, the dividend is a "2n" bit word and the divisor is an "n" bit word.

Remember that in division the 18-bit divisor word must be greater in magnitude than the 18 high-order bits of the dividend for division to occur without overflow. Therefore, the programmer should scale the values so that division will occur with maximum dividend and minimum divisor. For example, if both dividend and divisor are stored at minimum binary scale, the dividend should be shifted one position to the right by a double-shift subroutine prior to division to insure that overflow does not take place.

## FIXED-POINT ADDITION

Fixed-point addition of a number contained in a core memory location, to a number contained in the accumulator, is performed directly through use of the ADD (1s complement add) instruction, or the TAD (2s complement add) instruction. This assumes the binary points have been aligned through scaling of the quantities and both numbers are properly represented in the appropriate complement notation. Addition can be performed without regard for the signs of the numbers. However, like signed numbers must be scaled to prevent the possibility of overflow.

## FIXED POINT SUBTRACTION

Subtraction in the PDP-9/L is performed by complementary addition; i.e., the subtrahend is converted to its appropriate complement notation and then added to the minuend. As in addition, the binary points of both numbers must be aligned through the provisions of scaling, and both must be represented in the same complement notation. Subtraction can be performed without regard for the signs of the numbers. Assuming that two numbers are both stored in memory locations, typical routines to find the value of A-B follows:

```
     1s Complement                 2s Complement

LAC B   /LOAD SUBTRAHEND     LAC B   /LOAD SUB-
                                     /TRAHEND
CMA     /FORM 1S COMPLE-     CMA     /FORM 1S COM-
        /MENT                        /PLEMENT
ADD A   /-B+A                TAD (1  /FORM 2S COM-
                                     /PLEMENT
                             TAD A   /-B+A
```

Both routines eend with the result in the accumulator.

# CHAPTER 9
# INPUT/OUTPUT CONSIDERATIONS

## GENERAL

This chapter discusses the operation of and the programming techniques for the basic PDP-9/L input/output facilities, the real-time clock, and the Type KF09A Automatic Priority Interrupt and the Add-to-Memory features. The basic facilities include: provisions for executing program controlled (single word) transfers and data channel (block) transfers via the I/O bus; the program interrupt control; the I/O skip facility; and the I/O read status facility.

The PDP-9/L offers two modes for executing I/O data transfers:

1. Program controlled transfers.*
2. Data channel transfers.

Program controlled transfers occur as the result of IOT (input/output transfer) instruction execution. These instructions, contained in the body of a main program or in appropriate subroutines, are microcoded to effect response of a specific device interfaced to the I/O bus system. The microcoding includes the issuing of a unique device selection code and appropriate processor-generated pulses to initiate device operations, such as taking data from the bus or placing data on the bus or clearing device flags. All program controlled transfers are executed through the accumulator in parallel bytes up to 18 bits in length.

The data channel facility provides for relatively high speed transfers of data in blocks between peripherals (DECtape, magnetic tape, etc.) and system core memory. The transfers are controlled by word counter registers and current address registers contained in a memory bank. Eight pairs of these registers are provided to control non-overlapping data channel transfers to or from up to eight devices. A data channel transfer request "breaks" program control at completion of the current instruction** and suspends execution of the program in progress until the current word transfer is completed (three machine cycles for input to memory, four cycles for output to device). Successive breaks are granted to either the active device or another data channel device provided the request for service is made prior to completion of the current channel transfer action. The maximum transfer rate for the facility is between 250,000 and 333,333 words per second, depending on the mix of input and output transfers.

All I/O data transfers function within the precedence of the following priority structure.

1. Data channel (DCH) requests.
2. Real-time clock counting.
3. Automatic priority interrupts (API), 8 levels (optional).
4. Program interrupts (PI).
5. Main program in progress (lowest priority).

A higher priority request for service interrupts any in-process service of a lower priority at the end of the current instruction. Program interrupts and priority interrupts require that the main program transfer control to specific service subroutines. These routines must be programmed to restore control to the interrupted program at completion of the service interval. Computer-granted breaks satisfy data channel requests; i.e., program execution is delayed but not disturbed while the data channel transfers information between memory and the requesting device via the MB.

## PROGRAM CONTROLLED TRANSFERS

The majority of I/O transfers occur under control of the program, taking advantage of the

*In truth, all I/O transfers are made by program control. The differentiation made throughout this handbook refers to the relative degree of control; i.e., the program controlled transfer mode requires the execution of IOT instructions to effect the transfer of each data word, while the channel modes require only that the program initialize the parameters (word count, starting address, etc.) of the block transfer.

**An IOT or XCT instruction prohibits interruption of the instruction immediately following it; i.e., the break is not granted until completion of the instruction following the current IOT or XCT instruction.

control elements present in the computer and in device controls interfaced to the I/O bus. Programmed transfers require more computer and actual time than do data channel and direct memory access transfers. The simplicity and inherent lower cost of the device controls, however, coupled with the high speed of the computer relative to the operational speed of most peripheral devices offset this time discrepancy.

All program controlled transfers take place through the accumulator (AC) in bytes up to 18 bits in length. In transfers within the central processor and between the processor and core memory, data are processed as 18-bit words, the sole addressable unit in the PDP-9/L. For bytes of less than 18 bits, unused bits in the data word normally remain zeroed. Programming techniques of masking and shifting the contents of words may be used to pack and unpack bytes for the purpose of reducing core memory storage requirements. The following program sequence represents a single output transfer to a device:

```
LAC Y          /LOAD THE AC WITH DATA
IOT SKIP       /TEST DEVICE STATUS
JMP .-1        /TEST DEVICE UNTIL READY
IOT WRITE      /TRANSMIT DATA TO DEVICE
```

The LAC instruction reads the data word from its effectively addressed core memory location into the AC. The program then issues the IOT skip instruction to test the readiness of the selected device. It remains in the two-instruction loop completed by the JMP return instruction until the selected device indicates its readiness to accept the data by returning a signal via the I/O skip line to the processor. At that time, the IOT write instruction places the data on the I/O bus data lines, selects the device, and causes it to generate an internal command which strobes the bus data into its information register.

Each IOT instruction is microcoded to issue both unique device selection code and appropriate commands to effect the required device action (refer to chapter 7, for a description of the ITO instruction format; refer to chapters 4 and 5 for description of IOT instructions for peripheral devices offered in the PDP-9/L line).

For a simple input transfer, the following sequence is typical:

```
IOT SKIP       /TEST DEVICE STATUS
JMP .-1        /TEST DEVICE UNTIL READY
IOT READ       /ACCEPT DATA FROM DEVICE
DAC Y          /ENTER DATA IN MEMORY
```

Again, the program issues the IOT skip instruction to test the readiness of the selected device and remains in the two-instruction loop completed by the JMP return instruction until the device signals that it is ready to send data to the processor. At that time, the IOT read instruction selects the device and causes it to generate internal commands which strobe the data onto the I/O bus data lines and send a read request signal to the processor. The processor then inclusively ORs the bussed data into the AC. The DAC instruction deposits the data word in the effectively addressed core memory location. It is normal practice in input transfers to have the IOT read instruction microcoded to effect clearing of the AC prior to the entry of the data sent by the device.

The rate at which programmed transfers can be made is a function of the characteristics of the devices and the program's use of them. The IOT instruction capability of the PDP-9/L allows programmed control of up to 256 devices and the generation of up to three unique commands per each of the 256 possible device selection codes. Devices requiring more than three internal commands are simply assigned additional device selection codes.

## INPUT/OUTPUT READ STATUS FACILITY

Execution of the IOT instruction IORS (octal code 700314) enters the states of device flags (bipolar signals, i.e., 0 or 1) in specific bits of the AC. The state of a specific flag or group of flags can be quickly determined by programmed checks of the AC contents. Figure 9-1 shows the bit positions associated with the commonly interfaced flags. The IORS word can contain up to 18 flag bits. Those bits not used are zeroed. The presence of a flag is indicated by a 1 state in the corresponding AC bit.

Switching the REGISTER DISPLAY control (console) to the STATUS position simulates execution of the IORS instruction with the processor in the "program stop" condition. The contents of the IORS word (i.e., the states of the device flags) are displayed in the REGISTER indicators (console) at this time.

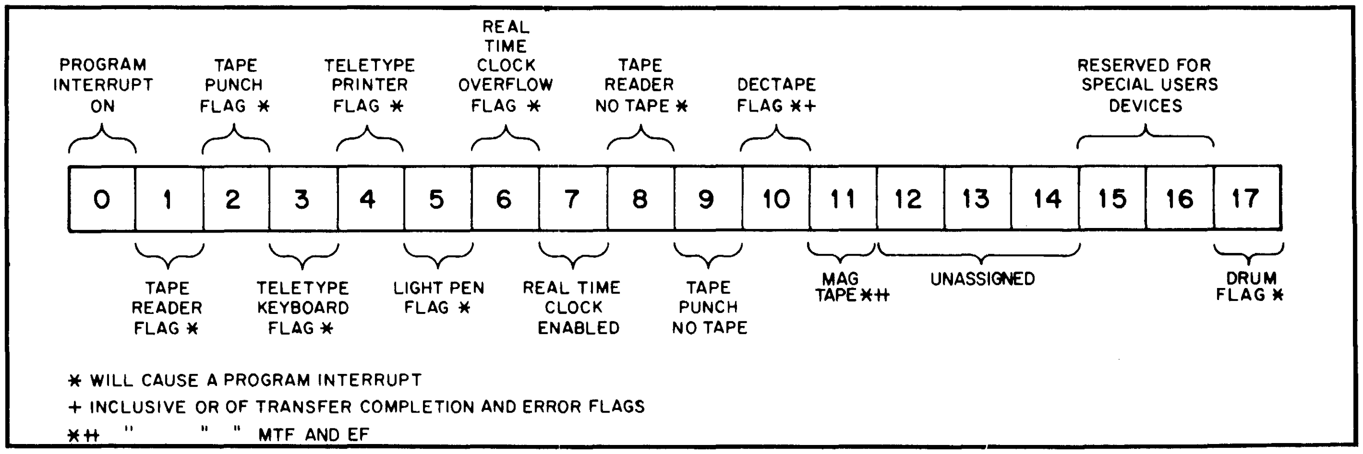


Figure 9-1 IORS Word-Status Bit Assignment

The functions of the device flags normally interfaced to the IORS facility are as follows:

*Program Interrupt* - a 1 bit indicates that the program interrupt control is enabled. A 0 bit indicates that it is disabled. The program interrupt contro is automatically disabled upon the grant of a program interrupt request.

*Tape Reader* - a 1 bit indicates that the reader was previously selected and has assembled a character in its buffer for transfer to the AC upon execution of a "read buffer" IOT instruction. This flag is also interfaced to the program interrupt control to request program interruption when the flag goes to the 1 state.

*Tape Punch* - a 1 bit indicates that the paper tape punch has punched a line of tape relating to the contents of the AC at the time of selection. The flag is also interfaced to the program interrupt control to request program interruption when the flag goes to the 1 state.

*Teletype Keyboard* - a 1 bit indicates that the keyboard buffer has assembled a character code relating to a struck key. The flag is cleared when the assembled code is read into the AC by an IOT instruction. The flag is also interfaced to the program interrupt control to request program interruption when the flag goes to the 1 state.

*Teletype Printer* - a 1 bit indicates that the teleprinter is ready to accept a character code from the AC. The flag is cleared when the teleprinter buffer is loaded and it remains so until the action called by the code has been executed. The flag is then again set to 1. The flag is also interfaced to the program interrupt control to request program interruption when the flag goes to the 1 state.

*Light Pen* - a 1 bit indicates that the Type 370 Light Pen has detected the presence of illumination, normally a CRT-displayed point. The pen is equipped with a manually operated shutter which should be opened only when the pen is positioned on the face of the CRT display. The flag is also interfaced to the program interrupt control to request program interruption when the flag goes to the 1 state.

*Real-Time Clock Overflow* - a 1 bit indicates that the real-time clock counter (stored in memory location 00007 of bank 0) has overflowed; i.e., the initialized clock count (in 2s complement form) has been incremented to zero. The flag is also interfaced to the program interrupt control to request program interruption when the flag goes to the 1 state.

*Real-Time Clock Enabled* - a 1 bit indicates that the real-time clock is enabled and incrementing the contents of location 00007 by one at the rate of 60 times per second (or 50 times per second for 50 Hz powered PDP-9/L systems). A 0 bit indicates that the real-time clock is disabled. The flag is not interfaced to the program interrupt control.

*Tape Reader No Tape* - a 1 bit indicates that the paper tape reader has detected a no-tape condition and halted. In the case of a tape break, since the break may be skewed, approximately 12 lines of previously read tape should be considered as invalid data upon detection of the no-tape flag going to a 1. Although this flag is not interfaced to the program interrupt control, it does force the tape reader flag to go to the 1 state and hence request program interruption for the no-tape condition. A program may make use of the no-tape flag by executing an IORS instruction and testing the AC contents prior to each selection of the reader. An alternative method calls for a program interrupt accessed subroutine to execute the IORS instruction and check the states of the tape-reader and tape-reader-no-tape flags to determine which flag initiated the interruption. While the no-tape flag is a 1, the tape reader will not respond to IOT selection; i.e., the reader is inhibited from reading tape lines. Momentarily depressing the FEED button on the tape reader after loading a tape for readin clears the no-tape flag.

*Tape Punch No Tape* - a 1 bit indicates that the supply of unpunched tape in the internal magazine has been exhausted save for approximately one inch. This length is adequate for the punching of several characters; it may be used also for splicing purposes. This flag does not request a program interruption. Users who desire to make use of this flag must include an execution of the IORS instruction and a test of the AC contents prior to each selection of the paper tape punch.

*DECtape* - a 1 bit indicates that the DECtape flag and/or the error flag (both contained in the TC02 DECtape control unit) are set. This flag is interfaced to the program interrupt control to request program interruption when the flag goes to the 1 state. The function of the DECtape and error flags are discussed in the description of the TC02 control (refer to chapter 5).

## INPUT/OUTPUT SKIP FACILITY

The input/output skip (IOS) facility, as previously shown in the program controlled transfer description, permits IOT instruction testing of those device flags interfaced to it. Such an instruction issues a unique device selection code and then tests the state of the respective device flag. If the flag is in the skip state (normally, the set state), the processor is directed to increment the contents of the PC by one and thus skip the next instruction in sequence. If the flag is found to be not set, the next instruction in sequence is executed. The skip conditions for the various peripherals offered with the PDP-9/L are presented in the IOT instruction descriptions for these devices (refer to chapters 4 and 5).

## PROGRAM INTERRUPT CONTROL

The program interrupt (PI) 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. The program interrupt (PI) control is enabled or disabled by programmed instructions. The following IOT instructions provide for control of this facility:

| Mnemonic | Octal Code | Function |
|---|---|---|
| ION | 700042 | Enable the PI |
| IOF | 700002 | Disable the PI |

The PI is automatically disabled when an interrupt is granted or when the I/O RESET key (Console) is depressed. The PI is temporarily inhibited while the automatic priority interrupt system is processing a priority interrupt request. The PIE indicator (console) is lighted while the PI is enabled.

A program interrupt is granted at completion of the current instruction (IOT and XCT instructions are exceptions)* provided that the PI is enabled and no I/O action of a higher priority is in progress. At the grant of the interrupt, the program in progress "traps" to memory location 00000. This location stores the following data (see figure 9-2 for the storage format):

1. The content of the link register.

2. The contents of the 13-bit PC (or the 15-bit extended PC, if the Type KG09A Memory Extension Control has been included in the system due to memory expansion).

3. The state of the extend mode (on or off) if the KG09A option is present.

4. The state of the memory protection mode (on or off) if the Type KX09A Memory protection option is present.

If the options are not present, the respective bit positions of location 00000 remain zeroed. Following the storage action, the extend mode and the memory protection modes are turned off, and the instruction stored in location 00001 is executed. This instruction is an enter extend mode (EEM) instruction (see chapter 6) that re-enables the extend mode. Immediately thereafter, the instruction in location 00002 is executed. This instruction is a JMP I (indirect address to allow addressing of any memory bank)**;

*An IOT or XCT instruction prohibits interruption of the sequence of it and the instruction immediately following it.

**Both actions (the EEM and JMP I instruction execution) assume the presence of expanded memory and enabled API.

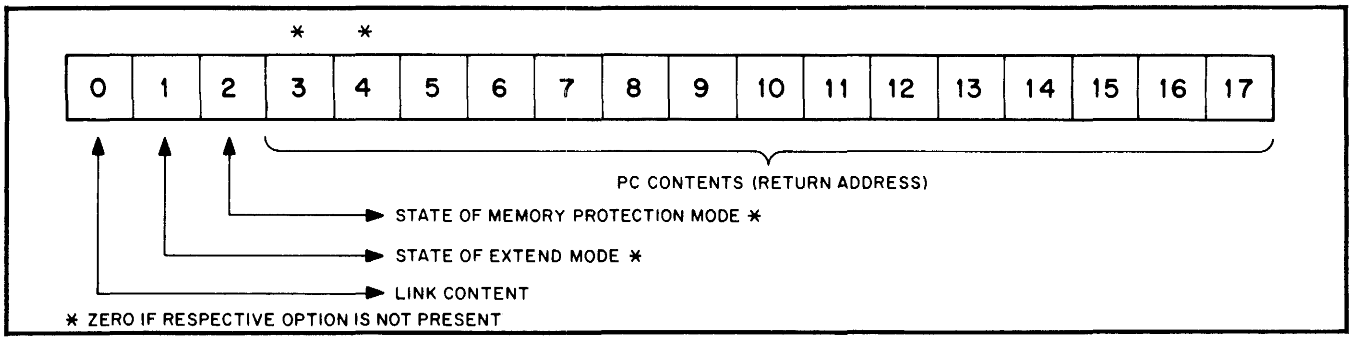

* *
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
PC CONTENTS (RETURN ADDRESS)
STATE OF MEMORY PROTECTION MODE *
STATE OF EXTEND MODE *
LINK CONTENT
* ZERO IF RESPECTIVE OPTION IS NOT PRESENT

Figure 9-2 Program Interrupt, JMS Instruction, or CAL Instruction Storage Word Format*

hence, core memory address 00001 must determine which device requested an interrupt. The interrupt routine interfaced to the PI will service that device. Where multiple devices are interfaced to the PI, the subroutine accessed by the JMP I instruction must execute a flag search to determine which device initiated the interrupt request. Although the PI is not priority oriented (i.e., the interrupt request line accepts the inclusive OR of all device request flags), the order in which the search routine tests the device flags does establish a priority sequence of service.

Several factors must be considered in the use of the PI. First, althoug the PI does automatically save the Link, the PC, etc. to facilitate restoration of the interrupted program at completion of interrupt service, it does not provide for the saving of the contents of other active registers. Thus, interrupt-accessed service subroutines should begin with instructions to temporarily preserve the contents of any registers that may be used by the subroutines.

Secondly, the PI must not be enabled by execution of the ION instruction until the exit from the subroutine. If this precaution is not observed, a second interrupt request could be granted before the current subroutine is completed. The resultant overwriting of location 00000 when the latter request traps to this location will destroy the previously saved data and, hence, prevent restoration of the interrupted main program. Normally, interrupt requests made during current service of an interrupt remain on-line and are answered when the current interrupt service terminates provided the delay does exceed device limitations. The third and final factor requires that interrupt-accessed subroutines terminate with the following instruction sequence:

```
LAC ACSAV     /RESTORE AC
ION           /TURNS ON PI
DBR           /PRIMES SYSTEM TO RESTORE L,
              /PC SAVED MODES AT JMP I TIME
JMP I 0       /RESTORES SYSTEM TO STATUS
              /AT TIME OF INTERRUPT
```

The DBR (debreak and restore) instruction (octal code 703344) is an IOT instruction which sets up the system to restore the Link, PC, extend mode, and memory protection mode to their status at the time of the program interruption; DBR must immediately precede the JMP I instruction. The DBR instruction is fully described in the discussion of the automatic priority interrupt system, but is implemented on all PDP-9/L's.

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 shown in example 1.

In most PDP-9/L systems numerous devices are connected to the PI facility, so the routine beginning in core memory address 00001 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 should also be connected to the IOS.

The program (example 2) illustrates how the program interrupt routine can determine which device is requesting service (this routine assumes expanded memory and active API).

*PI interrupt turns off extend mode and memory protection mode. API interrupt and execution of CAL instruction turns memory protection mode off; extend mode is not affected.

*Example 1:*

```
Tag         Address          Instruction                 Remarks

            01000                 .                 /MAIN PROGRAM
            01001                 .                 /MAIN PROGRAM CONTINUES
            01002                 .                 /INTERRUPT REQUEST OCCURS
                                         INTERRUPT OCCURS
            00000                                   /PROGRAM COUNT, ETC ARE STORED IN 00000
            00001*           JMP SR                 /ENTER SERVICE ROUTINE
SR,         02000                 .                 /SERVICE SUBROUTINE FOR INTERRUPTING
                                  .                 /DEVICE
            03001            ION                    /TURN ON INTERRUPT
            03002            DBR
            03003            JMP I 0000             /RETURN TO MAIN PROGRAM
            01003                 .                 /MAIN PROGRAM CONTINUES
            01004                 .
```

*Note: This routine illustrates PI programming for a PDP-9/L system without expanded memory and API.

*Example 2:*

```
Tag         Address          Instruction                 Remarks

            01000                 .                 /MAIN PROGRAM
            01001                 .                 /MAIN PROGRAM CONTINUES
            01002                 .                 /INTERRUPT REQUEST OCCURS
                                         INTERRUPT OCCURS
            00000                                   /STORE PC, LINK, ETC.
            00001            EEM                    /ENTER EXTEND MODE
            00002            JMP I 00003            /ENTER ROUTINE
FLGCK,                       IOT 3401               /SKIP IF DEVICE 34 IS REQUESTING
                             JMP SR 34              /ENTER SERVICE ROUTINE 34
                             IOT 4401               /SKIP IF DEVICE 44 IS REQUESTING
                             JMP SR 44              /ENTER SERVICE ROUTINE 44
                             IOT 5401               /SKIP IF DEVICE 54 IS REQUESTING
                             JMP SR 54              /ENTER SERVICE ROUTINE 54
00003, FLGCK
```

## AUTOMATIC PRIORITY INTERRUPT

The API option, Type KF09A, adds eight additional levels of programming priority to the PDP-9/L. The upper four levels are assigned to I/O devices and are initiated by flags (interrupt requests) from these attached devices. The lower four levels are assigned to the programming system and are initiated by software requests. The priority network insures that high data rate or critical devices assigned to high priority levels will always interrupt slower device service routines while holding still lower priority interrupt requests off line until they can be serviced. The API also identifies the cause of the interrupt directly, eliminating the need for the service routines to flag search.

The key elements in the API are priority level and channel. Each I/O device in a PDP-9/L system is assigned to one of the 4 hardware API priority levels or to the program interrupt facility so as to maximize performance of the entire I/O system. The channel assignment (API provides for 32) of every device is fixed and cannot be changed.

The API operates in the following manner. An I/O device requests service by transmitting an interrupt request signal to the processor on a line corresponding to its specific, preassigned priority level. If this priority level is higher than the priority of the device which requested a currently active program segment, an interrupt is granted to the new device. Upon receipt of the grant signal, the device transmits its channel address back to the processor. The processor executes the instruction in the specified memory address; this is always a JMS (or JMS I)* to the device service subroutine. The new priority level is remembered and no further acceptance of requests on this or lower priority levels is permitted until the device service subroutine is exited.

The hardware insures that simultaneous requests by multiple devices are handled in the proper priority sequence. If interrupt requests occur at different priority levels, the highest priority request will be serviced first. If multiple interrupt requests occur at the same priority level, the device closest on the bus to the processor will be serviced first. The entire API system may be enabled or disabled with a single instruction, however, many devices provide facility to

*The indirect address permits access of a subroutine stored in a memory bank other than bank 0.

separately connect and disconnect their flags from the interrupt.

The major advantage of this API system lies in the proper use of the software levels. In the real-time environment, it is necessary to maintain data input/output flow, but it is not possible to perform long, complex calculations at priority levels which shut out these data transfers. With the API, a high priority data input routine which recognizes the need for the complex calculation can call for it with a software level interrupt. Since the calculation is performed at a lower priority than the data handling, the latter can go on undisturbed. Further, there is no need to interface the data collection routine with the lowest priority (background) program which may run independently of the real-time system. Since the priority juggling is handled by the hardware it is quite efficient.

## Priority Level

At any time, the computer is actually executing instructions from one and only one program segment. However, if the program segment being executed is the result of an interrupt (due to a peripheral device flag), then both the interrupted program and the interrupt service program can be thought of as concurrently active. Such is the case in the basic PDP-9/L when the normal program interrupt facility (PI) is used. The API options add 8 additional priority levels above the two available (PI and main program) in the basic computer. Thus, a total of 10 levels of priority exist. Priority levels 0-3 are for hardware; levels 4-7 for software. Level 0 is the highest priority.

Each peripheral device attempts to interrupt at a specific priority level. If there are no active programs at that or higher priority levels, the program segment in progress will be interrupted, return information stored, and the new device service routine entered. If there is a higher priority level program active, the device request is ignored until the higher priority program segment terminates. The high priority levels then go inactive and the requesting device is serviced.

A CAL instruction causes priority level 4 to be activated after the CAL instruction is executed. A break to level 4 will occur after all higher priority level requests are honored.

## Channels

Each peripheral device is assigned to a channel independent of its priority level. The channel assignment of a device defines a device service subroutine entry point in the following manner: When an interrupt is granted to a device, the device transmits an address to the computer that is simply related to its channel number. (Channel address = $40_8$ + channel number). The assigned priority level flip-flop is turned on and the instruction at the channel address is executed. This is always a JMS to the device service subroutine.

There are $32_{10}$ channels numbered $0\text{-}37_8$ with corresponding core addresses of 40-77. Four channels (40-43) are used for software priority levels, leaving 28 for device use. Each of the four hardware priority levels is multiplexed such that up to eight devices (channels) may use it.

It is not possible to enable or disable an individual channel. Rather, the more sophisticated I/O devices connected to the interrupt will have the ability to enable or disable themselves from their interrupt lines. Simple devices such as the reader, punch, etc., will clear their flags to disconnect, just as they disconnect from the program interrupt.

## API IOT Instructions

The instructions listed in table 9-1 are supplied with the API option. Note that although the DBR (debreak and restore) instruction appears in this listing, it is a basic machine instruction; hence, DBR may be used in PI-entered subroutines as well as in API-entered subroutines.

## Dynamic Priority Reallocation

In order to most efficiently service real-time devices, the hardware includes provision for dynamic priority reallocation. There are three distinct methods.

1. *Device dependent* - Since channel and priority level are independent, a device *may be designed* to interrupt at any of several priority levels without grossly affecting programming. In a control application, the device could raise its priority (under program control) when, for example, the data rate increased. In this case the device would make use of more than one device priority level.

2. *Program generated service requests* - The program may generate interrupt requests on any of the four software priority levels. If the priority level of the request is below an active priority level, the request will (eventually) be serviced when the higher priority active levels are dismissed. If the priority level of the request is above all active levels,

TABLE 9-1 API IOT INSTRUCTIONS

| Mnemonic Symbol | Octal Code | Function |
|---|---|---|
| SPI * | 705501 | Skip on priorities inactive. The next instruction is skipped if any AC bit is set and the corresponding API condition is true (see table 9-2). |
| ISA* | 705504 | Initiate selected activity. The API activity specified by a set bit in the AC is initiated (see table 9-3). |
| DBK | 703304 | Debreak. This instruction is used to release the highest active priority level. Its use is to return a subroutine's priority to the normal assignment after the requirement for an interim ISA-initiated raising of priority has been satisfied. DBK should not be used to terminate a subroutine as it does not provide for restoration of the PC, Link, etc. |
| DBR | 703344 | Debreak and restore. In addition to releasing the active priority level, this instruction primes the PDP-9/L to restore the Link, the PC (or EPC), the extend mode, and the memory protect mode to their status at the time of interruption. The actual restoration occurs at the execution of the JMP I instruction exiting the subroutine. DBR must immediately precede the JMP I instruction. Where DBR is used in subroutines not entered by API action, the debreak operation has no significance. |
| RPL | 705512 | The priority levels are read into the AC according to the bit usage shown in table 9-4. AC bits 2-9 indicate the presence of devices requesting service on levels 0-7. AC bits 10-17 indicate those priority levels that are active. (A level becomes active if service at this level has commenced, or if a raise priority level instruction (ISA) has been executed initiating activity on a higher level.) |

*Programming Note: normally, the SPI and ISA instructions are combined (microcoded) to first test that the program segment currently in progress is not already at the requested priority level and then if not, to initiate a raising of priority to the requested level. Hence, if a program segment cannot raise its priority, the segment must be already at the requested level or higher. The ISA instruction cannot be used to lower the priority level of an active program segment. The hardware will not recognize the priority change.

the request will be serviced immediately. The instruction (JMS) in the software priority level channel will be executed, storing the current program counter and entering a new program segment.

3. *Programmed priority changes* - In order for an interruptable program to change parameters in an interrupt service subroutine, the priority interrupt system is normally turned off while the changes are affected. Unfortunately, all interrupts are shut out during this time including those that indicate machine errors or are vital to control real-time processes. Thus, the API has been designed so that a program segment may raise its priority only high enough to shut out those devices whose service routines require changes.

The method of raising priority and lowering it requires minimum possible time. By issuing the ISA instruction with the proper bits set in the accumulator the priority of the currently active program segment is raised. No instruction in a channel is executed. The program merely continues on at its higher priority level. To restore the program segment to its original priority level, a DBK instruction is issued.

For example: a priority 2 routine is entering data in memory locations A through A + 10; but, based on a calculation made by a priority 6 routine, it becomes necessary to move the data to memory locations B through B + 20. The changes in the routine at level 2 must be completed, without interruption,

TABLE 9-2 STATUS BITS ASSOCIATED WITH THE SPI INSTRUCTION

| AC Bit | Function |
|---|---|
| 0 | API enable (system is preesntly enabled if 1, disabled if 0) |
| 1 | Unused |
| 2 | Unused |
| 3 | Unused |
| 4 | Unused |
| 5 | Unused |
| 6 | Unused |
| 7 | Unused |
| 8 | Unused |
| 9 | Unused |
| 10 | Priority level 0 inactive, hardware* |
| 11 | Priority level 0 and 1 inactive, hardware |
| 12 | Priority level 0-2 inactive, hardware |
| 13 | Priority level 0-3 inactive, hardware |
| 14 | Priority level 0-4 inactive, software |
| 15 | Priority level 0-5 inactive, software |
| 16 | Priority level 0-6 inactive, software |
| 17 | Priority level 0-7 inactive, software** |

*Highest priority.
**Lowest priority.

once they are started. This is possible by the level 6 program raising itself to level 2 (devices on the *same* or lower priority may not interrupt), completing the change, and debreaking back to level 6.

## Programming Examples

*Input 10 Words from A/D Converter* - A service routine INAD has been written to input 10 words to a FORTRAN array for later processing. The core location of the A/D channel contains a JMS INAD. The basic components of INAD are:

```
INAD,   0              /ENTRY POINT
        DAC SAVAC      /SAVE AC
        IOT            /READ A/D BUFFER
         .             /STORE IN ARRAY
         .             /TEST FOR LAST WORD - IF
                       /YES, INITIATE SOFTWARE
                       /INTERRUPT TO ACCESS DATA
                       /FORMATTING SUBROUTINE
        IOT            /ELSE, START NEXT CON-
                       /VERSION
        LAC SAVAC      /RESTORE AC
        DBR            /DEBREAK AND RESTORE
        JMP I INAD     /RETURN
```

The program segment to start the conversion would look as follows:

TABLE 9-3 CONTROL BITS ASSOCIATED WITH ISA INSTRUCTION

| AC Bit | Function |
|---|---|
| 0 | API enable (enable API system if bit is a 1, disable system if bit is a 0). |
| 1, 2 | For paper tape reader: AC BIT 1 2 = 0 0: Reader priority level 2; 0 1: 1; 1 0: 0; 1 1: Remove PTR from API |
| 3 | Unused |
| 4 | Unused |
| 5 | Unused |
| 6 | Request interrupt at priority level 4 |
| 7 | Request interrupt at priority level 5 |
| 8 | Request interrupt at priority level 6 |
| 9 | Request interrupt at priority level 7 |
| 10 | Raise priority to priority level 0 |
| 11 | Raise priority to priority level 1 |
| 12 | Raise prioirty to priority level 2 |
| 13 | Raise priority to priority level 3 |
| 14 | Raise priority to priority level 4 |
| 15 | Raise priority to priority level 5 |
| 16 | Raise priority to priority level 6 |
| 17 | Raise priority to priority level 7 |

```
         .           /INITIALIZE INAD
        IOT          /SELECT CONVERTER FOR
                     /FIRST CONVERSION
         .           /CONTINUE WITH PROGRAM
```

If INAD were active, one could instruct it to input an additional 10 words with the following segment:

```
        LAC ()       /CONTROL WORD
        ISA          /RAISE PRIORITY TO
                     /LOCK OUT INAD
                     /CHANGE INAD
                     /PARAMETERS
        DBK          /RESTORE PRIORITY TO
                     /ORIGINAL LEVEL
```

*Simulation of Hardware Interrupt* - A hardware interrupt may be simulated by

```
        LAC ()       /CONTROL WORD
        ISA          /RAISE TO HARDWARE PRIOR
                     /PRIORITY
        JMS INAD     /ENTER INAD
```

*Use of Software Levels* - The organization of a program on 5-levels might be as follows (in order of decreasing priority).

| | |
|---|---|
| Interrupt level 0 - | highest priority alarm conditions, computer or process malfunction |
| Interrupt level 1 - | control process A/D-D/A, sense and control input/output routines |
| Interrupt level 3 - | Teletype I/O routines for operator interface, operator can inquire or demand changes as required. |
| Interrupt level 4 - (Software) | FORTRAN subroutines to calculate process control in- |

TABLE 9-4 STATUS BITS ASSOCIATED WITH THE RPL INSTRUCTIONS

| AC Bit | Function |
|---|---|
| 0 | API enabled |
| 1 | Unused |
| 2 | Device requesting service on priority level 0 |
| 3 | Device requesting service on priority level 1 |
| 4 | Device requesting service on priority level 2 |
| 5 | Device requesting service on priority level 3 |
| 6 | Device requesting service on priority level 4 |
| 7 | Device requesting service on priority level 5 |
| 8 | Device requesting service on priority level 6 |
| 9 | Device requesting service on priority level 7 |
| 10 | Priority level 0 active |
| 11 | Priority level 1 active |
| 12 | Priority level 2 active |
| 13 | Priority level 3 active |
| 14 | Priority level 4 active |
| 15 | Priority level 5 active |
| 16 | Priority level 6 active |
| 17 | Priority level 7 active |

| | |
|---|---|
| Interrupt level 4 (continued) | - put/output data. Direct digital control routines. |
| Main Program | - lowest priority-operator interface programming, requested readouts, etc. |

### Queueing

High priority/high data rate/short access routines cannot perform complex calculations based on unusual conditions without holding off further data inputs. To perform the calculations, the high priority program segment must initiate a lower priority (interruptable) segment to perform the calculation. Since, in general, many data handling routines will be requesting calculations, there will have to be a queue of calculation jobs waiting to be performed at the software level. Each data handling routine must add its job to the appropriate queue and issue an interrupt request (ISA instruction) at the corresponding software priority level.

## DATA CHANNEL TRANSFERS

The data channel control offers a relatively high-speed data path (via the I/O bus) for the transfer of data in blocks between core memory and high data rate devices, such as DECtape and standard magnetic tape transports. The data channel control can service up to eight devices. The priority of service is established by the physical order in which the devices are interfaced to the I/O bus.

Device requests for data channel transfer action are honored at completion of the current instruction, provided that such completion is interruptable; i.e., the current instruction cannot be an IOT instruction or an XCT instruction. Each type prohibits interruption of the instruction sequence until the instruction which immediately follows the current instruction (and which is not itself an IOT or XCT instruction) completes its execution. A data channel request made during the current instruction has priority over an API or PI request made during the same interval. An in-process data channel transfer cannot be interrupted. Device requests for data channel transfer action which occur during an in-process data channel transfer will be honored at completion of the current channel operation on the basis of their priority relationship; i.e., a higher priority device will be serviced immediately while the lower priority device or devices wait for service.

A data channel device functions with processor-granted program breaks to interleave its data transfers with execution of the program in progress. These breaks suspend rather than interrupt the program's execution. The transfers are made via the MB and do not disturb the contents of other active registers in the processor (AC, PC, etc.). Data is read into memory in three machine cycles and out of memory in four cycles (the additional cycle allows I/O bus settling and the setting of control gates prior to the strobing of the data into the device's buffer register). The maximum data rate capacity of the data channel control lies between 160,000 and 220,000 words per second, depending on the mix of input and output transfers.

Each data channel device is associated with a unique pair of memory locations which function as a word counter (WC) register and a current address (CA) register for controlling data transfers made to and from the device. These registers must be initialized by the program prior to the device request for data transfer action. The register (memory location) assignments for the four standard data channels are as follows:

| *Data Channel* | *WC (octal)* | *CA (octal)* |
|---|---|---|
| 0 | 30 | 31 |
| 1 | 32 | 33 |
| 2 | 34 | 35 |
| 3 | 36 | 37 |
| 4-7 | Not Assigned | |

PDP-9/L systems programs presently provide for the assignment of DECtape to data channel 0 and the assignment of standard magnetic tape to channel 1 (refer to chapter 5 for discussions of these devices).

The WC register (whose memory address must be even and the lower numerical quantity of the data channel pair) is initialized to contain the number of words to be transferred; this number must be entered in 2s complement notation, as the WC counts up towards zero and indicates by the all-zero condition that the transfer has been completed. The CA register is initialized to contain the starting address minus one of the sequential block of memory locations which are to deliver data or receive data from the device.

When the processor honors a data channel request, the respective device transmits the address of its assigned WC register. During the first cycle of the channel transfer, the contents of the WC are incremented by one and the address of the CA register is established. In the second cycle, the contents of the CA are incremented by one to establish the effective address of the memory location delivering or receiving the data word. During the third cycle, or fourth cycle in the case of out transfers, the actual data transfer occurs.

The device normally continues to request data channel service until the WC overflows (goes to all-zeros), indicating that the initialized number of data words will have been transferred at the completion of the current data channel action. A signal sent to the device at this time may be used to initiate API or PI access of a subroutine for the purpose of again initializing the channel's WC and CA registers.* Because the block transfers are primarily automatic in nature, the programmer need only concern himself with providing the appropriate subroutines to initialize the data channel WC and CA registers and to initiate the device request for data channel service.

Since Data Channel requests are only honored at completion of instructions, the type of instruction in progress determines the waiting time until the interrupt is granted. The following considerations apply.

1. The IOT instruction and subsequent instruction are noninterruptable. The interrupt request will be honored at the completion of the instruction which follows the IOT.

2. The EAE instructions delay interruption until they complete. This may be as long as 17 microseconds.

3. The XCT instruction is noninterruptable. The interrupt request will be honored at the

*The "overflow" signal normally shuts down the device to prevent further transfer requests until WC and CA are re-initialized.

completion of the instruction referenced by the XCT.

4. Lower priority channel-interfaced devices wait for the completion of data transfers on the requesting higher priority channel. Hence, if four requests come up simultaneously, the lowest priority device may wait 12 microseconds.

Long XCT chains on sequential IOT instructions can lock out channel requests for indeterminate periods of times. These should be avoided in programs operating devices requiring fast response to their requests. EAE instructions requiring more than 12 microseconds are uncommon but possible. Unfortunately, requests tend to stack up during these waiting periods so that lower priority devices may wait even longer.

## ADD-TO-MEMORY CAPABILITY

When the Increment MB (memory) capability is used, the device data register is gated to the address lines using the ENA (1) level generated by a W104 module (see figure 14-2). A break is initiated in the usual manner by the W104 module. The ENA (1) level from this module should be used, through an inverter, to ground the INC MB control line. The overflow pulse I/O OVFLO should be gated with ENB (1) to produce a control pulse for the device indicating that a location has been incremented to zero.

A device utilizing the add-to-memory capability is connected to the DCH in the normal manner. The word count memory location is specified by the device and gated to the address lines by the W104 module. The data to be added is gated onto the I/O data bus in the usual fashion (see figure 14-3) with IOP2 during the CA cycle. ENB (1) is used to request both READ RQ and WRITE RQ. If there is an add overflow, a pulse, DATA OFLO, (200 nano, -3v) comes back on I/O bus line A27E. If desired, the sum may be gated into an external register with IOP4 during the DATA cycle.

## REAL-TIME CLOCK

The real-time clock produces clock pulses at the rate of one every 16.7 msec (or every 20 msec for 50 Hz powered systems). While the facility is enabled, each such pulse occurrence initiates a request for a program "break" at the completion of the current instruction. At the grant of the break, the contents of the clock counter register (memory location 00007) are incremented by one. This register is program initialized to contain the 2s complement of the desired number of clock pulses. Clock breaks continue to be requested until the register overflows (i.e., reaches the all-zeros condition). At this time, the clock flag is set to initiate interruption of the program in progress. The clock flag is interfaced to the program interrupt control and to the API system if the latter is included in the PDP-9. The real-time clock facility's incrementing of the clock register functions like a one cycle data channel transfer; i.e., at the break, the contents of 00007 are extracted, incremented and then rewritten in the same location, all within one cycle. Hence, the restrictions of "current instruction" apply here also.

The IOT instructions which are provided for use with the clock are listed in table 9-5.

TABLE 9-5 CLOCK IOT INSTRUCTIONS

| Mnemonic Symbol | Octal Code | Function |
|---|---|---|
| CLON | 700044 | Clock on. The real-time clock's incrementing of location 00007 is enabled and the clcok flag is cleared. |
| CLOF | 700004 | Clock off. The clock's incrementing of location 00007 is disabled and the clock flag is cleared. |
| CLSF | 700001 | Skip on clock flag. The next instruction is skipped. |

The CLK switch (console) must be in the down position to permit programmed control of the facility. In the up position, the facility remains disabled. Depressing the I/O RESET key (console) disables the facility and clears the clock flag.

While the facility is enabled, requests for clock breaks have priority of acceptance over API and PI requests. The first clock break may occur at any time up to 17 msec after the facility has been enabled. Subsequent breaks occur at the clock rate (60 or 50 times per second).

# CHAPTER 10
# CONTROLS AND INDICATORS

## OPERATOR CONSOLE

The PDP-9/L operator console (figure 10-1), an integral part of the main computer frame, includes a work shelf and a control console equipped with rocker switches, rotary switches, and indicators for operator control and monitoring of system operation. Typical console uses are:

Manual entry of instruction and/or data; start/stop/continue control of program execution.

Stepping through a program sequence by instruction or by machine cycle for debugging or maintenance purposes.

Visual examination of register contents and/or of system status.

Table 10-1 details the functional use of items on the control console. Indicators on the panel show the existing binary states of specific register bits and control flip-flops by being lighted for binary 1s and being extinguished for binary 0s. The operator console can be electrically locked by a control on the marginal-check panel to prevent undesired alteration of the program in progress. While the console is locked, operation of any switch, etc., will not affect the system.

## MARGINAL CHECK PANEL

The marginal check panel (figure 10-2) is concealed behind the red hinged panel on the front of the central processor. Table 10-2 details the functions of the panel-mounted controls and indicators.

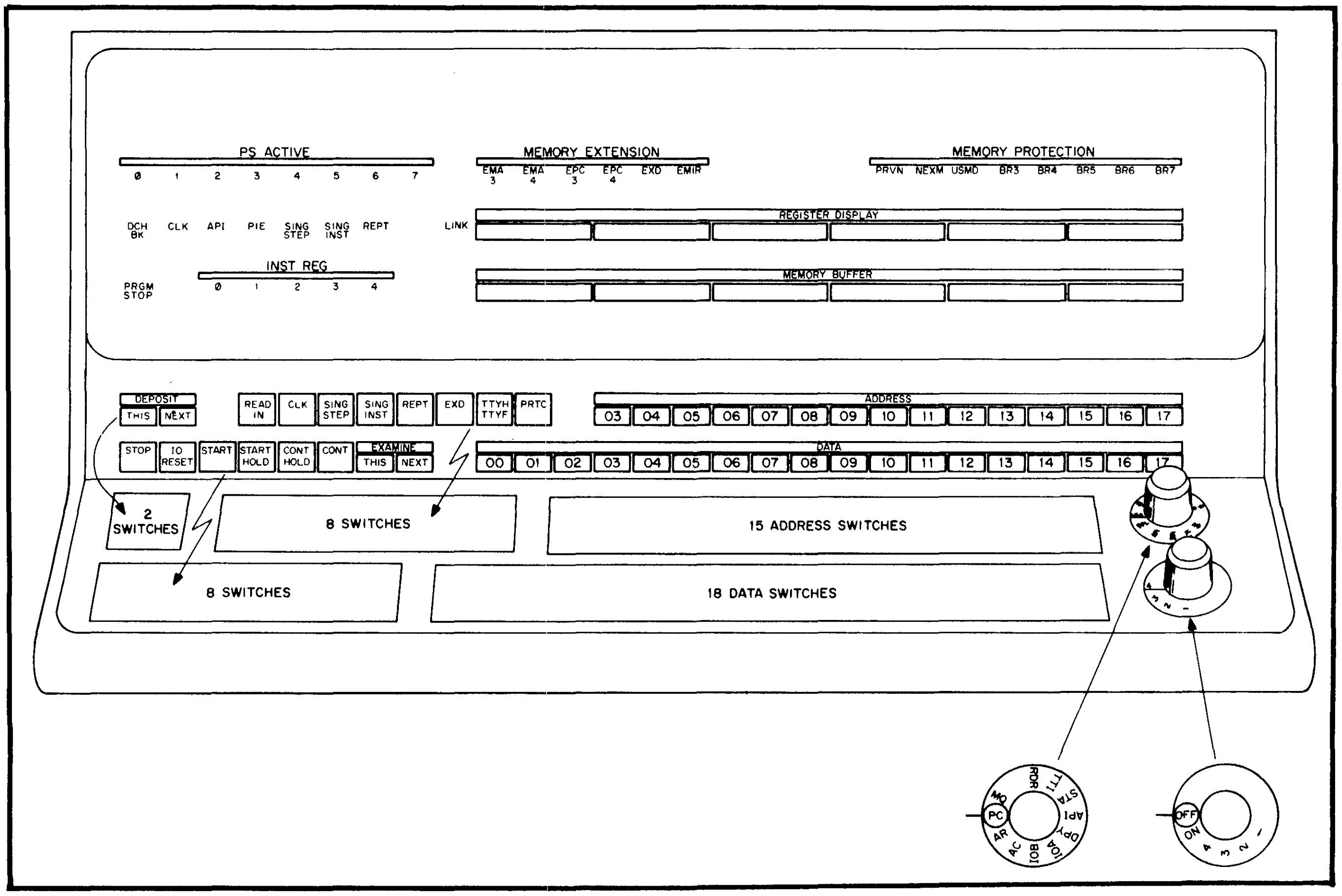


Figure 10-1. PDP-9/L Operator Console

TABLE 10-1 OPERATOR CONSOLE CONTROLS AND INDICATORS

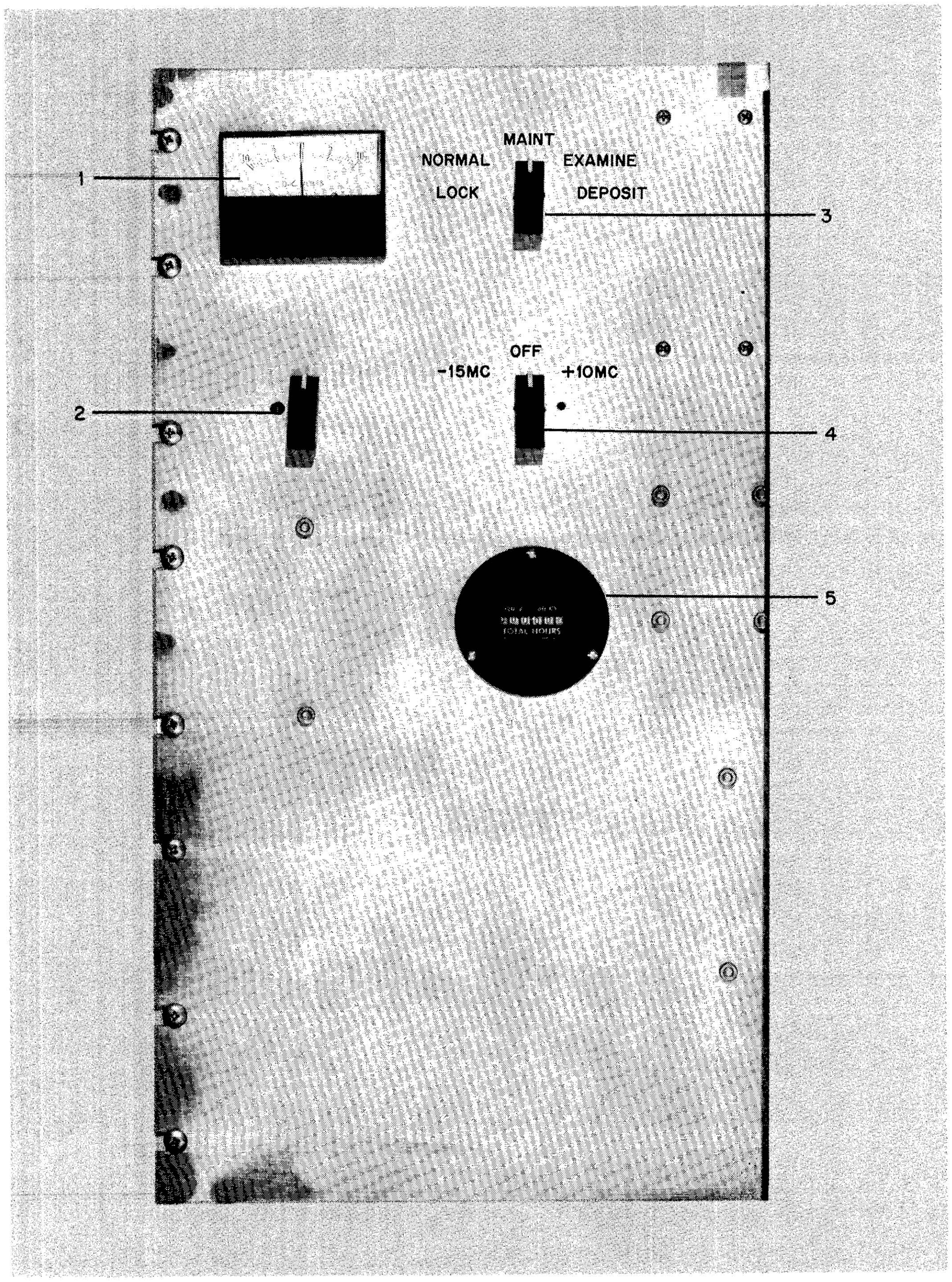
| Name | Function |
|---|---|
| START and START HOLD switches | Depressing START starts program execution at the location specified by the ADDRESS switches. The START HOLD switch is used for maintenance purposes. |
| IO RESET switch | Two positions: off (center) and operate (down, spring-loaded return). Depressing switch generates the CAF (clear all flags) instruction to clear all I/O device flags, clears the AC, MQ, and the link, turns off the real-time clock, program interrupt facility, and API system and disables the memory protection and extended memory modes. |
| STOP switch | Two positions: off (center) and operate (down, spring-loaded return). Operate halts program execution at completion of the current instruction. |
| CONT and CONT HOLD switches | Depressing CONT resumes program execution from the point at which it stopped. The CONT HOLD switch facilitates use of the REPT (repeat) function for the single instruction and single step provisions. |
| EXAMINE THIS and EXAMINE NEXT switches | Depressing the EXAMINE THIS switch transfers the contents of the memory location specified by the ADDRESS switches from memory to the MB. After the transfer, the contents of the ADDRESS switches appear in the AR as the address of the memory location examined. |
| | Depressing the EXAMINE NEXT switch increments the contents of the AR by one and transfers the contents of the newly addressed memory location from memory to the MB. EXAMINE NEXT facilitates monitoring of sequential memory locations as the ADDRESS switches need only be set to the lowest memory location. The use of EXAMINE THIS transfers the contents of this location to the MB and enters the lowest order address in the AR. Thereafter, use of EXAMINE NEXT step advances the addresses through the sequential memory locations. |
| DEPOSIT THIS and DEPOSIT NEXT switches | Depressing DEPOSIT THIS switch deposits the contents of the DATA switches in the memory location specified by the ADDRESS switches. After the transfer, the contents of the ADDRESS switches appear in the AR as the address of the memory location in which the data was entered. |
| | Depressing the DEPOSIT NEXT switch increments by one the AR contents, and deposits the contents of the DATA switches in the memory location specified by the new address. DEPOSIT NEXT facilitates the entering of data and/or instruction words in sequential memory locations as the ADDRESS switches need only be set to the lowest order address. |
| | The DEPOSIT THIS function deposits the DATA switch word in this location and transfers the address to the AR. Thereafter the DEPOSIT NEXT function step advances the addresses through the sequential memory locations. |
| READ IN switch | Two positions: off (center) and operate (down, spring-loaded return). Depress switch to initiate readin of paper tape punched in binary code (each set of three 6-bit lines read from tape forms one 18-bit computer word). Storage of words read in begins at the memory location specified by the ADDRESS switches. At the completion of tape readin, the computer reads the last word from core memory and executes it. Read-in occurs at the selected repeat speed. |

TABLE 10-1 OPERATOR CONSOLE CONTROLS AND INDICATORS (continued)

| Name | Function | |
|---|---|---|
| REPT (repeat) control and System ON-OFF switch | With REPT switch and CONT HOLD up, the control establishes one of four speeds at which single-step or single-instruction operations repeat without operator intervention. The repeating speeds range from approximately 40 microseconds (position 1) to 8 seconds (position 5). | |
| REGISTER DISPLAY control and display control and REGISTER DISPLAY indicators | Eleven-position switch: Each position interrogates a specific register and displays its contents in the REGISTER DISPLAY indicators. REGISTER DISPLAY indicators display the contents of selected register only when machine is stopped. Moving selection switch while program is running has no effect. The functions of the positions are as follows: | |
| | RDR | Display contents of the paper tape reader information buffer. |
| | TTI | Display contents of the teleprinter keyboard information buffer. |
| | STAT | Display status of flags for I/O devices connected to status reading facility of I/O system (see figure 9-1 for standard status bit assignments). |
| | API | Display activity of automatic priority interrupt system's four device-oriented priority levels. |
| | DPY | Display 34 display x, y buffers. The x buffer is displayed in the nine most significant REGISTER indicators; the y buffer is displayed in the nine least significant indicators. The least significant bit of each buffer is not displayed. |
| | IOA | Display 15-bit address word present on address lines of I/O bus for data channel and API operation. |
| | IOB | Display 18-bit data word present on data lines of I/O bus for program controlled and data channel data transfers. |
| | AC | Display contents of the AC. |
| | AR | Display contents of the AR. |
| | PC | Display contents of the PC and status bits as stored during this instruction. |
| | MQ | Display contents of the MQ. |
| PRTC switch and indicator | The up position causes the memory protection mode to be entered by operation of the START switch. In either position, the mode may be enabled or disabled by program control. While the console is locked, the switch is electrically in the down position, regardless of its actual position. The indicator is lit while the mode is in effect. (Memory protection is a system option.) | |
| EXD switch and indicator1 | The up position causes the extend mode of addressing to be entered by operation of the START switch. In either position, the mode may be enabled or disabled by program control. While the console is locked, the switch is electrically in the down position, regardless of its actual position. The indicator remains lit while the mode is in effect. (Extend mode is a system option.) | |
| CLK switch and indicator | The up position disables the real-time clock facility. The down position allows program control to enable or disable the clock. The indicator remains lit while the clock is enabled. While the console is locked, the switch is electrically in the down position, regardless of its actual position. | |

TABLE 10-1 OPERATOR CONSOLE CONTROLS AND INDICATORS (continued)

| Name | Function |
|---|---|
| SING STEP indicator and switch | The indicator lights when the associated switch is up. This enables the single-step mode which halts program execution at each machine cycle. Repetitive depressing of the CONT HOLD switch, while the mode is enabled, steps the program through the sequence one cycle at a time. When the console is locked, this switch is disabled. |
| SING INST indicator and switch | The indicator lights when the associated switch is up. This enables the single instruction mode which halts program execution at completion of each instruction. Repetitive depressing of the CONT HOLD switch, while the mode is enabled, steps the program through its sequence one instruction at a time. When the console is locked, this switch is disabled. |
| TTYH/TTYF switch | Determines whether teletype operation is half or full duplex. |
| REPT indicator and switch | The indicator lights when the associated switch is up. This enables the repeat function. This function causes operations initiated by actuation of CONT HOLD, EXAMINE NEXT, or DEPOSIT NEXT switches to repeat while the key remains in an operator position. The repeat speed control establishes the rate of repetition. |
| ADDRESS switches (0-17) | Establish a 18-bit core memory address to be entered in the PC by operation of the START switch, or in the AR by operation of the EXAMINE THIS or DEPOSIT THIS switch. Switch is placed up for a 1 bit and down for a 0 bit. The 18 switches to the right (0-17) set up the address of a location within an 8192-word memory block. The two switches to the left (0 and 0) are for extended memory addressing of locations, in up to three other 8192-word memory blocks of the system. |
| DATA switches | Establish an 18-bit data or instruction word to be read into memory by DEPOSIT THIS or DEPOSIT NEXT operation, or to be entered in the AC by a programmed LDS (load DATA switches) instruction. Up position of the switch is a binary 1; down position is a binary 0. |
| PRGM STOP indicator | Lights when the "run" flip-flop has been cleared to stop program execution. |
| INST REG | The five indicators reveal the contents of the IR, being lit for 1 bits and extinguished for 0 bits, to show the operation code of the instruction just executed or in progress, and indirect address occurrence. |
| DCH BK | Lights to indicate that data channel activity is in progress; i.e., data is being transferred between core memory and a data channel I/O device via the I/O bus. |
| PS ACTIVE indicators | Each indicator, relating to one of the API system's eight priority levels, individually lights to show the priority program interrupt request currently being serviced. Indicators 0, 1, 2, and 3 show activity resulting from device-initiated requests; indicators 4, 5, 6, and 7 show activity resulting from program-initiated requests. The priority levels for each set decrease in rank from left to the right with any device request having higher priority than any program request. |
| PIE indicator | Lights when the PI system has been enabled by program control. |
| API indicator | Lights when the PI system has been enabled by program control. |
| LINK indicator | Shows the content of the link register. |
| MEMORY BUFFER indicators | Shows the contents of the MB register. |



*LEGEND:*

1. Marginal-check voltmeter
2. Marginal-check voltage control
3. Maintenance switch
4. Marginal-check selector switch
5. Elapsed-time meter

Figure 10-2. Marginal-Check Panel

TABLE 10-2 MARGINAL CHECK PANEL CONTROLS AND INDICATORS

| Name | Function |
|---|---|
| Marginal-check voltmeter (1, figure 10-2) | Indicates the selected voltage output of the marginal check power supply. The center of the scale relates to the reference voltage slected, either +10 or -15 volts dc. Movement of the pointer to the right indicates an increase in magnitude for the marginal check voltage. |
| Marginal-check voltage control (2, figure 10-2) | Establishes the marginal check voltage level of the selected output. Voltage is increased with cw rotation. |
| Maintenance switch (3, figure 10-2) | Five positions:<br><br>LOCK - electrically locks the control console. With the console in the locked condition, operation of any console contrlo cannot affect the program in progress.<br><br>NORMAL - Control console is not locked; all controls may be used.<br><br>MAIN - With the switch in this position and the REPT switch (control console) in the up position, the built-in maintenance test program circulates a self-incrementing count through all active CPU registers to verify both their operation and the internal transfer paths. The program proceeds at the rate selected by the repeat speed control (control console).<br><br>EXAMINE - simulates the "examine" function. With the switch in this position, the CPU responds as if the EXAMINE THIS switch (control console) was being actuated at the rate selected by the repeat speed control (control console). With the REPT switch in the down (inoperative) position, each movement of the selector switch to position EXAMINE simulates an actuation of the EXAMINE THIS switch.<br><br>DEPOSIT - Simulates the "deposit" function. With the switch in this position and the REPT switch (control console) in the up position, the CPU responds as if the DEPOSIT key was being actuated at the rate selected by the repeat speed control (control console). With the REPT switch in the down (inoperative) position, each movement of the selector switch to position DEPOSIT THIS simulates an actuation of the DEPOSIT THIS switch. |
| Marginal-check selector switch (4, figure 10-2) | Three positions:<br><br>OFF - center<br><br>+10 MC - selects the +10-bolt output of the marginal check power supply.<br><br>-15 MC - selects the -15-volt output of the marginal check power supply. |
| Elapsed-time meter (5, figure 10-2) | Indicates, to the nearest tenth of an hour, the cumulative number of hours in which the system has been in the "power on" state. Meter counts from 00000.0 to 99999.9. |

# CHAPTER 11
# INTRODUCTION TO INTERFACING

## GENERAL

PDP-9/L offers a complete line of standard peripheral equipment and compatible interface controls. Selections from this line may be appended to a PDP-9/L system at any time without modification of the central processor unit. Where applications require that the system operate special purpose equipment or user-designed external devices, the PDP-9/L I/O control scheme affords simple and economical implementation of the necessary interfacing.

This section contains information of interest to the interface designer. It describes the PDP-9/L I/O facilities and defines the requirements for interfacing a device to them. The information presented includes: definitions of signal characteristics; timing relationships; and, where appropriate, schematic representations of typical interfaces configured from Digital's line of inexpensive FLIP CHIP hybrid integrated circuit modules. The PDP-9/L is designed to be compatible with the 2 mHz, R series FLIP CHIP modules.

Complete descriptions of all standard FLIP CHIP modules, compatible power supplies, and hardware (mounting panels, cables, etc.) can be found in the "Digital Logic Handbook", sent free of charge upon request to any of the Corporation's local offices. Items are readily available for low cost fabrication of external devices and/or special interface controls.

Customers are invited to consult Digital System and Application Engineering personnel for assistance in the design of devices and interface controls. Their knowledge of data processing disciplines and their applied experience in answering industrial and scientific needs offer fast resolution of data handling problems.

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 license to use, manufacture, or sell equipment constructed in accordance therewith.

## CIRCUIT MODULES FOR INTERFACING

The following FLIP CHIP circuit modules are typical of those recommended for use in fabricating device interfaces compatible with the PDP-9/L I/O facilities. Their input, output, and power requirements are as specified in the "Digital Logic Handbook".

| *Type Number* | *Function Name* |
|---|---|
| R107 | Inverter |
| R123 | Diode Gate |
| R200 | Flip-Flop |
| R201 | Flip-Flop |
| R202 | Flip-Flop, Dual |
| R203 | Flip-Flop, Triple |
| R204 | Flip-Flop, Quadruple |
| R205 | Flip-Flop, Dual |
| W103 | Device Selector |
| W500 | High Impedance Follower |
| W640 | Pulse Amplifier |
| B105 | Inverter |

## LOGIC SYMBOLS

The symbols used to indicate logic circuits and signals in the schematic illustrations included in this manual are defined below.

| *Symbol* | *Represents* |
|---|---|
| ——▶ | Negative or negative-going pulse. |
| ——▷ | Positive or positive-going pulse. |
| ——◆ | Negative level. |
| ——◇ | Positive level. |
| ——➤ | Direction of flow. |
| ●╱╲╱╲— | 15v load resistor clamped at -3v |

## I/O COMMUNICATIONS

The PDP-9/L offers facilities for communicating with external devices by program control and data channel. Devices using program controlled and data channel facilities interface to the I/O bus cables. Program controlled transfers may make use of direct, program interrupt (PI) or automatic priority interrupt (API) access of subroutines.

The data rates, the latency times (i.e., the time which a device must wait before its request for service is answered), and the representative degrees of efficiency for the PDP-9/L modes of I/O service are presented in table 11-1. The "direct" mode refers to program controlled transfers made to or from a single device without interruption by other I/O facilities.

Chapter 12 describes the total I/O bus system interface. Subsequent chapters describe the interface requirements for use of the program controlled transfer mode (including use of the program interrupt control, input/output skip facility, input/output read status facility, and the automatic priority interrupt option), and the data channel transfer mode.

TABLE 11-1 DATA TRANSFER RATES

| **Mode** | **Data Rate (maximum/typical)** | **Latency (maximum/typical)** | **Efficiency** |
|---|---|---|---|
| Direct | 100 kHz | | 10% |
| PI | 25 kHz/1 kHz | 45 microseconds for one device; 100+ microseconds for two devices/ 100 microseconds | 2% |
| API | 35 kHz/10 kHz | 30 microseconds/50 microseconds (average per channel) | 3% |
| DCH | 250 kHz/50 kHz | 20 microseconds/5 microseconds | 25% |

# CHAPTER 12
# THE I/O BUS

## GENERAL

The PDP-9/L I/O bus consists of cables which chain link all I/O device controls to a common interface point at the central processor (figure 12-1). Each device control must have input and output connector provisions to receive the bus and pass it on to the next device. The bus is the major I/O communication path for a PDP-9/L configuration. It provides signal lines for command and data transmissions arising from programmed transfers, data channel transfers, and operation of the multilevel automatic priority interrupt, program interrupt, I/O status read, and I/O skip facilities.

## PHYSICAL DESCRIPTION

Two cables of 36 two-wire pairs each comprise the I/O bus cables. The ground sides of all pairs are connected in common at the connectors. Female connector ends are used in device controls; male connectors are on both ends of the bus cables. The female connector is the DEC Type H800, available in the DEC Type 1943 (FLIP CHIP) Mounting Panel. The connector may be purchased separately. Two such connectors are needed for each device, one to bring in the I/O bus and one to send it out to the next device. Each such female connector receives the two male connectors of the two I/O bus cables.

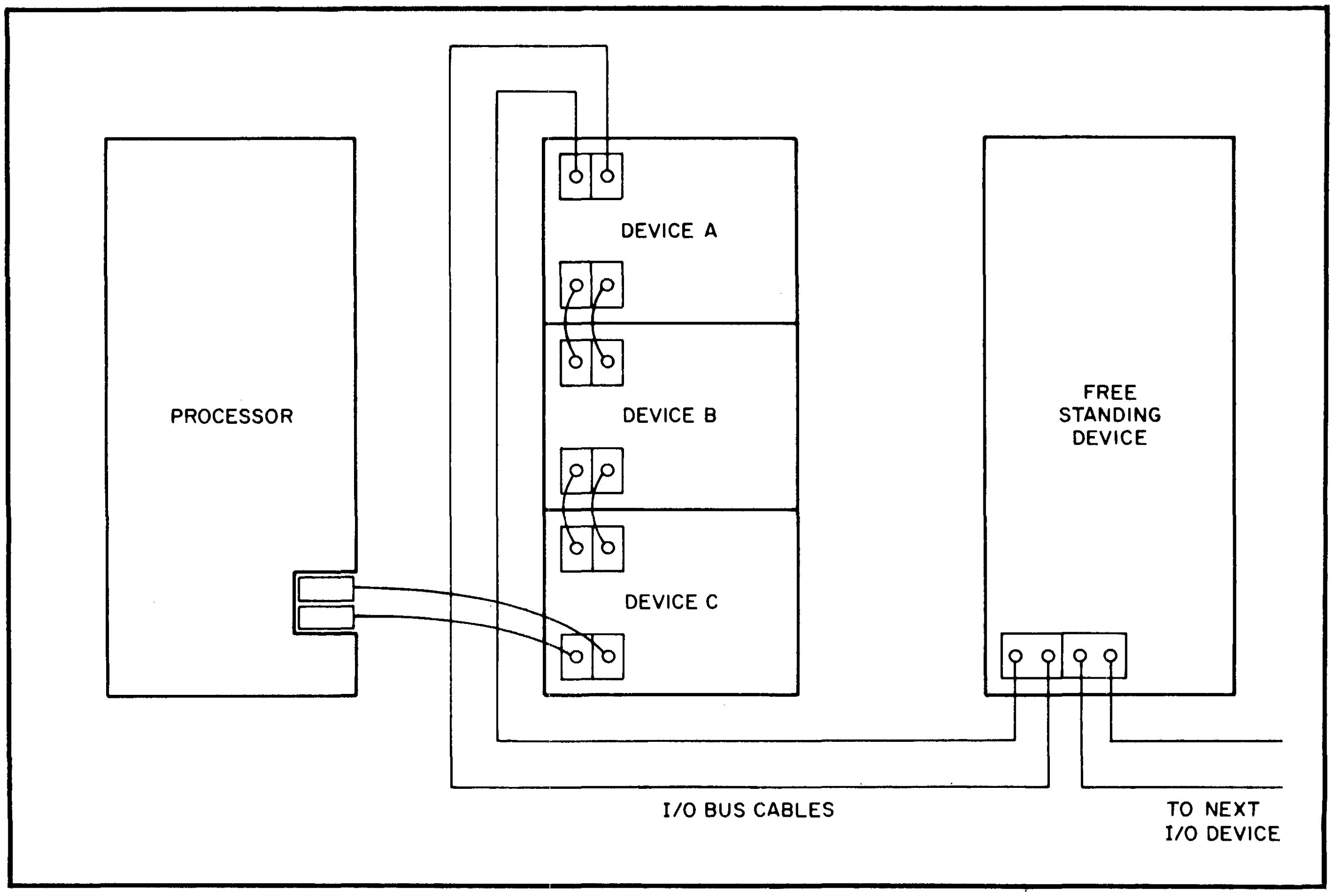


Figure 12-1. I/O Bus Connections

Each I/O bus cable is a DEC Type BC09 cable assembly; completed cables may be purchased. The connectors have a special locking provision which insures against accidental removal and, if properly strain relieved at each end, may be conveniently run between free standing cabinets. In free standing cabinet devices, the I/O connectors are located at the left (when viewed from wiring side) end of the lowest 1943 Mounting Panel. The "input" bus connector is to the left of the output connector. In devices requiring only a few mounting panels, the "input" bus connector is located at the lower left corner, the "output" connector at the upper left-hand corner.

## I/O POWER

The device input I/O bus cable connector must be supplied with -15 volts on each pin B. A total of 600 ma is required. The output cable connector need not be supplied with power.

## INTERFACE SIGNALS

The following describes the function of all I/O bus signal lines linking the central processor with the external I/O devices. Electrical characteristics and line terminating requirements are included. Figure 12-2 illustrates the interface. Maximum total length of the I/O bus is 50 feet (15.24 meters).

### Data Lines

Eighteen data lines constitute the bidirectional facility for transferring data in bytes up to 18 bits in length between the central processor and all I/O devices. Transfers are made on a dc basis with the processor or device allowing bus settling time before data on the lines is strobed into the receiving register. The data lines convey transfers between the AC and a selected device buffer register for data channel operation. The bidirectional characteristic requires that the device use unclamped collectors for data transmission to the processor. Emitter followers must be used in a device for data reception to avoid loading the bus on a dc basis. The data lines are terminated in the central processor by 15-ma clamped loads.

### Output Control Signals

Seven output control signals are generated within the processor to effect specific functions in a selected device. The signals are bus driven at the CPU-I/O interface.

*I/O Power Clear* - issued by power turn-on warm-up, by occurrence of a CAF instruction (mnemonic for clear all I/O flags), and by actuation of the I/O RESET key on the control console. The signal resets all flip-flops storing device-to-processor flag indications (e.g., ready, done, busy). It is developed as a 400 nsec, nominal width, negative-going pulse.

*I/O Sync* - issued every memory cycle. The signal may be used to synchronize I/O device control timing to execution of the program in progress. The signal is developed as a 400 nsec, nominal width, negative-going pulse.

*IOP1, IOP2, IOP4* - microprogrammable signals to effect IOT instruction-specified operation within a selected I/O device. Processor automatically generates IOP2 and IOP4 for, respectively, data channel input and output transfers. Although they may be used for any control function, the common uses of the IOPs are:

IOP1 - normally used in an I/O skip instruction to test a device flag. It may be used as a command pulse, but it cannot be used to initiate loading of or reading from a device buffer register.

IOP2 - usually used to effect transfer of data from a selected device to the processor, or to clear a device register. It may not be used to determine a skip condition.

IOP4 - usually used to effect transfer of data from the processor to a selected device register. It may not be used to determine a skip condition or to effect transfer of data from a selected device to the processor.

The IOP signals occur as 1 microsecond, nominal width, negative-going pulses.

*Read Status* - issued by execution of the IORS instruction (mnemonic for input/output read status). Loads the AC with an 18-bit word containing device flag indications for devices interfaced to read status facility. The signal occurs as a 1 microsecond, nominal width, negative-going pulse. The signal also occurs when the REGISTER DISPLAY switch (console) is placed in the STATUS position and the processor is stopped.

*Overflow* - issued during the first cycle of a data channel transfer if the contents (2s complement) of the word counter assigned to the currently active data channel device become zero when incremented. This indicates that

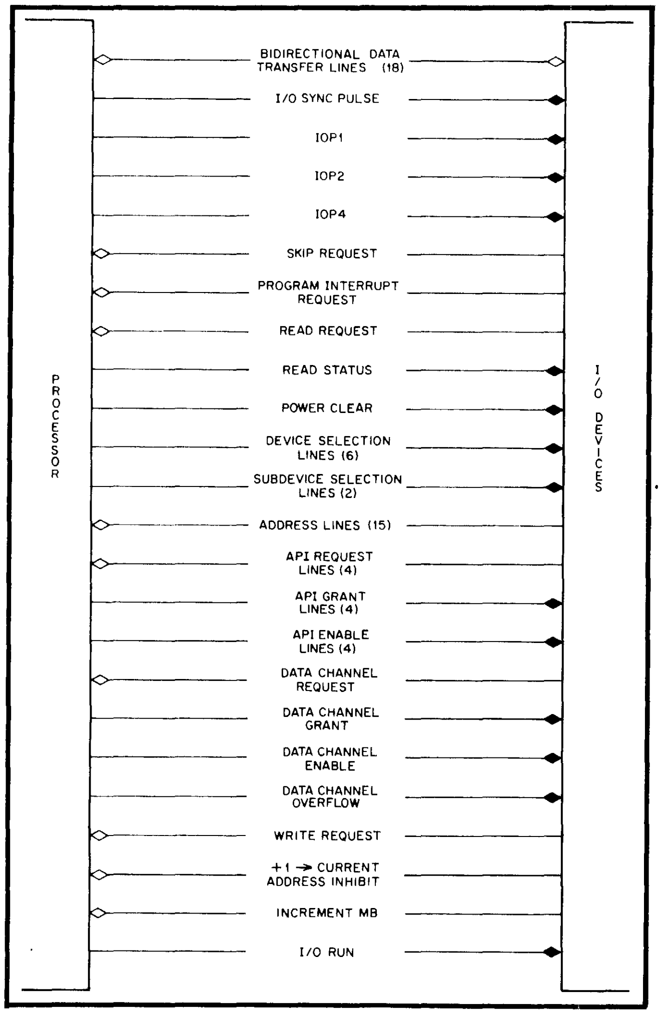


Figure 12-2. I/O Bus Interface

the program specified number of words will have been transferred at completion of the data channel transfer in progress. It is normally used to turn off the respective device, preventing further data channel action by that device until a service subroutine reinitializes the channel word counter and current address registers, and the program turns on the device request flag. The overflow signal may also be used to initiate a program interrupt through the program interrupt or automatic priority interrupt facilities for access to the initializing subroutine. The signal occurs as a 400 nsec, nominal width, negative-going pulse.

## Device Selection Levels

Detection of the current instruction as being an IOT causes the bit pattern placed in $MB_{6-13}$ at the fetch of the instruction to be bus driven and sent via eight bus lines to Type W103 Device Selection modules, contained in the control logic for each device. These eight levels form a 6-bit device selection code, DS0-DS5 (relating to $MB_{6-11}$) and a subdevice or mode select code extension, SD0 and SD1 (relating to $MB_{12-13}$). Assertion, or binary 1, is defined as a -3v. Negation, or binary 0, is defined as ground level. Each W103 is configured for response to only one of the 64 possible DS codes. Cooperating pairs of W103s permit unique response to any of the 256 DS-plus-SD codes. Each selection code configured in a device permits the internal generation of up to three associated commands through the W103 ANDing of IOPs and the device selection code.

## I/O RUN

The I/O run signal is available at the interface for use as the interface designer requires. This bus driven level switches to the -3v level and remains there while the "run" flip-flop in the CPU is set. A ground level indicates that the "run" flip-flop has been cleared.

## Input Control Levels

Six input control level signals arrive at the I/O control section in the central processor. These levels are at ground for assertion and at -3v for negation. Each signal line is terminated in the processor with a 15 ma clamped load. The line must be driven from the unclamped collector of a saturated transistor whose emitter is grounded. The individual functions of the input control levels are:

*Skip Request* - the return of this level to the processor indicates that an IOT instruction test for a skip condition in a selected device has been satisfied (e.g., a test of ready status). The PC is subsequently incremented by one to effect a skip of the next instruction of the program in progress.

*Program Interrupt Request* - a device delivers this level to request interruption of the program in progress. The program traps to location 00000 when no I/O transfer action of higher priority is in progress. The instruction resident in location 00001 is fetched and executed. This instruction is usually a JMP to a subroutine which determines through a search process ("skip chain") the device making the program interrupt request. Access is then made of the appropriate service subroutine. Up to 64 devices may be interfaced to the program interrupt request line. The limiting factor is solely the program overhead incurred in the search for the requesting device.

*Read Request* - this level requests that the processor execute a read transfer of device-offered data word.

*Write Request* - this level requests that the processor execute a data channel write transfer of a data word into the selected device's information register.

*MB Increment* - this level requests that the processor increment by one the contents of the memory location addressed by the 15-bit address on the I/O bus address lines. The provision is available when the Type KH09A Add-to-Memory option is included.

*Current Address Inhibit* - this is a special signal line required by devices which automatically search for records, etc. Typical are DECtape and magnetic tape. The presence of the level inhibits normal incrementing of the device-assigned current address register during a data channel transfer.

**Multiplexed Control Lines**

Fifteen control lines, constituting five multiplexed subsets of three lines each, provide processor-device control information paths for the multiplexed data channel and the four priority levels on which the automatic priority interrupt option processes device channel requests for service. The functions of the lines in the subsets are as follows:

*Request* - a device transmits a service request to the processor via the appropriate request line. Each request line is terminated in the processor by a 15-ma clamped load. The line must be driven to ground for assertion by an unclamped collector of a saturated transistor whose emitter is grounded.

*Grant* - the processor indicates a grant of the service request by driving the associated grant line negative. All grant lines are buffered by bus driver modules in the processor.

*Enable* - the enable signal controls the priority order for answering service requests of devices interfaced to the data channel control or to one of the API's 28 device channels. Each API channel may be uniquely assigned to one device for fast access of the appropriate service subroutine, or interfaced to any number of devices. The latter case requires a search subroutine to determine the requesting device. Priority for a channel (data or API) is allocated in descending order from the device nearest the processor I/O bus interface. Occurrence of an enable signal permits service of the requesting device with the highest channel priority and inhibits all lower priority devices from making requests during the interval of service. A bus driver module in the processor buffers each enable line.

**Address Lines**

Fifteen lines, of which only the least significant six are normally used, constitute an input bus for the devices which must deliver address data to the processor. The lines are terminated in the processor by 15-ma clamped loads. Each line must be driven to ground for assertion by an unclamped collector of a saturated transistor whose emitter is grounded. There are two uses for the address bus:

1. When a device interfaced to the multilevel, automatic priority interrupt option receives a processor grant of its interrupt request, it delivers to the processor a hardware-defined address, relating to its API channel assignment. This channel address indicates the unique entry point to the device's service subroutine. The instruction resident in the addressed memory location is always a JMS (or JMS I or XCT of a JMS), offering fast access to the appropriate service routine.

2. When a data channel device receives a processor grant of its transfer request, it delivers to the processor a hardware-defined address, relating to the memory location of the assigned channel word counter register.

**Driving Address and Data Lines**

Connection to the Address Lines and Data Lines is made by AND gates without clamp loads. Each gate must be capable of driving 30 mils at ground. Suggested gates include the S123 and the R111. The R111 requires a 2 mil clamp load tied to its input mode to deliver the required current.

## I/O BUS INTERFACE SUMMARY

The following summarizes the I/O bus interface at the processor. Figure 12-3 illustrates the key for determining the connector and associated pin for each bus line. Provision is made for two I/O bus connections at the computer; I/O block No. 1, and I/O block No. 2.

Reference Symbols:

CO - Collector Output, no clamped load, normally Type R111 or S123. Can drive a 30-ma load at ground. 0-ma at -3v.

BD - Bus Driven output. Can drive 25-ma load at ground, 7-ma load at -3v.

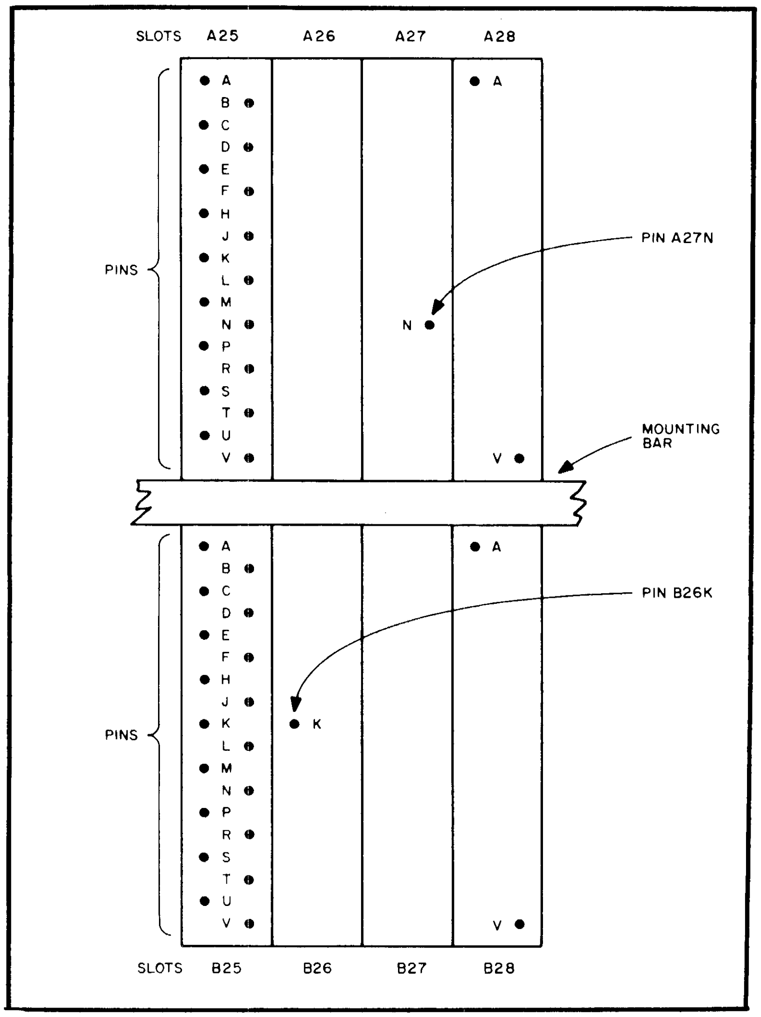


Figure 12-3 Interface Connectors and Pins

TABLE 12-1 I/O BUS INTERFACE CHART

| I/O Block No. 1 | I/O Block No. 2 | Type | Assertion | Signal Name |
|---|---|---|---|---|
| A25D | A29D | CO | ◇ | I/O BUS 00 |
| A25E | A29E | CO | ◇ | I/O BUS 01 |
| A25H | A29H | CO | ◇ | I/O BUS 02 |
| A25K | A29K | CO | ◇ | I/O BUS 03 |
| A25M | A29M | CO | ◇ | I/O BUS 04 |
| A25P | A29P | CO | ◇ | I/O BUS 05 |
| A25S | A29S | CO | ◇ | I/O BUS 06 |
| A25T | A29T | CO | ◇ | I/O BUS 07 |
| A25V | A29V | CO | ◇ | I/O BUS 08 |
| A26D | A30D | BD | ◆ | I/O SYNC |
| A26E | A30E | BD | ◆ | IOP 1 |
| A26H | A30H | BD | ◆ | IOP 2 |
| A26K | A30K | BD | ◆ | IOP 4 |
| A26M | A30M | CO | ◇ | SKIP RQ |
| A26P | A30P | CO | ◇ | PROG INT RQ |
| A26S | A30S | CO | ◇ | READ RQ |
| A26T | A30T | BD | ◆ | RD STATUS |
| A26V | A30V | BD | ◆ | I/O PWR CLR |

TABLE 12-1 I/O BUS INTERFACE CHART (Continued)

| I/O Block No. 1 | I/O Block No. 2 | Type | Assertion | Signal Name |
|---|---|---|---|---|
| A27D | A31D | BD | ◆ | I/O RUN (1) |
| A27E | A31E | – | | ADD OVFLO |
| A27H | A31H | BD | ◆ | I/O OVFLO |
| A27K | A31K | CO | ◇ | I/O ADDR 03 |
| A27M | A31M | CO | ◇ | I/O ADDR 04 |
| A27P | A31P | CO | ◇ | I/O ADDR 05 |
| A27S | A31S | CO | ◇ | I/O ADDR 06 |
| A27T | A31T | CO | ◇ | I/O ADDR 07 |
| A27V | A31V | CO | ◇ | I/O ADDR 08 |
| A28D | A32D | CO | ◇ | WRITE RQ |
| A28E | A32E | CO | ◇ | INC MB |
| A28H | A32H | CO | ◇ | + 1→CA INH |
| A28K | A32K | CO | ◇ | API 0 RQ |
| A28M | A32M | BD | ◆ | API 0 GR (1) |
| A28P | A32P | BD | ◆ | API 0 EN |
| A28S | A32S | CO | ◇ | API 1 RQ |
| A28T | A32T | BD | ◆ | API 1 GR (1) |
| A28V | A32V | BD | ◆ | API 1 EN |
| B25D | B29D | CO | ◇ | I/O BUS 09 |
| B25E | B29E | CO | ◇ | I/O BUS 10 |
| B25H | B29H | CO | ◇ | I/O BUS 11 |
| B25K | B29K | CO | ◇ | I/O BUS 12 |
| B25M | B29M | CO | ◇ | I/O BUS 13 |
| B25P | B29P | CO | ◇ | I/O BUS 14 |
| B25S | B29S | CO | ◇ | I/O BUS 15 |
| B25T | B29T | CO | ◇ | I/O BUS 16 |
| B25V | B29V | CO | ◇ | I/O BUS 17 |
| B26D | B30D | BD | ◆ | DS 0 |
| B26E | B30E | BD | ◆ | DS 1 |
| B26H | B30H | BD | ◆ | DS 2 |
| B26K | B30K | BD | ◆ | DS 3 |
| B26M | B30M | BD | ◆ | DS 4 |
| B26P | B30P | BD | ◆ | DS 5 |
| B26S | B30S | – | | SPARE |
| B26T | B30T | BD | ◆ | SD 0 |
| B26V | B30V | BD | ◆ | SD 1 |
| B27D | B31D | CO | ◇ | I/O ADDR 09 |
| B27E | B31E | CO | ◇ | I/O ADDR 10 |
| B27H | B31H | CO | ◇ | I/O ADDR 11 |
| B27K | B31K | CO | ◇ | I/O ADDR 12 |
| B27M | B31M | CO | ◇ | I/O ADDR 13 |
| B27P | B31P | CO | ◇ | I/O ADDR 14 |
| B27S | B31S | CO | ◇ | I/O ADDR 15 |
| B27T | B31T | CO | ◇ | I/O ADDR 16 |
| B27V | B31V | CO | ◇ | I/O ADDR 17 |

TABLE 12-1 I/O BUS INTERFACE CHART (Continued)

| I/O Block No. 1 | I/O Block No. 2 | Type | Assertion | Signal Name |
|---|---|---|---|---|
| B28D | B32D | CO | ◇ | API 2 RQ |
| B28E | B32E | BD | ◆ | API 2 GR (1) |
| B28H | B32H | BD | ◆ | API 2 EN |
| B28K | B32K | CO | ◇ | API 3 RQ |
| B28M | B32M | BD | ◆ | API 3 GR (1) |
| B28P | B32P | BD | ◆ | API 3 EN |
| B28S | B32S | CO | ◇ | DCH RQ |
| B28T | B32T | BD | ◆ | DCH GR (1) |
| B28V | B32V | BD | ◆ | DCH EN |

NOTE: All pins C, F, J, L, N, R, U, should be connected together and wired into the ground mesh of the device.

# CHAPTER 13
# PROGRAM CONTROLLED TRANSFERS

## GENERAL

The majority of I/O transfers occur under control of the program, taking advantage of control elements present both in the computer and in the device controls interfacing to the I/O bus. Program-controlled transfers require more computer and actual time than data channel and direct memory access transfers, but the simplicity and inherent lower cost of the device controls coupled with the high speed of the computer relative to the operational speed of most peripheral devices offset this extra cost in time.

All programmed I/O transfers take place through the accumulator (AC) in bytes up to 18 bits in length. In transfers within the central processor, and between the processor and core memory, data are processed as 18-bit words, the sole addressable unit in the PDP-9/L. For bytes of less than 18 bits, unused bits in the data word normally remain zeroed. Programming techniques of masking and shifting the contents of words are also used to pack and unpack bytes of less than 18 bits, to reduce core memory storage requirements.

The rate of programmed transfers is a function of the device characteristics and the program manipulation required for each data byte transferred. The IOT instruction capability of the PDP-9/L allows programmed control of up to 256 devices, as well as the generation of up to three unique commands for each of the 256 possible device selection codes. Devices requiring more than three internal commands are assigned additional device selection codes.

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

1. The ability of each device to sample the select code generated by the computer during IOT instructions and, when selected, to be capable of producing sequential command pulses in accordance with computer-generated IOP pulses. Circuits performing these functions in peripheral devices are called device selectors (DS). A single double-sized module, the W103, provides all of these functions.

2. Each device receiving output data from the computer must contain gating circuits at the input of a receiving register capable of strobing the data on the I/O bus into the register when triggered by a command pulse from the DS. Such gates are called device input gates.

3. Each device supplying input data to the computer must contain gating circuits at the output of the transmitting register capable of strobing the information from the output register to the I/O bus, and furnishing a read request signal level to the computer when triggered by a command pulse from the DS. Such gates are called device output gates.

4. Each device must contain a busy/done flag (flip-flop) and gating circuits that can output a signal to the computer input/output skip bus upon command from the DS. The flag is set to indicate that the device is ready to transfer another byte of information.

## INPUT/OUTPUT TRANSFER INSTRUCTIONS

Input/output transfer (IOT) instructions initiate transmission of signals through the I/O bus to control peripheral devices, sense their status, by means of the I/O skip facility, and effect programmed transfers between them and the processor. A PDP-9/L IOT instruction has the format shown in figure 13-1.

The PDP-9/L IOT instruction has the following characteristics:

1. An operation code of $70_8$.

2. An 8-bit device selection code to discriminate among up to 256 peripheral devices (selection logic in a device's I/O bus interface responds only to its preassigned code). In normal practice, bits 6 through 11 perform the primary device discrimination among up to 64 devices, with bits 12 and 13 coded to select an operational mode or subdevice.

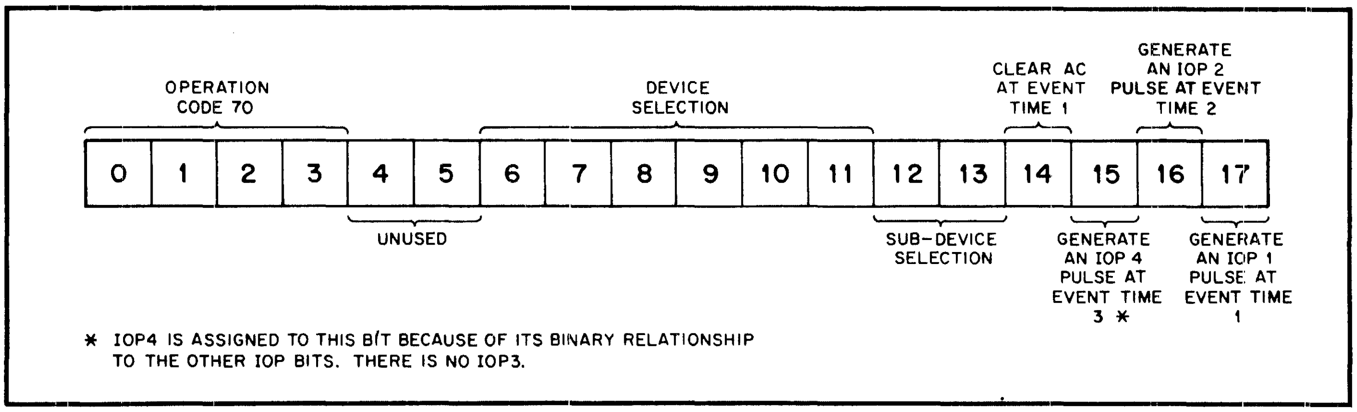

OPERATION CODE 70
DEVICE SELECTION
CLEAR AC AT EVENT TIME 1
GENERATE AN IOP 2 PULSE AT EVENT TIME 2
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
UNUSED
SUB-DEVICE SELECTION
GENERATE AN IOP 4 PULSE AT EVENT TIME 3 *
GENERATE AN IOP 1 PULSE AT EVENT TIME 1
* IOP4 IS ASSIGNED TO THIS BIT BECAUSE OF ITS BINARY RELATIONSHIP TO THE OTHER IOP BITS. THERE IS NO IOP3.

Figure 13-1 PDP-9/L IOT Instruction Format

Table 13-1 indicates the device selection codes assigned to standard PDP-9/L device and facilities. Codes not so used are available for assignment to user-designed interfaces or special-purpose equipment.

3. A command code (bits 14 through 17) capable of being microprogrammed to clear the AC, and issue up to three pulses through the I/O bus.

Any IOT instruction may be microcoded to produce more than one IOP pulse by setting one, two, or three bits of bits 15-17 of the instruction word to a 1. The resulting device IOT pulses appear in the time sequences defined by the IOT timing diagram (see figures 13-2 and 13-3).

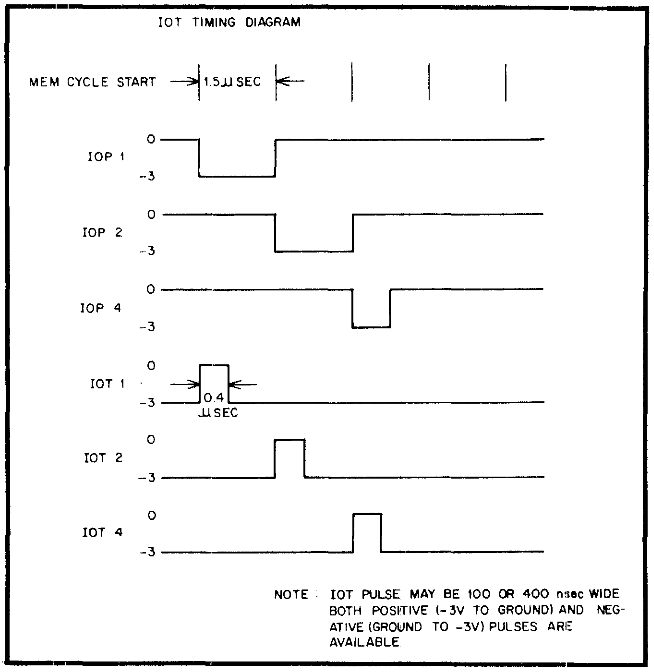

IOT TIMING DIAGRAM
MEM CYCLE START
1.5µ SEC
IOP 1
IOP 2
IOP 4
IOT 1
IOT 2
IOT 4
0
-3
0.4 µSEC
NOTE: IOT PULSE MAY BE 100 OR 400 nsec WIDE BOTH POSITIVE (-3V TO GROUND) AND NEGATIVE (GROUND TO -3V) PULSES ARE AVAILABLE

Figure 13-2 IOT Timing Diagram

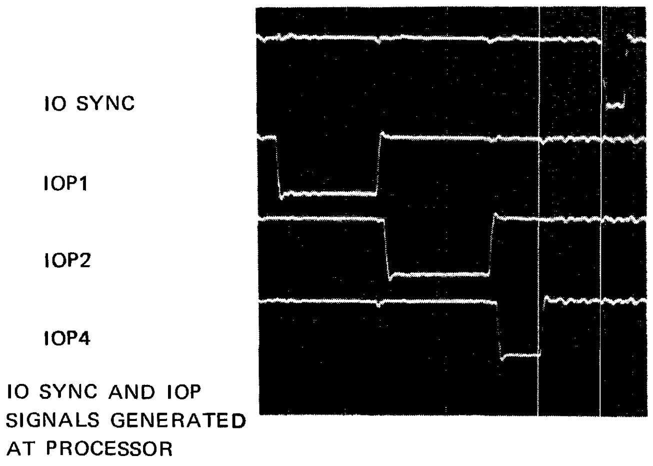


Figure 13-3 IOT Pulse Waveforms

The MB bits corresponding to the device selection levels are buffered by B213 Bus Drivers. Bus Driver outputs are labeled $DS_0$-$DS_5$ (device selector), with the following correspondence:

| | |
|---|---|
| $MB_6$ | $DS_0$ |
| $MB_7$ | $DS_1$ |
| $MB_8$ | $DS_2$ |
| $MB_9$ | $DS_3$ |
| $MB_{10}$ | $DS_4$ |
| $MB_{11}$ | $DS_5$ |

MB bits 12 and 13 are also bus driven, and available at the interface as sub-device bits (SD). These are labeled:

| | |
|---|---|
| $MB_{12}$ | $SD_0$ |
| $MB_{13}$ | $SD_1$ |

Each peripheral device contains *at least* one Device Selector module W103. This module produces specific device IOT pulses upon receipt of a unique 6-bit device selection code (device number) and the IOP pulses.

TABLE 13-1. ASSIGNED PDP-9/L DEVICE SELECTION CODES

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 00 1 RT Clock<br>2 Prog. Interrupt<br>4 RT Clock | 10 | 20 Memory<br>Increment<br>KH09A | 30 | 40 LT09A<br>Line 1<br>Teleprinter | 50 | 60 | 70 |
| 01 Standard<br>Perforated-<br>Tape Reader | 11 Analog-to-<br>Digital or<br>Digital-to-<br>Analog<br>Converter | 21 Relay Buffer<br>DR09A | 31 | 41 LT09A<br>Line 1<br>Keyboard | 51 | 61 | 71 |
| 02 Standard<br>Perforated-<br>Tape Punch | 12 A/D or<br>D/A<br>Converter | 22 | 32 | 42 LT09A<br>Line 2<br>Teleprinter | 52 | 62 | 72 |
| 03 1 Keyboard<br>2 Keyboard<br>4 IORS | 13 A/D<br>Converter | 23 | 33 1 33 KSR Skip<br>2 Clear All<br>Flags<br>4 DBR, DBR | 43 LT09A<br>Line 2<br>Keyboard | 53 | 63 | 73 Tape Control<br>TC59 |
| 04 Teleprinter | 14 | 24 Incremental<br>Plotter<br>Control<br>Type 350 | 34 | 44 LT09A<br>Line 3<br>Teleprinter | 54 | 64 | 74 Tape Control<br>TC59 |
| 05 Displays<br>Types 34F,<br>30D, | 15 | 25 DP09A | 35 | 45 LT09A<br>Line 3<br>Keyboard | 55 Automatic<br>Priority<br>Interrupt<br>KF09A | 65 Automatic<br>Line Printer<br>Type 647 | 75 DECtape<br>Control<br>TC02 |
| 06 Displays | 16 | 26 DP09A | 36 | 46 LT09A<br>Line 4<br>Teleprinter | 56 | 66 Automatic<br>Line Printer<br>Type 647 | 76 DECtape<br>Control<br>TC02 |
| 07 Display<br>and<br>Light Pen | 17 Memory<br>Protection<br>KX09A | 27 Memory<br>Parity<br>MP09 | 37 | 47 LT09A<br>Line 4<br>Keyboard | 57 | 67 Card Reader<br>Type CR01E<br>or Type CR02B | 77 Memory<br>Extension<br>KG09A |

For example the instruction:

703401

applies $34_8$ to the I/O bus device selection lines. The device selector module in device number 34 responds and, at IOP1 time, issues IOT 3401. The device number of any W103 is determined by clipping out diodes from the board. (See figure 13-4.)

The usual use of the three IOPs is given below.

IOT Pulse 1 - normally used in an I/O skip instruction to test a device flag. May be

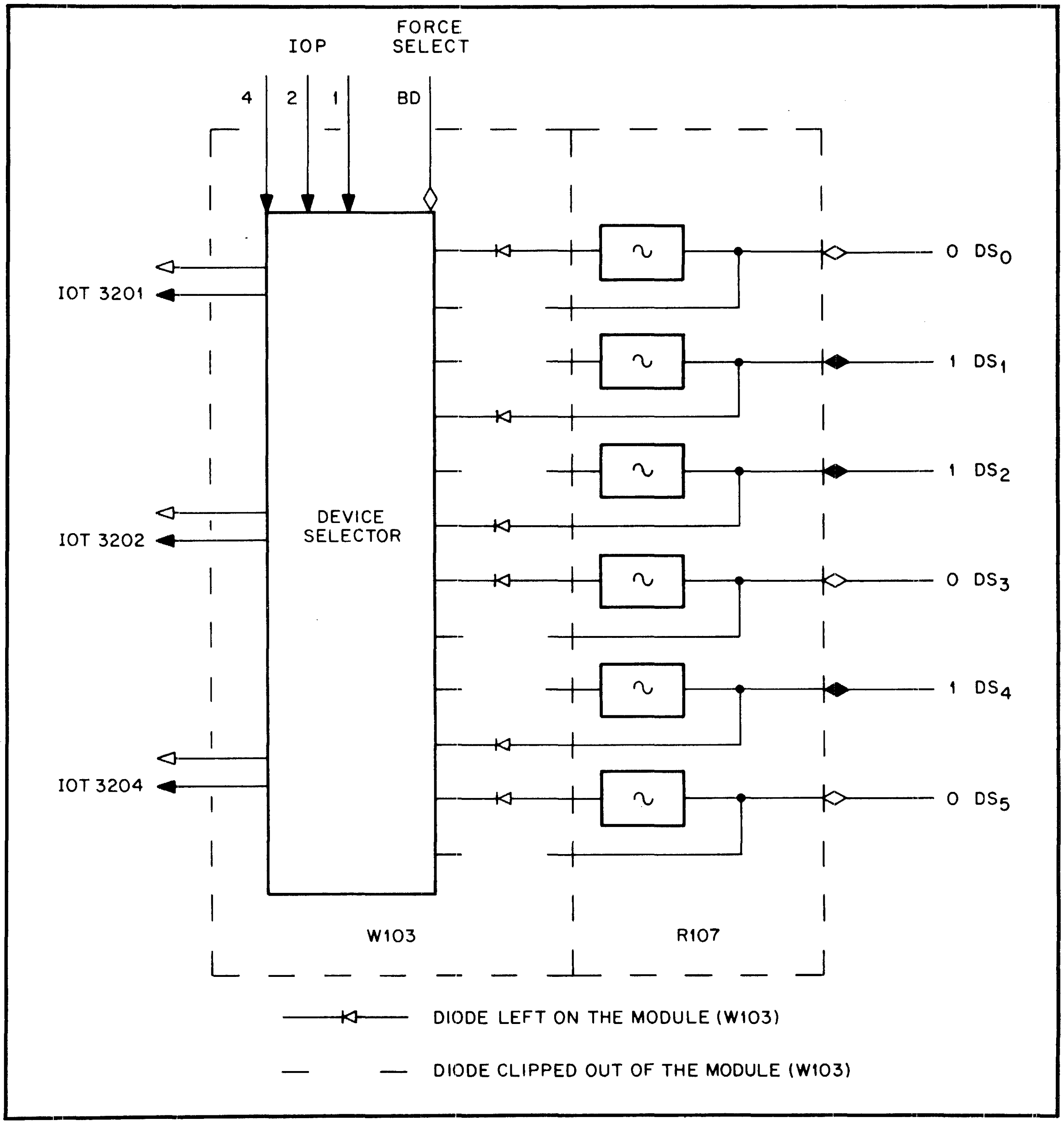


Figure 13-4. Device Selector Configuration

used as a command pulse, but not to initiate either a "load" or "read" from a device.

IOT Pulse 2 - usually used to transfer data from the device to the computer, or to clear a register. May not be used to determine a "skip" condition.

IOT Pulse 4 - usually used to transfer data from the computer to the device. May not be used to determine a "skip" condition or to effect transfer of data from a selected device to the processor.

## Reading a Device Buffer into the AC

The reading of a device buffer into the computer is accomplished in the following manner:

1. IOT Pulse 2 is generated in the device selector module, and used to trigger a W640 Pulse Amplifier (which must be in each input device) jumpered to produce 1-microsecond pulses.

2. This 1-microsecond pulse is used to gate the device data buffer onto the I/O data bus.

3. Simultaneously a read request positive pulse is generated (by an R111 or S123) on the read request line.

4. The central processor receives the request signal and allows time for the data bus to settle completely.

5. The I/O bus is strobed into the AC.

*The output of the W640 ought to be a 1-microsecond pulse, which is regarded by the I/O bus system as a level. Either pulse or level notation may be used, provided this definition is kept in mind.

Since the I/O bus is ORed into the AC (figure 13-4), the read IOT instruction is usually micro programmed to clear the AC prior to reading (i.e., MB14=1). I/O waveforms are shown on figure 13-6.

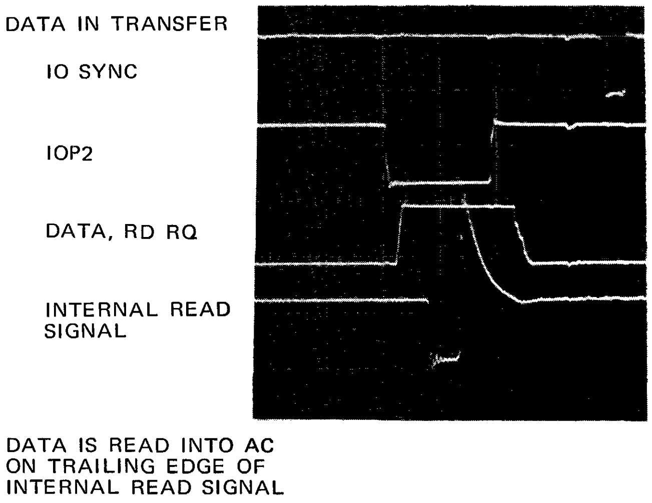


Figure 13-6 I/O Signals from Buffer to I/O Bus

## Loading a Device Buffer from the AC

The loading of a device buffer from the AC (figure 13-7 and 13-8), is usually accomplished by, but not restricted to, the following two steps. The first IOT clears the device buffer, and the second IOT ORs the contents of the AC into the cleared buffer. For such a transfer, the details are as follows:

1. Prior to the end of the first cycle the AC is placed onto the I/O bus. Note that in any output transfer, IOP1 must not be used to transfer the AC to an external register.

2. IOT2 is generated in the Device Selector module W103, and transmitted to all "clear" inputs of the flip-flops in the device buffers.

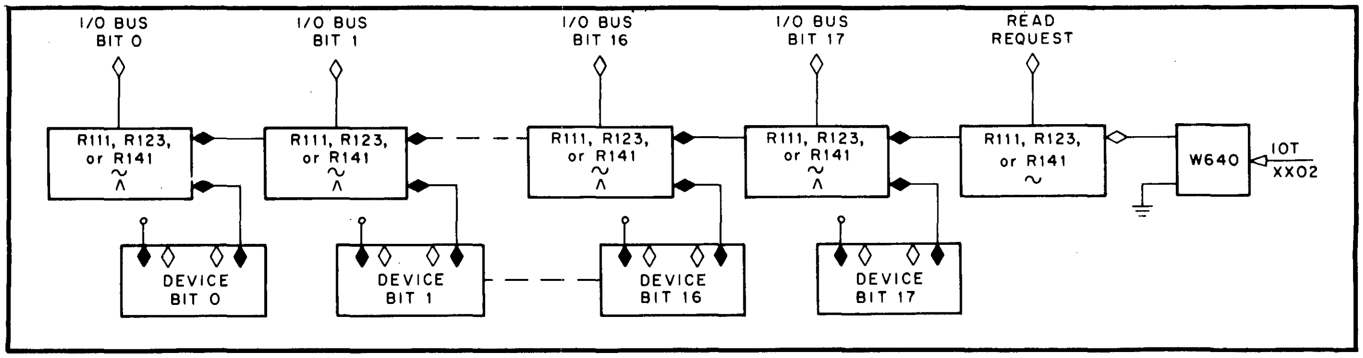


Figure 13-5 Loading the AC from a Device Buffer

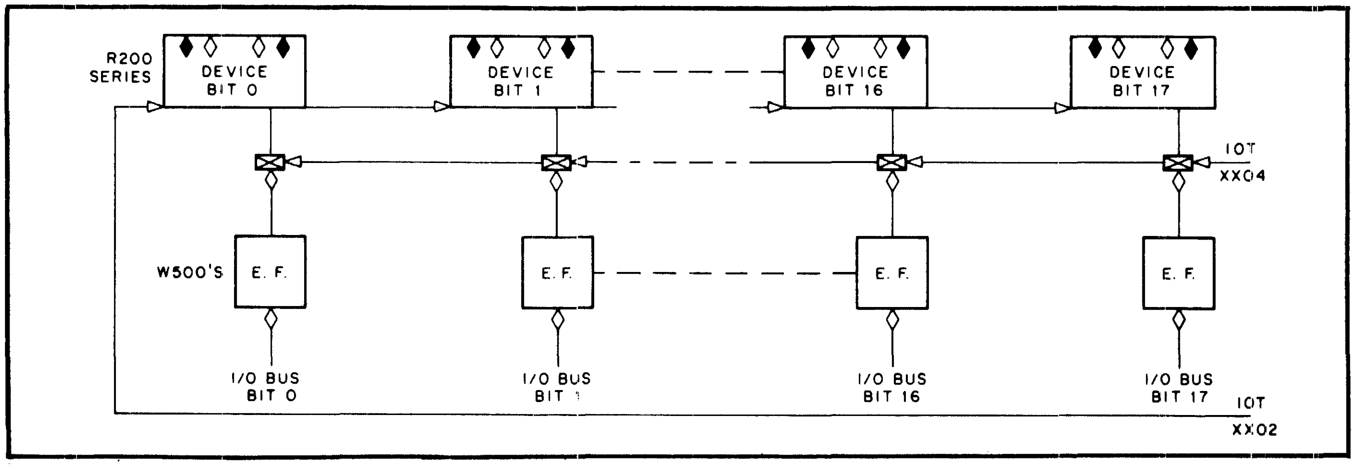


Figure 13-7 Loading a Device Buffer from AC

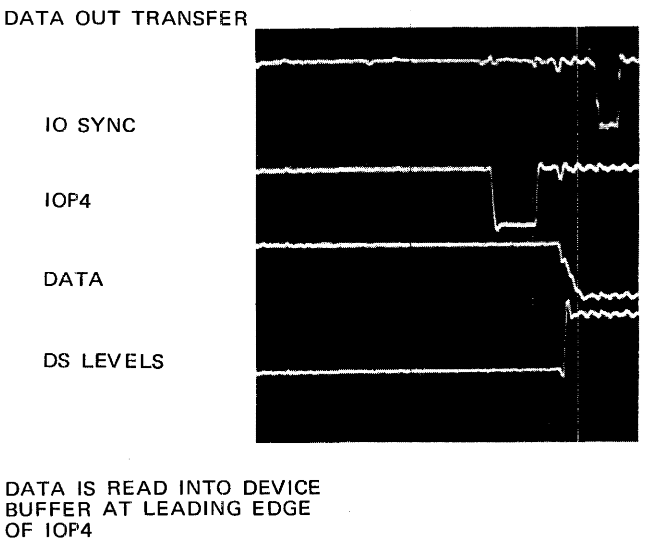


Figure 13-8 I/O Signals from I/O Bus to Buffer

3. IOT4 is generated in the W103 and applied to the DCD input gates of the device buffer. Note that the input DCD gates for each bit in each output device must be buffered from the I/O bus by Emitter Followers W500.

## I/O SKIP FACILITY

When IOT pulse 1 is used in a I/O skip instruction (example CLSF = IOT 0001 = 700001), it is gated with the device flag flip-flop through a diode gate and returned to the computer on the I/O skip request bus line. A positive pulse, 1 microsecond wide, is returned to the processor, if the tested flag is a binary 1. The skip flip-flop in the processor is set, and the instruction following the IOT instruction in the program sequence is not executed -- it is skipped. The 15 ma to -15v load for the skip bus line is located at its termination in the processor. The signal on the skip line is sampled 600 nsec after IOP1 is issued. An R111 without a clamped load or a S123 Diode Gate must be used in the device.

The sensing of a device flag (figure 13-9) is accomplished in the following manner:

1. IOT pulse 1 is generated in the device selector module.

2. This pulse is gated with the 1 output of the device flip-flop (i.e., flag) through a diode gate (unclamped R111 or S123) tied to the "I/O skip line".

3. If (and only if) the sensed flag is in the 1 state, a 1-msec pulse is returned to the central processor via this I/O skip line.

4. This pulse sets the I/O skip flip-flop in the processor. In this instance, the instruction following the IOT instruction is not executed.

## STATUS WORD FACILITY

The IOT instruction IORS (=IOT 0314 = 700314) loads the accumulator with a word comprised of various device flags and control flip-flops. External devices, having status bits assigned, use the read status level to gate the device flag onto the corresponding I/O bus bit line. (Bits 15, 16, and 17 have been reserved for special customer devices.)

The status word is readable at the console when the register function switch is in the STATUS position. Figure 9-1 illustrates the AC bit posi-

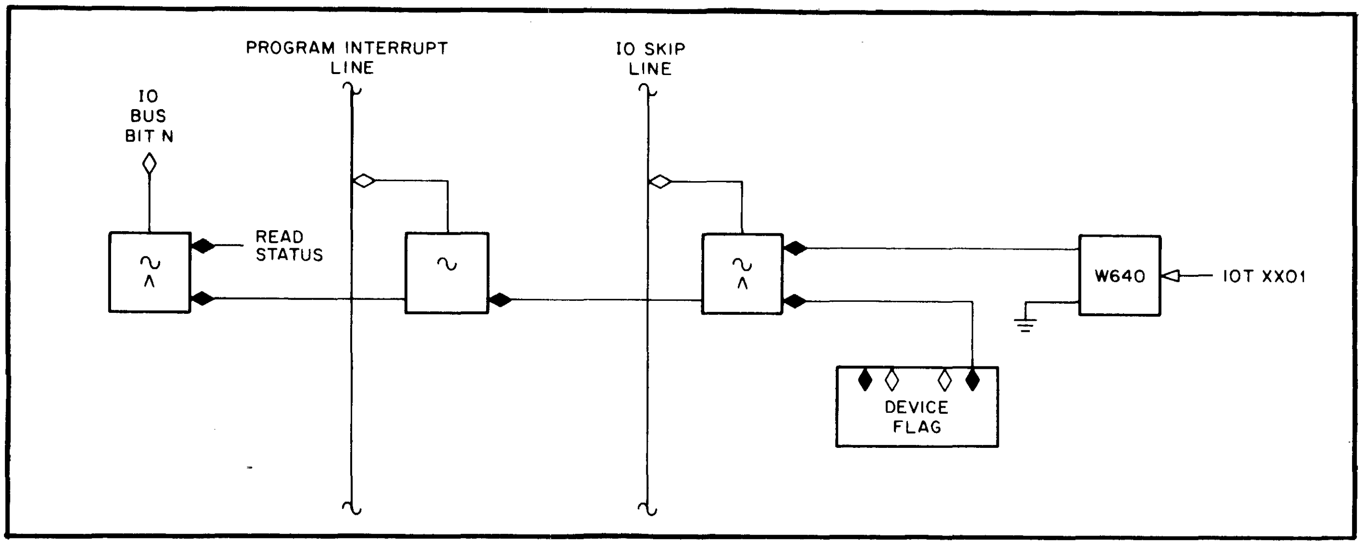


Figure 13-9 Device Flag Hardware

tions associated with the standard IORS assignments.

## PROGRAM INTERRUPT (PI) FACILITY

When computer time is at a premium, it is advantageous for the computer to perform other tasks rather than wait for some peripheral device to complete its operation. The program interrupt facility allows the program to ignore the peripheral device until it signals completion of operation. At that signal, the computer program can enter a subroutine to service the device. In case several devices are connected to the PI, a chain of I/O skip instructions and JMPs are issued by the program to determine which device caused the interrupt.

If the computer is servicing an API initiated interrupt, a PI request is not granted until all the API channels are dismissed.

Figure 13-9 shows the device hardware required for the program interrupt facility. The device flag is connected to the program interrupt request line by a Type R111 or S123 Diode Gate (unused input left floating). This request line is terminated in a 15-ma clamped load in the computer I/O frame.

Interrupts occur only when the program interrupt facility is enabled. IOT instructions associated with the PI are:

| *Mnemonic* | *Code* | *Operation* |
|---|---|---|
| IOF | 700002 | turn off PI |
| ION | 700042 | turn on PI |

When an interrupt is recognized, the computer stops execution of the main program. The 15-bit address of the next main program instruction and certain flags are stored in memory address 00000. The word is stored as shown on figure 13-10.

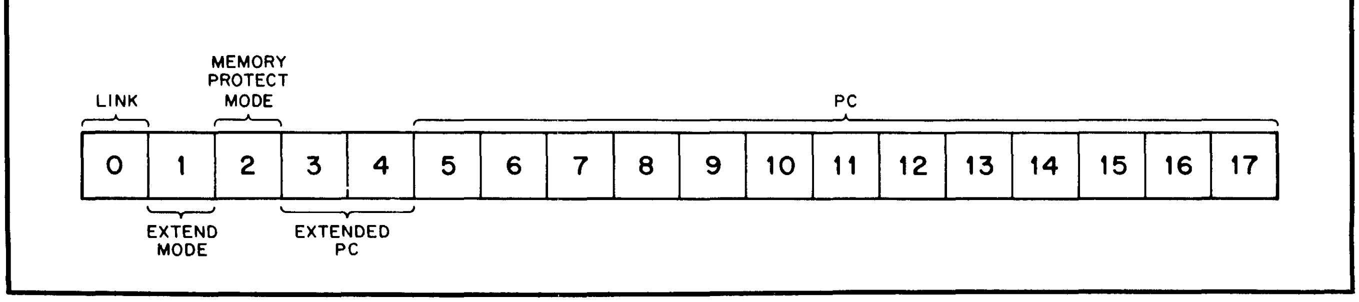


Figure 13-10 Program Interrupt Storage Word

## AUTOMATIC PRIORITY INTERRUPT, Type KF09A

In real-time computer systems such as the PDP-9/L a small number of peripheral devices with relatively few timing problems or priority considerations can be handled quite easily by the standard program interrupt facility without a significant loss in central processor efficiency. However, if the number of peripherals is large, with a hierarchy of priorities, or the desired data rate is over 50,000 words per second, signficant economies in central processor overhead time can be achieved by the use of a multi-channel automatic priority interrupt system.

The Type KF09A Automatic Priority Interrupt (API) option for the PDP-9/L provides the necessary hardware to handle a multiplicity of input/output devices with a mnimum of programming effort and a maximum of efficiency. Its priority interrupt structure permits high data rate devices to interrupt the service routines of slower devices with a minimum of system overhead. The API also permits the device service routines to be entered directly from hardware-generated entry points, eliminating the need for time-consuming flag searing to determine the identity of the device causing the interrupt.

The Type KF09A provides thirty-two channels, or unique entry points for device service routines, and eight levels of priority. The four highest priority levels are for device service requests, and interrupts on these levels are initiated by hardware service requests (i.e., device flags). The four lower priority levels are assigned to the processor for software routines; activities on these levels are always initiated by program requests, and four channels (or service routine entry points) are reserved for these priority levels.

Each device interfaced to the Automatic Priority Interrupt system must specify its "trap address" or unique entry point to its service routine. Locations $40_8$-$77_8$ are reserved as these service routine entry points, and "trap address" and channel numbers are related as follows:

$$(\text{TRAP ADDRESS})_8 = (\text{CHANNEL NUMBER})_8 + (40)_8$$

Locations $40_8$-$77_8$ should contain JMS or JMS I instructions to provide linkage to the actual service routines.

Table 13-2 shows the relationship between channel number and trap address, the channel assignments for standard PDP-9 I/O devices, and their suggested priority levels. All devices listed in Table 13-2 are connected to the API as shown, if the API option is purchased. The channel number-assignments should remain fixed for software compatibility, but priority levels may be changed at the user's option.

Devices are interfaced to the API in the same manner as they would be to the Program Interrupt (PI) facility for all signals except the device flag. For API devices, the device flag is tied to one of the API levels on the I/O bus through a W104 module in the device interface. (Device flags tied to an API level must AND their respective API RQ (1) level with the Device Flag to request PI.)

The W104 module is used to establish priority among devices tied to the same priority level; it also gates the trap address (channel number) onto the I/O Address (IO ADDR) lines at the appropriate time. Figure 13-11 shows the W104 module and its pin connections.

As Table 13-2 shows, there need not be any fixed relationship between channel number and priority level (except for the four software priority levels). The priority interrupt level of a device may be chosen by the user; it is determined by the set of lines to which the user interfaces.

The I/O bus contains twelve lines unique to the API; these include an API RQ (request), API GR (1) (grant), and API EN (enable) line for each of the four levels. Since these control lines are part of the standard I/O bus and pass through all devices, it is relatively easy to change a device's priority by disconnecting from one set of lines and attaching it to another. Because of the time delay encountered in propagating signals through a W104 module, no more than eight devices should be interfaced to one priority level.

Figure 13-12 shows an example of four devices tied to the API (only API-related lines are shown). In this example, the following relationship exists between device number and priority level.

| Device | Priority Level |
|---|---|
| A | 3 |
| B | 2 |
| C | 0 |
| D | 2 |

TABLE 13-2 CHANNEL AND PRIORITY ASSIGNMENTS

| Channel Number Octal | Trap Address | Standard Device | Suggested Priority Level | IO ADDR Bits 12-17 |
|---|---|---|---|---|
| 0 | 40 | Software channel 0 | 4 | 100 000 |
| 1 | 41 | Software channel 1 | 5 | 100 001 |
| 2 | 42 | Software channel 2 | 6 | 100 010 |
| 3 | 43 | Software channel 3 | 7 | 100 011 |
| 4 | 44 | DECtape (TC02) | 1 | 100 100 |
| 5 | 45 | Magtape (TC59) | 1 | 100 101 |
| 6 | 46 | Drum (RM09) | 1 | 100 110 |
| 7 | 47 | Disk | 1 | 100 111 |
| 10 | 50 | Papertape Reader | 2 | 101 000 |
| 11 | 51 | Clock Overflow | 3 | 101 001 |
| 12 | 52 | Power Fail (DP09) | 0 | 101 010 |
| 13 | 53 | Parity (MP09) | 0 | 101 011 |
| 14 | 54 | Light Pen (34H) | 2 | 101 100 |
| 15 | 55 | Card Readers (CR01E, CR02A) | 2 | 101 101 |
| 16 | 56 | Line Printer (647) | 2 | 101 110 |
| 17 | 57 | A/D (AF01) | 0 | 101 111 |
| 20 | 60 | DB99A/DB98A | 3 | 110 000 |
| 21 | 61 | Data Link to System 360 | 3 | 110 001 |
| 22 | 62 | Data Phone (DP09A) | 2 | 110 010 |
| 23 | 63 | Reserved for Systems Device | | 110 011 |
| 24 | 64 | Unassigned | | 110 100 |
| 25 | 65 | Unassigned | | 110 101 |
| 26 | 66 | Unassigned | | 110 110 |
| 27 | 67 | Unassigned | | 110 111 |
| 30 | 70 | Unassigned | | 111 000 |
| 31 | 71 | Unassigned | | 111 001 |
| 32 | 72 | Unassigned | | 111 010 |
| 33 | 73 | Unassigned | | 111 011 |
| 34 | 74 | Unassigned | | 111 100 |
| 35 | 75 | Unassigned | | 111 101 |
| 36 | 76 | Unassigned | | 111 110 |
| 37 | 77 | Unassigned | | 111 111 |

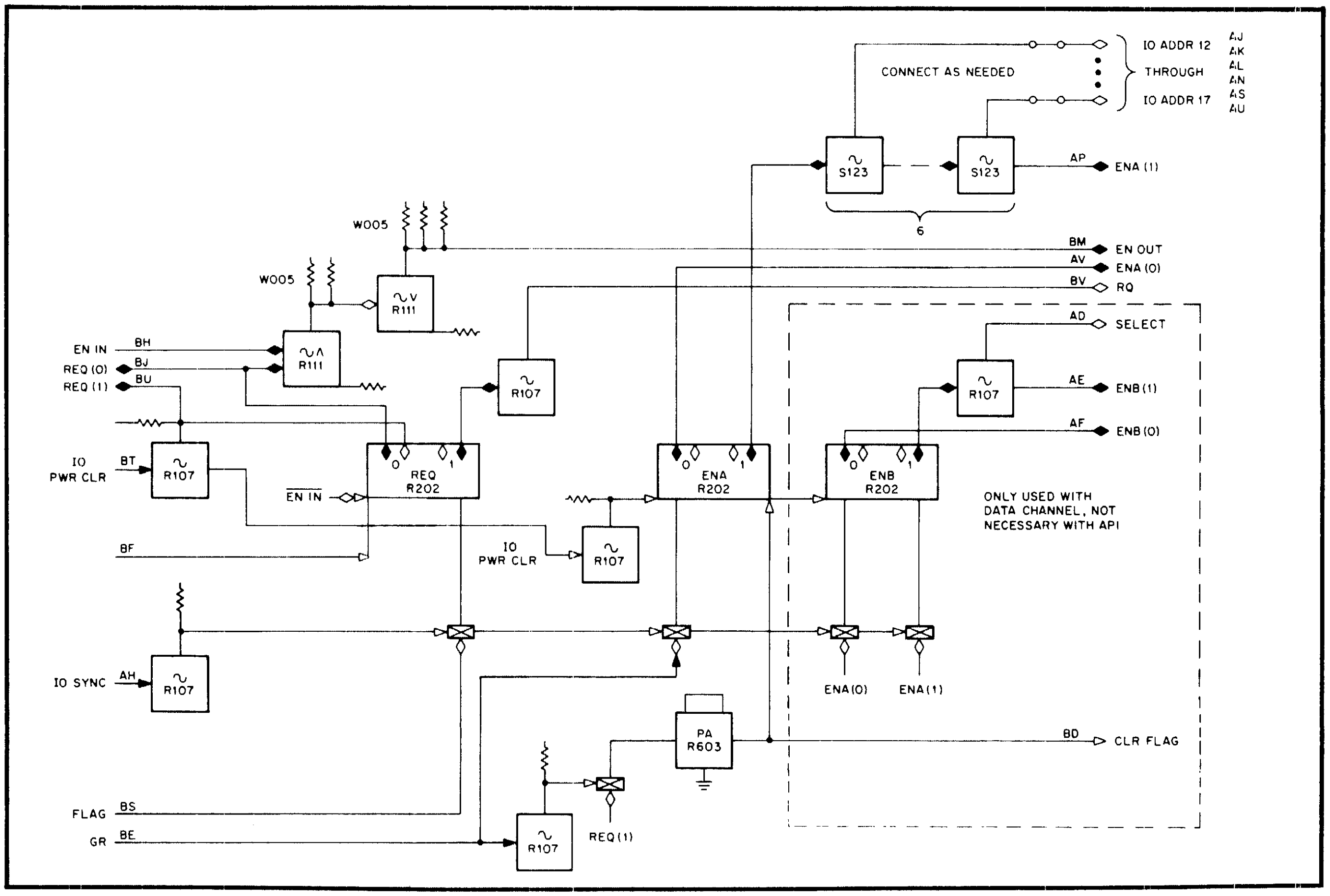


Figure 13-11 W104: PDP-9/L I/O Bus Multiplexer

If all four devices request service simultaneously, they are serviced in the following order: C, B, D, and A. Although B and D are on the same priority level, device B is serviced before D because it is closer to the computer on the I/O bus.

Each W104 module contains six address selection lines (pins AJ, AK, AL, AN, AS, AU). These lines are normally connected to the IO ADDR lines of the I/O bus to form the trap address. For standard API devices, pin AJ is connected to line as ($40_8$) and pins AK-AU from the channel number.

In some cases, trap addresses above $77_8$ may be used, although standard PDP-9 peripherals should be restricted to $40_8$-$77_8$. Figure 13-13 shows the possible connections for trap addresses between $100_8$ and $137_8$.

If a single device is required to generate a number of different addresses on the basis of a single flag, the W104 can be used to gate the address from a flip-flop register onto the IO ADDR lines. Figure 13-14 shows an example of this situation.

Figure 13-15 shows the proper connection of a device flag to the API system and also (optionally) the PI system.

Figure 13-16 shows the case of a single device with multiple flags, any one of which can cause a trap to a unique address. In this case, the different flags are all tied into the same request line. They may be tested individually by IOT instructions (I/O SKIP), so they should also be tied to the I/O SKIP line. They are also cleared individually, as shown. The flags are also connected to the ENA (1) line (as shown) to assure that the ENA flip-flop will be cleared when all of the flags are cleared, regardless of whether or not the API break is granted.

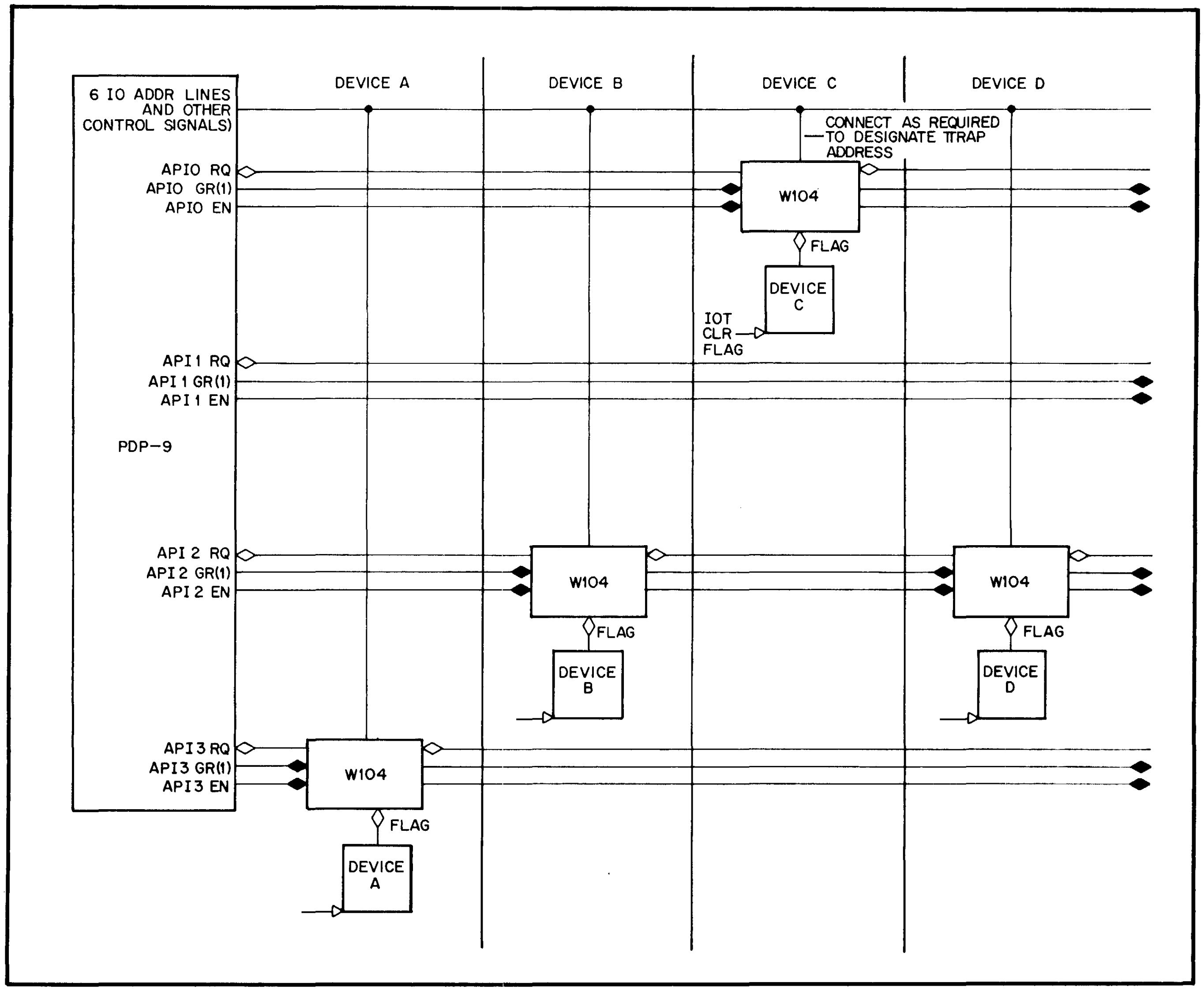


Figure 13-12 Devices on the Automatic Priority Interrupt

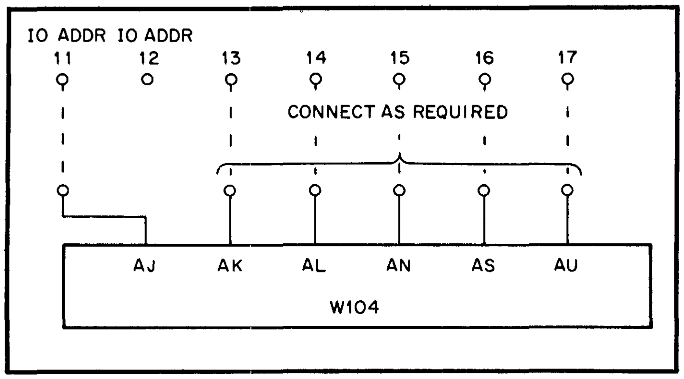


Figure 13-13 Connections for Trap Addresses Between $100_8$ and $137_8$

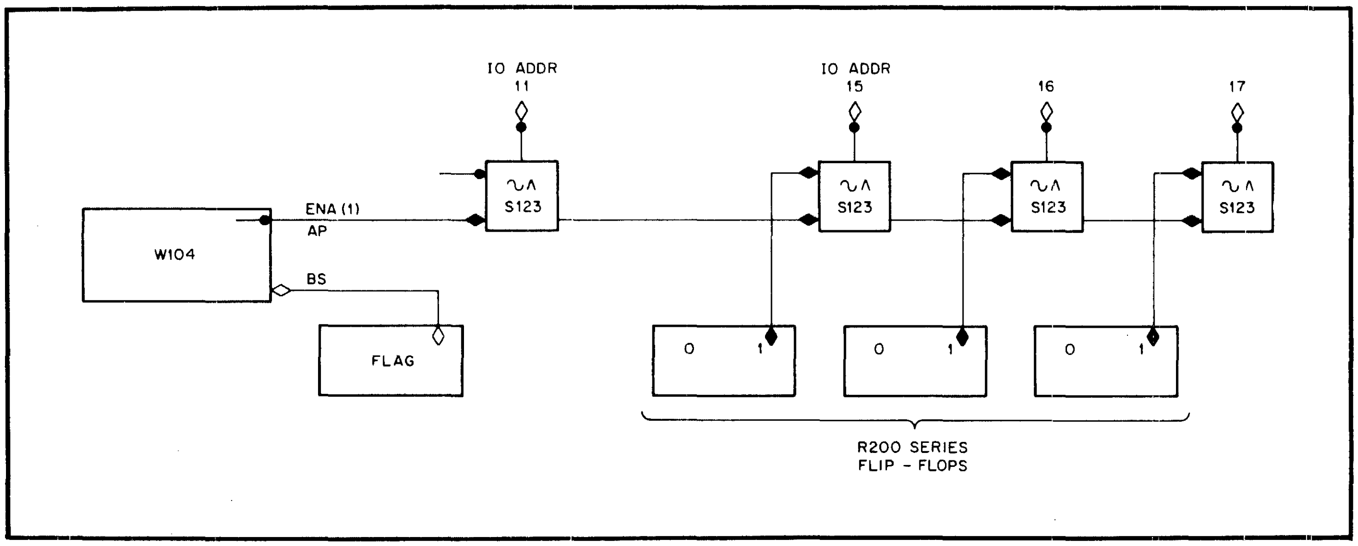


Figure 13-14 Gating Flip-Flop Register onto I/O Address Lines

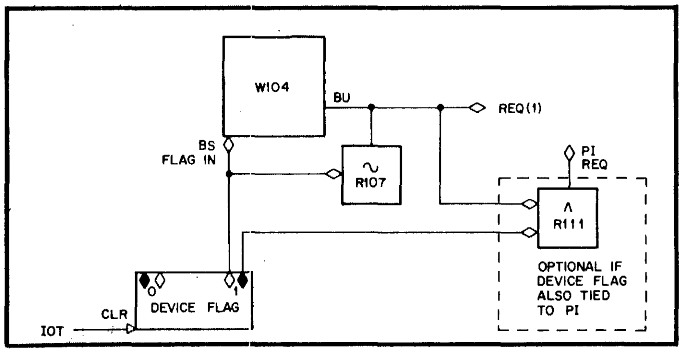


Figure 13-15 Interface of a Single Device Flag to both the PI and API

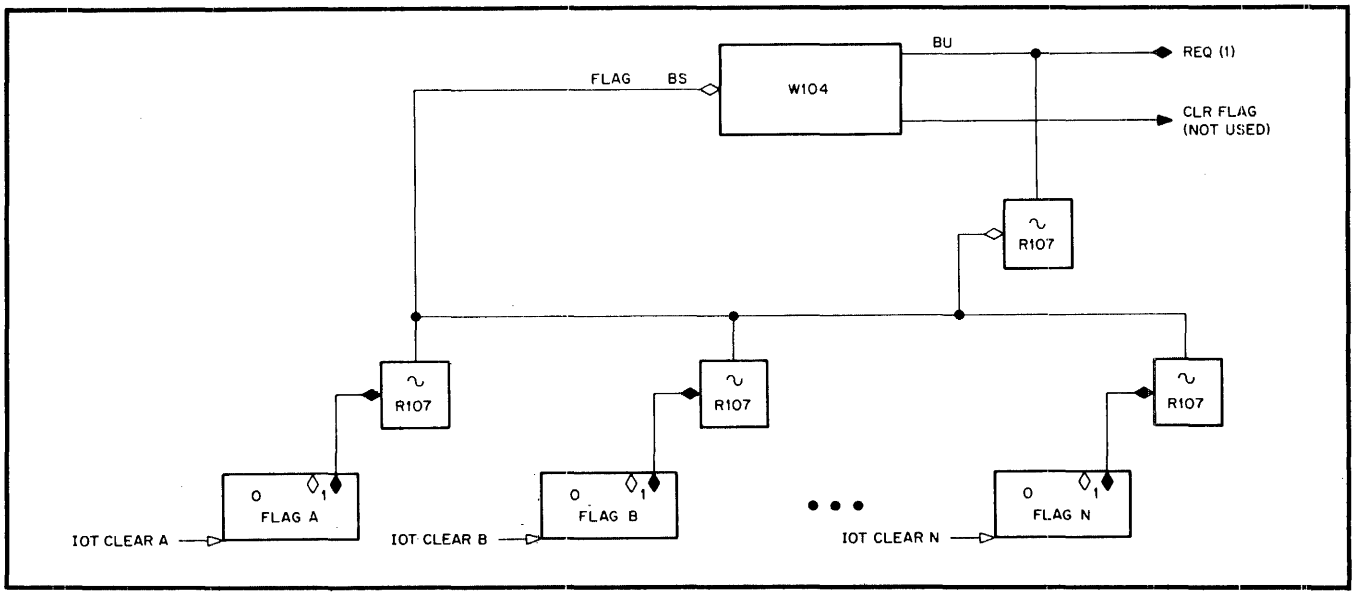


Figure 13-16 Single Device with Multiple Flags

# CHAPTER 14
# DATA CHANNEL

## GENERAL

The PDP-9/L data channel, multiplexed to permit interfaced service to eight peripheral devices, provides a relatively high-speed interface to the core memory along the I/O bus. Requests for data from I/O devices are honored by the channel at the completion of the instruction in progress at the time the grant signal is issued. The channel is controlled by word count and address registers held in core memory; each request updates these registers, and transfers the data between the memory and the device.

Each of the eight devices has a unique pair of (sequential) core memory registers associated with it. (The system software assumes that devices 0-3 use registers $30\text{-}37_8$.) These registers must be initialized by the program, before the peripheral device may begin transferring data through the channel. The first (word count) register, of lower numerical value, must be even, and is initialized to contain the 2s complement of the number of words to be transmitted. The second (address) register is initialized to contain 1 less than the first address of the data word block.

These registers may be examined at the end of channel operation to check for final address, if, for example, the device indicates that a short record was read. Peripheral devices normally issue a program interrupt (API) request at the completion of the transfer when the word count register has counted up to 0.

The maximum transfer capacity of the channel is between 160,000 and 220,000 words per second, depending on the mix of input and output rates. Each input transfer steals three processor cycles; each output transfer steals four processor cycles. The latency time (maximum wait before service is granted after a request is made) may be as high as 30 microseconds under adverse conditions (see latency section). Special care is necessary, however, when designing software for devices whose channel usage is greater than 50,000 words per second.

Priority among I/O devices making simultaneous requests is determined by their physical placement on the I/O bus, with devices closer to the processor having priority over devices further away. The establishment of priority requires that each device quickly propagate an enable signal to the next device on the bus, and a special module, the W104 Bus Multiplexer, has been designed for this purpose.

## LATENCY

Since data channel requests are only honored between instructions, the type of instruction in progress determines the waiting time until the interrupt is granted. The following considerations apply:

1. The IOT instruction is noninterruptible. The interrupt request is honored at the completion of the instruction which follows the IOT.

2. The EAE instructions delay interruption until they complete, which may be as long as 17 microseconds.

3. The XCT instruction is noninterruptible. The interrupt request is honored at the completion of the instruction referenced by the XCT.

4. Lower priority devices wait for the completion of data transfers on the requesting higher priority channel. Hence, if four requests come up simultaneously, the lowest may wait 12 microseconds, and indefinitely if a higher priority device is taking successive breaks.

Long XCT chains on sequential IOT instructions can lock out channel requests for indeterminate periods of times. These are to be avoided in programs operating devices requiring short latency. EAE instructions requiring more than 12 microseconds are uncommon, but possible. Unfortunately, requests tend to stack up during these waiting periods, so that lower priority devices must wait even longer. I/O system design must insure that the latency time requirement of each peripheral is satisfied.

## DEVICE INTERFACE HARDWARE

Each device connected to the data channel must have the interface hardware outlined below. The first four requirements are essentially the same as those met by devices connected to the program interrupt. They insure that the device hardware may also be checked by maintenance routines using special IOT instructions. Requirements 5, 6, and 7 are met by the Type W104 Bus Multiplexer module, strongly recommended for use in the interface. Basic connections are shown on figure 14-1. The W104 is shown on figure 14-2.

1. Each device must have the ability to decode the 6-bit selection code transmitted by the processor on the device selection lines. When selected, the device must be capable of producing internal command pulses in response to IOP pulses transmitted on the bus. The module performing such functions in the peripheral device is called the device selector. Furthermore, the device must have the ability to force selection of the device selector, regardless of the address on the selection lines. The Type W103 Device Selector module possesses this property.

2. Each device receiving output data from the computer must contain gating circuits at the input of the receiving register capable of strobing the I/O bus information into the register when triggered by a command pulse from the device selector. In addition, the device must supply a write request level to the channel during the period wherein it is selected.

3. Each device supplying input data to the computer must include gating circuits at the output of the transmitting register capable of gating this register onto the I/O bus, when triggered by a command pulse from the device selector. In addition, the device must supply a read request level to the channel while it is selected.

4. Each device must contain a request flag (flip-flop), which is set whenever the device is ready to receive (or transmit) another word of information. This flag is normally cleared when the transfer is complete.

5. The device data flag is used to request a break through a synchronizing flip-flop, which drives the request line (DCH RQ). The device data flag must be cleared when the break is granted (DCH GR).

6. Each device must be capable of propagating the enable signal (DCH EN) to the next device to establish priority of devices along the bus in case of simultaneous requests. The next device must be enabled if the current device is enabled, and is not itself requesting.

7. Each device must contain the gating circuits necessary to transmit the core memory address of the word count register assigned to the device. The address is transmitted by the selected device upon receipt of the grant signal (DCH GR) from the channel.

8. Each device must contain an "active" flip-flop which controls whether or not the device periodically requests data transfers through the channel. This flip-flop is normally turned on by the program with an IOT instruction, and off by the IO OVFLO signal transmitted to devices by the channel.

## INITIAL SEQUENCE OF DATA-IN TRANSFER (TO COMPUTER)

The device flag is raised asynchronously by some state change in the I/O device control or associated mechanical hardware. This flag is synchronized by the W104 Multiplexer, which requests a data channel interrupt through the DCH RQ line. If more than one device on the channel is requesting, the multiplexer insures that the lower priority device is shut out by driving its enable (DCH EN) input line to ground (disabled state). This request is recognized by the processor and, at the end of the current instruction, control is relinquished to the channel hardware.

The channel hardware begins operation by identifying the device requesting service. This is performed by issuing a grant signal (DCH GR) to all connected devices. Upon receipt of the grant signal, the device which supplied the DCH RQ transmits the core memory address of its word count register along the I/O address lines. The specified register is read from memory, incremented, and rewritten. If, in this word count updating procedure, the count reaches 0, an I/O overflow signal is sent to all devices. The device hardware interprets the overflow signal as a shut down command. No further trans-

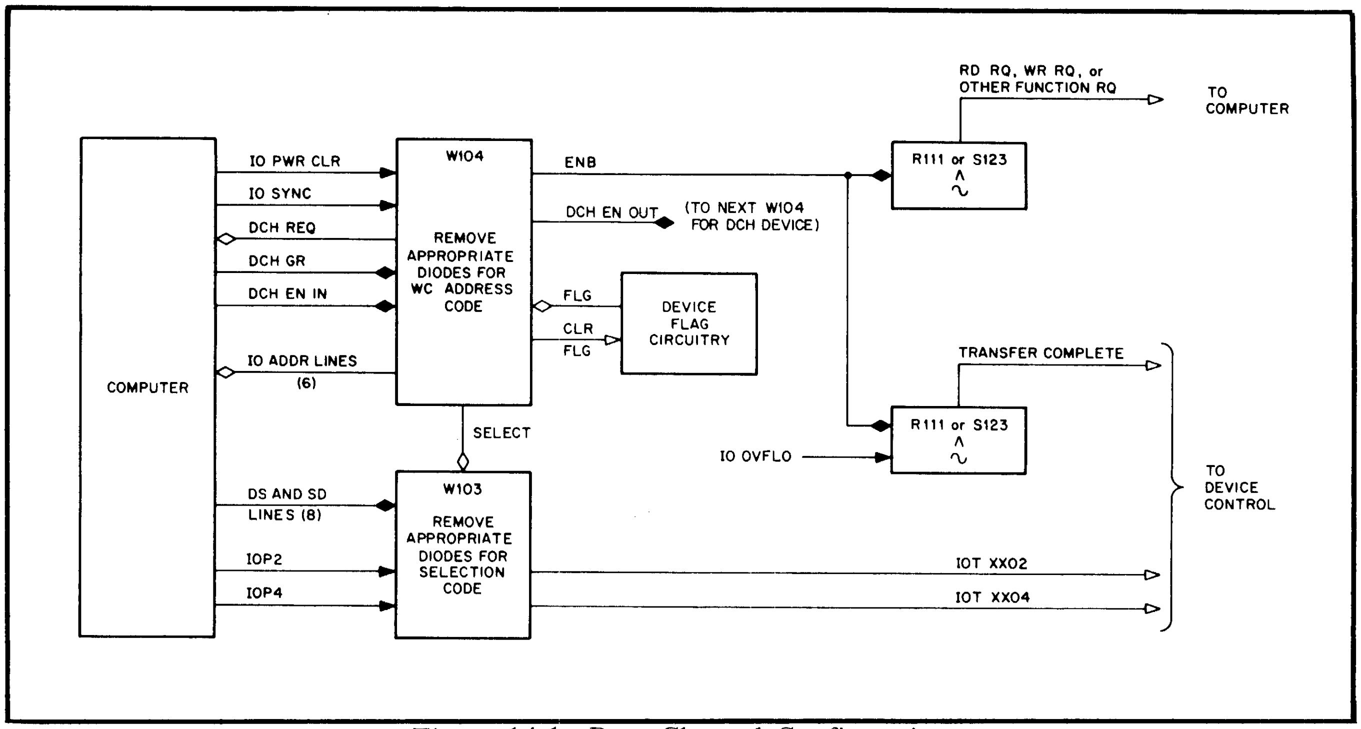


Figure 14-1 Data Channel Configuration

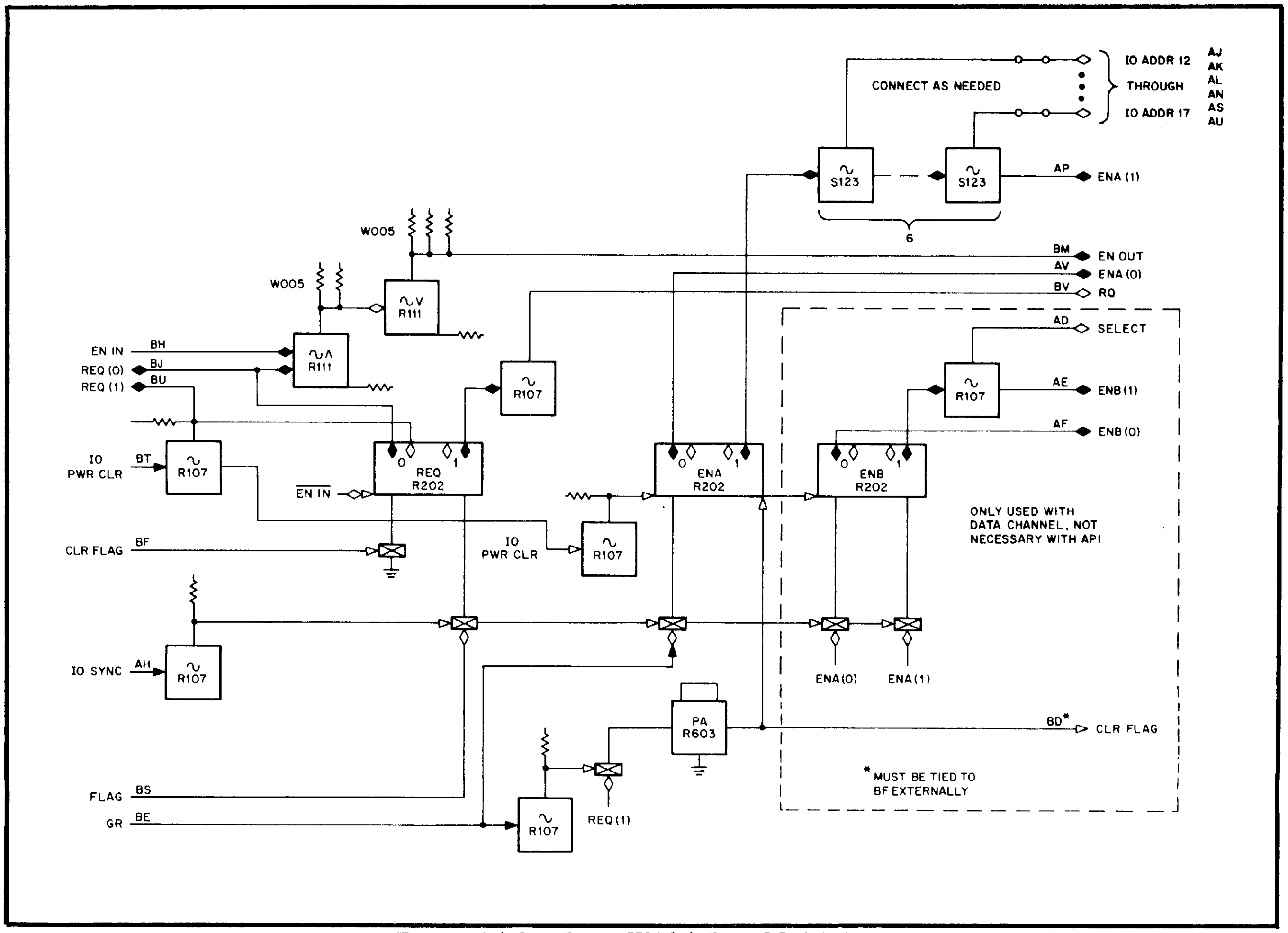


Figure 14-2 Type W104 Bus Multiplexer

fers are made until the device is re-initialized by the programmer.

After incrementing the word count register, the channel reads the next sequential word from memory. This is taken as the current address register, which is incremented and rewritten into memory. The update value is used to specify the location into (from) which the data is to be transferred.

### Operations Unique to Reading (Refer to figure 14-3)

If the read request (RD RQ) signal is present, and the write request (WR RQ) signal is not present, the transfer is taken as an into-computer data transfer. At the beginning of the second (current address) cycle, IOP2 is issued. At this time the device is expected to gate its data onto the I/O bus for subsequent readin by the channel. At the end of the second cycle, the data is read into the AR register. The third cycle stores the data word in the memory. This ends the sequence, and the channel relinquishes control to the processor.

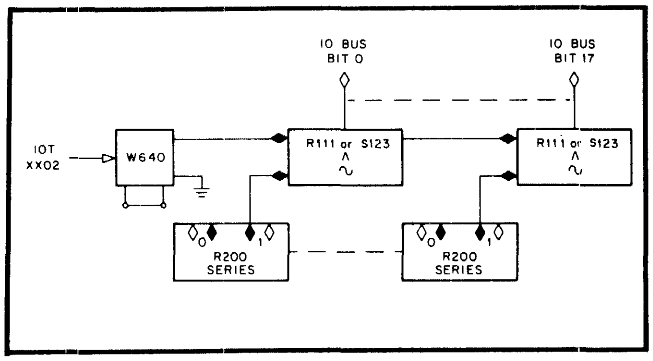


Figure 14-3 DCH In Transfer (To Computer)

## INITIAL SEQUENCE OF DATA–OUT TRANSFER (FROM COMPUTER)

The initial sequence of the data-out transfer is the same as the previously described data-in transfer.

### Operations Unique to Writing (Refer to figure 14-4)

If the write request signal is present, and the read request signal is not present, the transfer is taken as an out-of-computer data transfer. During the middle of the third (data) cycle, the channel places the requested data onto the I/O bus for subsequent readin by the device. The data remains available until the middle of the fourth cycle. At the beginning of the fourth cycle, IOP4 is issued instructing the device to clear its buffer, and to gate the data on the bus into its receiving register. DCD gates must be used. The sequence then ends, and the channel relinquishes control to the processor.

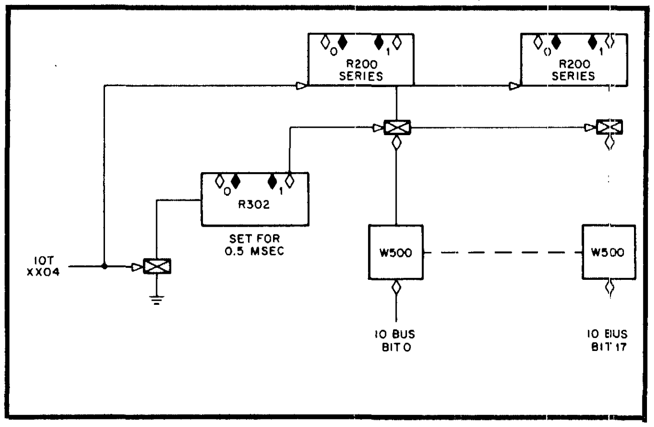


Figure 14-4 DCH Out Transfer (From Computer)

## EXPANSION TO EIGHT DEVICES

The number of devices connected to the channel is limited only by the maximum combined transfer rate capability and the propagation delay per device of the enable signal (DCH EN). Approximately 600 nsec are available between I/O SYNC and DCH GR, wherein the enable signal may propagate through the four multiplexers. This is always possible with maximum permissible I/O bus cable length and use of the Type W104 Bus Multiplexer module in each device. If cable lengths are kept to a minimum, it may be possible to attach more devices to the multiplexer. The limit with extremely short cabling and use of the W104 Bus Multiplexer is eight devices.

## SIGNAL DEFINITIONS

Table 14-1 lists I/O signals and their respective definitions. See chapter 12 for details of I/O connector and figure 14-5 for DCH timing.

## ADD-TO-MEMORY CAPABILITIES

With the add-to-memory capability, certain facilities are available. These include add-to-memory instructions, specified by sending both RD RQ and WR RQ, and memory increment, specified by sending a signal on the NEM INC line.

The add-to-memory operation is a combination of reading and writing. The data transmitted by the device is added to the word read from memory and re-written into memory. The sum is transmitted back along the I/O bus. Four cycles are required. Since the sum is re-trans-

TABLE 14-1 SIGNAL DEFINITIONS

| Signal | Definition |
|---|---|
| DCH RQ | The request signal from the device to the channel indicating that the device requires service. The line must be driven by a saturated transistor whose emitter is grounded. A 15-ma load to -15v terminates the line at the processor. |
| DCH EN | A bus driven signal which establishes priority along the device. Each device must supply a noninverting bus driver with a total transition time of 100 nsec into 5 feet of bus cable. |
| DCH GR | A bus driven signal emanating in the processor which instructs the selected device to transmit its address back to the processor. Maximum load is eight Type W104 Bus Multiplexer modules or equivalent. |
| RD RQ | The request signal from the device to the channel indicating that the device wishes the transfer to be in the "in" direction. The line must be driven by a saturated transistor whose emitter is grounded (R111 or S123). A 15-ma load to -15v terminates the line at the processor. |
| WR RQ | As RD RQ except specifies "out" direction. |
| I/O ADDR | Fifteen lines, of which only the least significant six are normally used. The device transmits the address of the core memory register specifying word count along these lines. Loading is the same as RD RQ. |
| I/O DATA | Eighteen lines of bidirectional data transfer between the channel and the device. Loading is the same as RD RQ. Receiving registers in output devices must buffer the incoming lines with W500 Emitter Follower modules. |
| +1 CA INH | A special signal line used by devices which automatically search. The presence of the +1 CA INH signal inhibits the normal incrementing of the current address word driving the second cycle. Load is the same as RD RQ. |
| MEM INC | Load is the same as RD RQ. |
| I/O OVFLO | A pulse originating in the channel logic which indicates the transfer of the specified number of data words has been completed. It is transmitted during the first cycle if the word count register increments to zero. |

mitted to the device at IOP4 time, the device may detect overflow.

The memory increment operation is the first cycle only of a channel transfer. The word specified on the I/O address lines is incremented. An overflow signal (IO OVFLO) is transmitted to the device if the data word is incremented to 0. Maximum rate is 160,000 increments per second, but this may be unattainable if long instructions are encountered.

## STANDARD CORE REGISTER ASSIGNMENT

Standard core register assignments of the DECtape; Magtape and Interprocessor Buffers are listed in table 14-2.

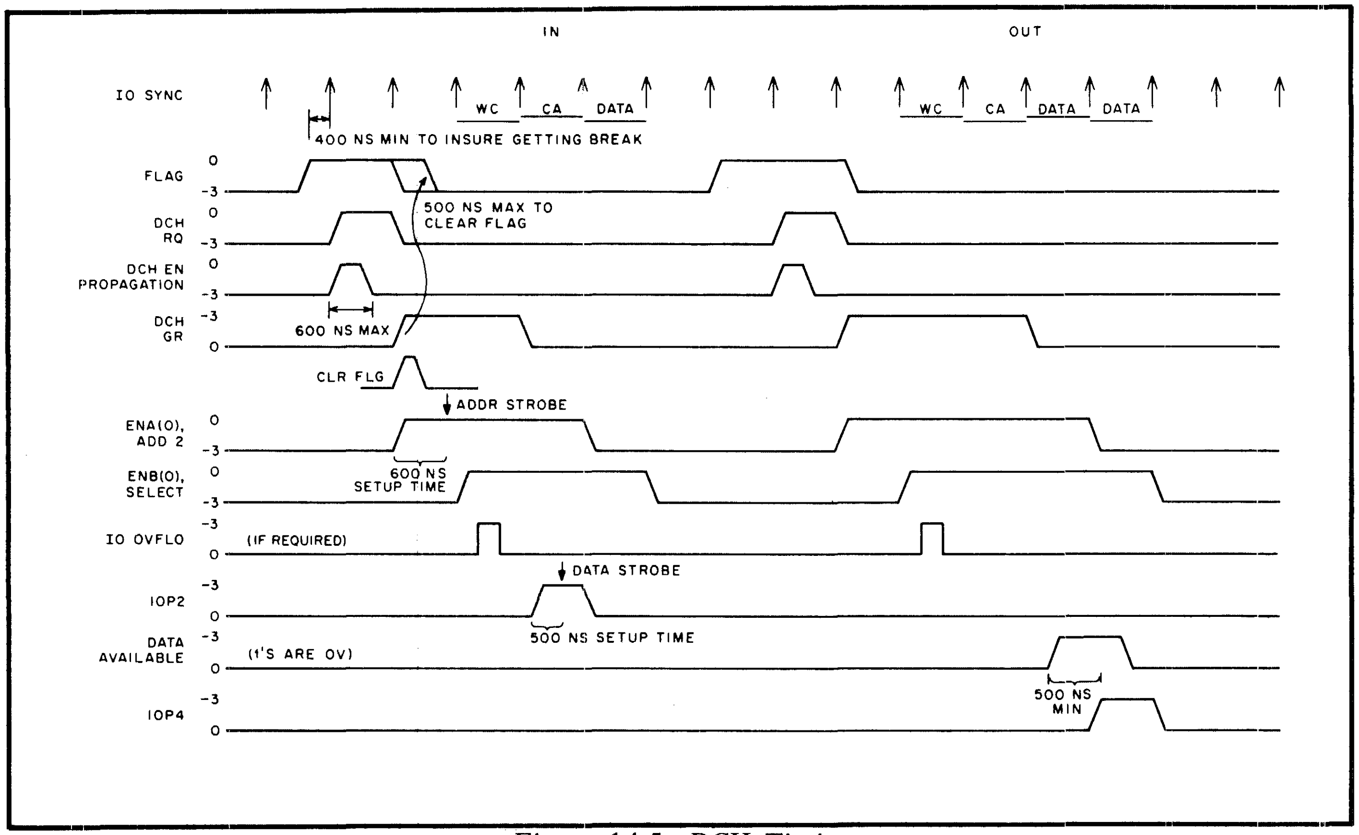


Figure 14-5 DCH Timing

TABLE 14-2 STANDARD CORE REGISTER ASSIGNMENT

| Device | Word Count | Initial Address |
|---|---|---|
| DECtape | 30 | 31 |
| Magtape | 32 | 33 |
| Interprocessor Buffers | 34 | 35 |
| Not presently assigned | 36 | 37 |

# CHAPTER 15
# INSTALLATION PLANNING

## GENERAL

This chapter describes the physical dimensions of a basic PDP-9/L and expander cabinets. Power requirements, heat produced, and the physical sizes of options are detailed in tables 15-1 through 15-3. Several typical configurations are shown.

## PHYSICAL CONFIGURATION

The basic PDP-9/L is housed in a single cabinet, 32-½ in. wide (with end panels), 27-¾ in. deep, and 69-½ in. high. The operator's table projects forward 22 in. and a rear clearance of 31 in. is needed for access to the logic. Physical dimensions of the basic PDP-9/L are shown in figure 15-1.

The PDP-9/L is painted black, with grey end panels, and has a red accent panel on the front. The rear door also contains a red accent stripe. A black chair is supplied with the PDP-9/L.

In the basic PDP-9/L cabinet, the paper tape reader/punch and operator's console are mounted on the front, while most of the logic is mounted on the rear door. In the center of the front is an operator's console, with the standard paper tape reader/punch unit mounted above and a table mounted below. The power supplies are underneath the table. Behind the red accent door, to the left of the paper tape reader/punch, is the marginal check panel, the maintenance panel, and the console lock switch. Above the paper tape reader/punch unit, space is reserved for the Type ME09B Option Panel which contains the wiring for the Type KG09A Memory Extension Control, and the Type KX09A Memory Protection Option. Figure 15-2(A) shows the front of the PDP-9/L cabinet.

Most of the PDP-9/L logic is contained on the rear door (see figure 15-2(B)). The logic is contained on three wire-wrapped frames, self-contained, with fans, fuses, and marginal check switches. The top frame contains a 16,384 word memory wing, the center frame is the central processor, while the bottom frame contains the I/O logic. These three frames are bolted together to form a solid door, and connections are made from frame to frame with ribbon cables. Hold-down bars are available as an option.

Cooling is accomplished by the fans built into the frames. Air is sucked in through vents in the rear door, blown past the logic by the fans, and exits through vents in the top of the cabinet.

Several options are wired into the frames of the basic PDP-9/L for easy implementation. The central processor frame contains the logic for the Extended Arithmetic Element, Type KE09A. The I/O frame has the Automatic Priority Interrupt, Type KF09A, the Power Failure Protection Option, Type KP09A, and the Oscilloscope Display Control, Type 34H, already wired in. Space is reserved in the basic cabinet for the Type ME09B (KG09A, KX09A) option to keep it as close as possible to the central processor logic.

Memory above 8192 words is expanded by adding the Memory Extension Control, Type KG09A (housed in the Type ME09B option above the paper tape reader/punch), and extra memory logic in the memory wing. The second 16,384 words of memory are housed in a second 32-inch cabinet to the left (facing the system) of the basic PDP-9/L. This is shown in figure 15-3. The cabinet contains room for the second 16,384 words of memory *on the front door,* while power supplies are mounted in the rear. The rear door of this additional 32-inch cabinet also contains a red stripe, but the front door is solid black.

Other options are added to the PDP-9/L by attaching standard 19-inch cabinets to the right of the basic cabinet (facing it). The dimensions of these cabinets are shown in figure 15-4. Connections to these options are made from the PDP-9/L I/O logic via I/O Bus Cables, Type BC09A. Figure 15-5 shows a one-bay expansion of a PDP-9/L with the addition of a DECtape Control, Type TC02, and four (4) DECtape Transports, Type TU55.

Several options, including the Type 30D, and the Type TU20 and TU20A tape transports, are composed, in part, of free-standing units. These are housed in one or more 19-in. cabinets and are interfaced to the PDP-9/L via 25-ft. cables.

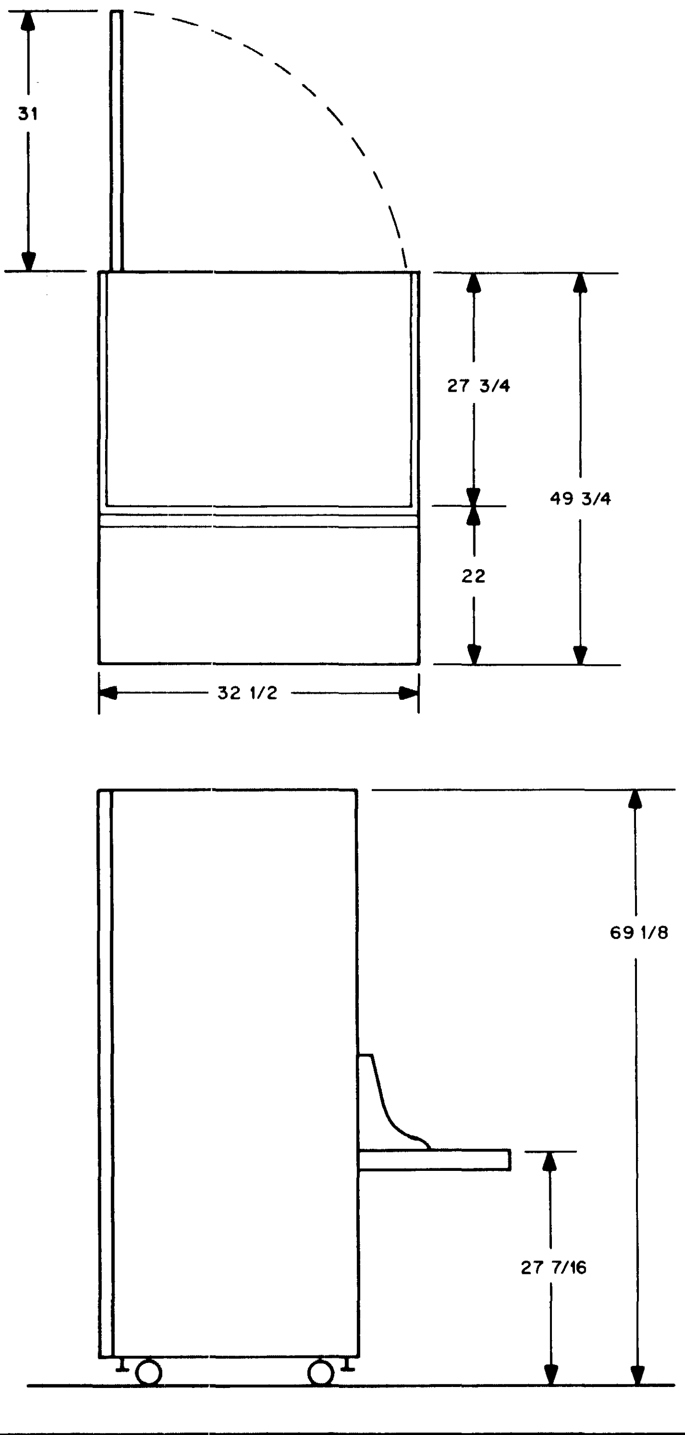


Figure 15-1. Basic PDP-9/L Cabinet Specifications

## PLACEMENT OF OPTIONS

PDP-9/L systems are assembled according to the following guide-lines*:

*Customers may request deviations from these procedures at extra cost.

1. Cabinets are numbered 0 through n (or 1 through n, if no extra memory is required), with the numbers always running from left to right (figure 15-6), and the central processor cabinet is always designated no. 1. Bays 0 and 1 are 31-in. cabinets, and 2 through n are 19-in. cabinets. All cabinets except no. 1 have

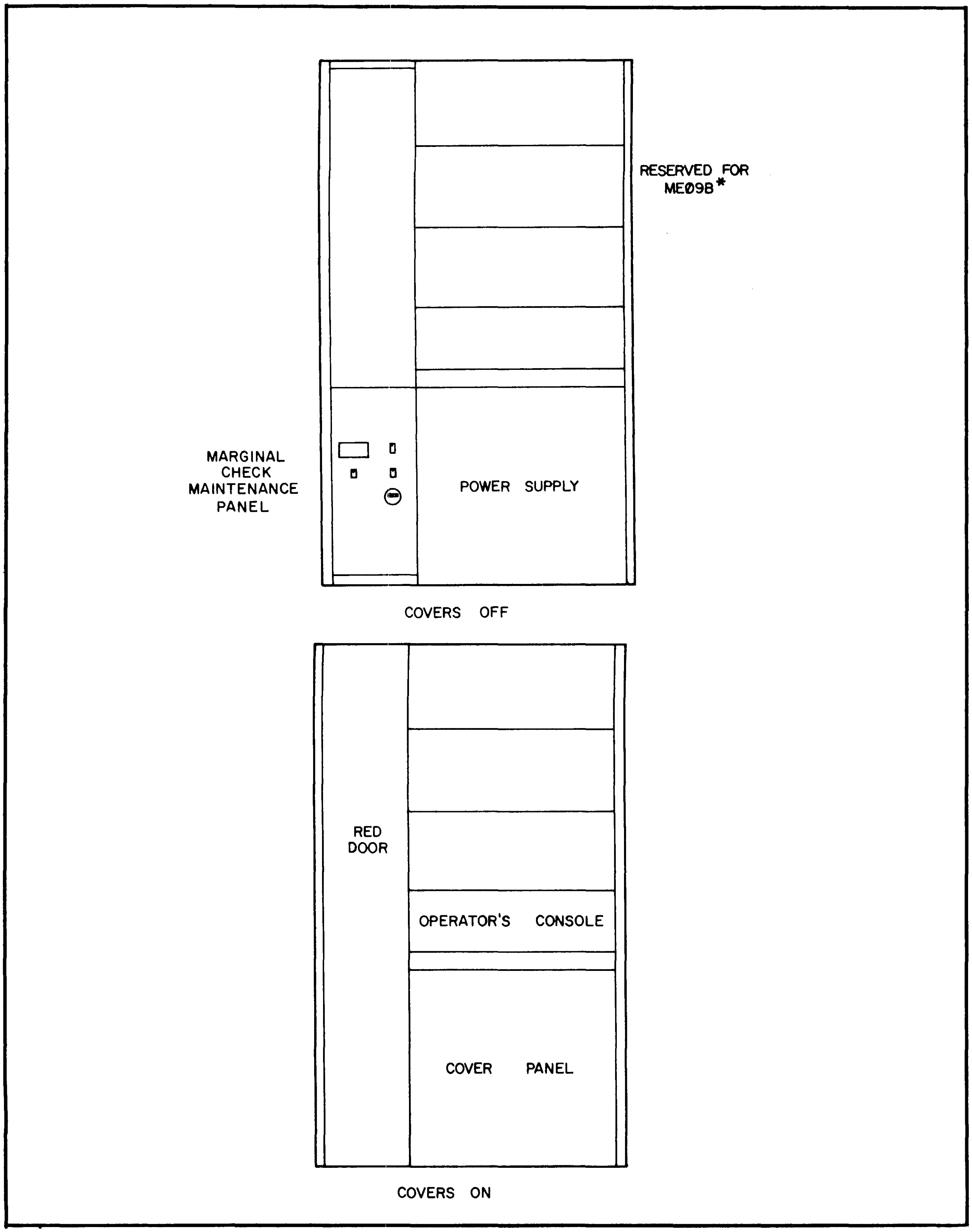


Figure 15-2(A). Basic PDP-9/L (Front)

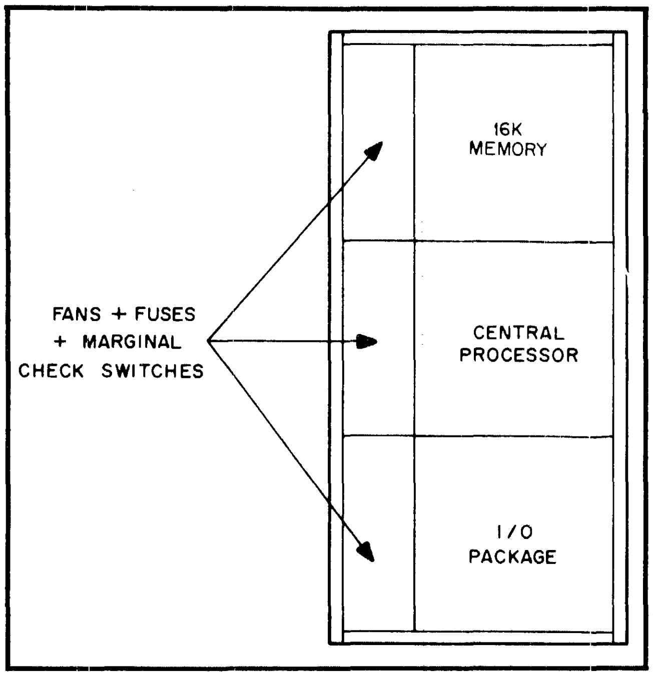


Figure 15-2 (B). Basic PDP-9/L (Rear, Back Door Removed)

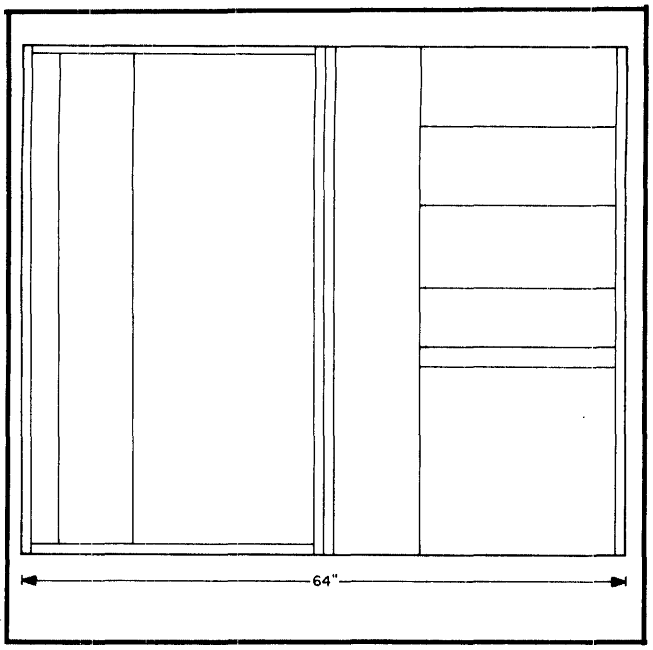


Figure 15-3. PDP-9/L With Extra Memories (Front)

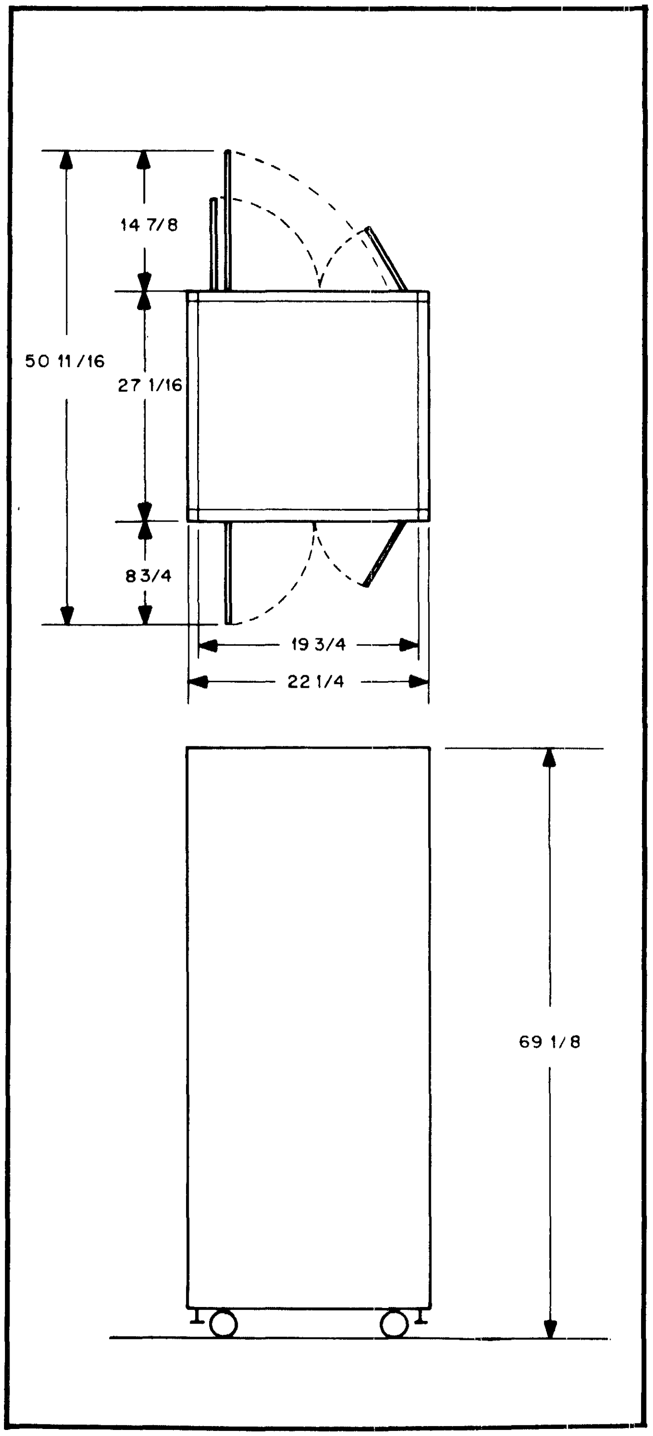


Figure 15-4. 19-Inch Cabinet Specifications

their logic mounted on the front. The 19-in. cabinets can hold eleven standard 5½-in. mounting panels of logic; these are lettered A through K. The bottom panel space cannot be utilized.

2. The options listed below are wired in and require no additional space.

a. Extended Arithmetic Element, Type KE09A

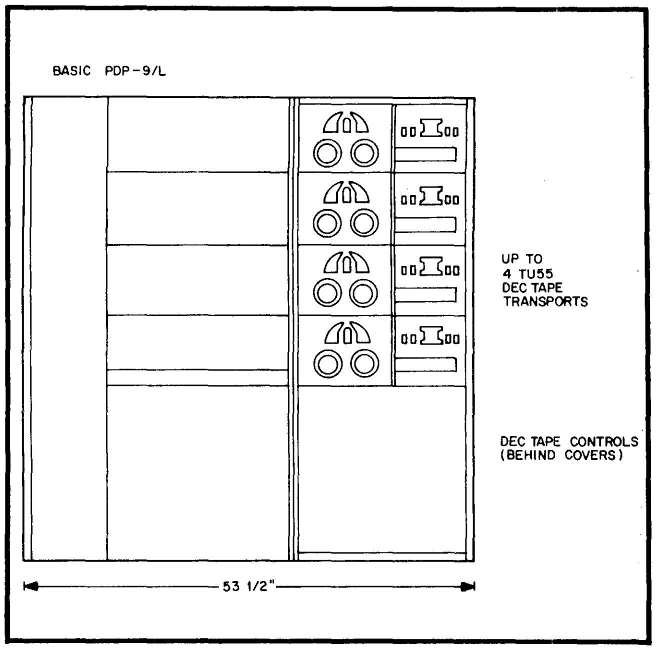

BASIC PDP-9/L
UP TO 4 TU55 DEC TAPE TRANSPORTS
DEC TAPE CONTROLS (BEHIND COVERS)
53 1/2"

Figure 15-5 PDP-9/L with DECtape and PC09A

b. Automatic Priority Interrupt, Type KF09A

c. Power Failure Detection Option Type KP09A

d. Oscilloscope Display Control Type 34H

3. Several options requiring additional logic will also be housed in bay no. 1, in 19-in. mounting panels, above the paper tape reader/punch unit. The Memory Extension Control, Type KG09A, and the Memory Protection Option, Type KX09A are wired into one pair of panels (Type ME09B) which fit in the two spaces immediately above the reader/punch unit.

4. All memory in excess of 16,384 words is housed in bay no. 0. *No other options* are permitted in this cabinet.

5. All cabinet-mounted input/output options and interfaces to free-standing options are housed in bays no. 2 through n, with the following priorities governing proximity to the central processor.

a. DECtape

b. All other standard DEC options

c. Special systems built by DEC

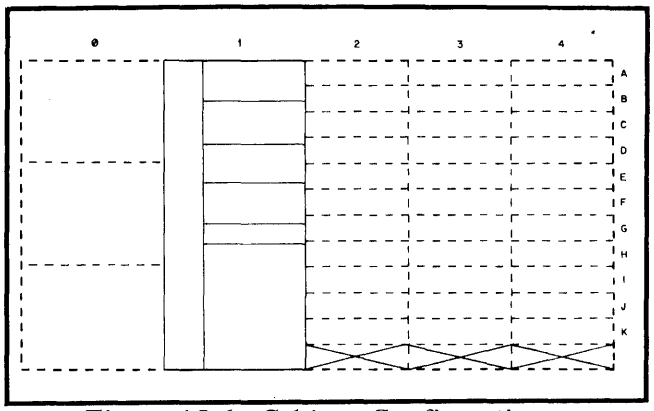


Figure 15-6 Cabinet Configurations

d. Special systems (or the cabinets for them) built by customers.

6. DECtape, when included in the system will occupy bay no. 2 and, if necessary, bay no. 3. Even if the DECtape does not completely fill bay no. 2, *no other options* will be permitted in the bay with it. This allows room for expansion of the DECtape system. If bay no. 3 is also used for DECtape, the bottom three panels (I through K) are available for other options (figures 15-7 and 15-8).

7. Oscilloscope displays are mounted at the *top* of the cabinet closest to the central processor after DECtape (see figure 15-9).

8. Analog-to-digital converters and output relay buffers are mounted as near to the top of the cabinets as possible.

9. Interfaces to large-screen CRTs, card readers, line printers, and interprocessor buffers will be mounted as low as possible to shorten the length of external cables going out of the bottom of the cabinet.

10. Magnetic Tape Controls, Type TC59, will be mounted at the *bottom* of the cabinet and as far to the right as possible (figure 15-9).

## ENVIRONMENTAL REQUIREMENTS

PDP-9/L systems operate satisfactorily under ordinary conditions of humidity, shock, and vibration. They are tested in the plant between 50 degrees and 122 degrees F. The best operating temperatures, however, are between 70 degrees and 85 degrees F and a humidity between 30 and 80% are recommended. If room air conditioning is planned, consult tables 15-1 through 15-3 for heat outputs of the system components.

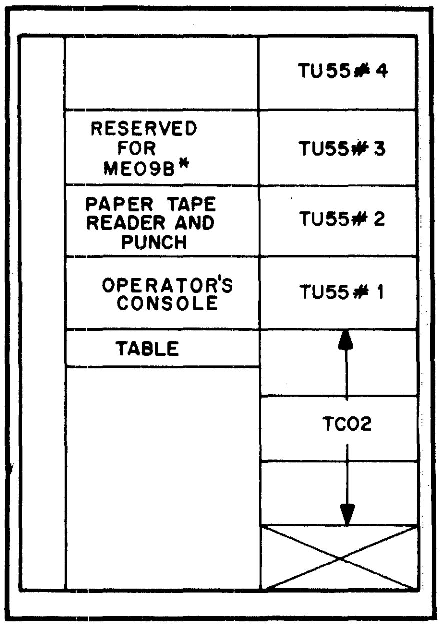


Figure 15-7 Basic PDP-9/L with DECtape and PC09A

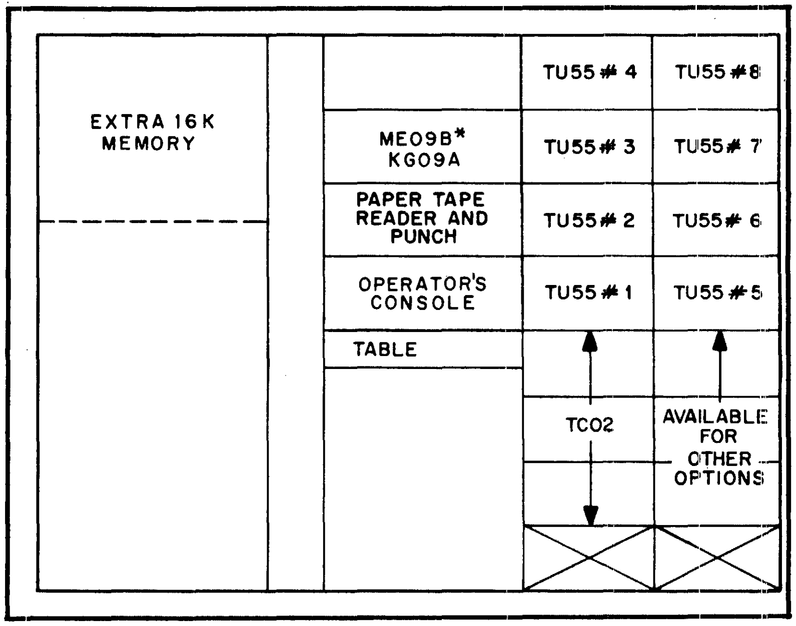


Figure 15-8 Memory Expansion and DECtape Expansion

## POWER REQUIREMENTS

The PDP-9/L requires a source of 115-v, 60-Hz, single-phase power. Upon order, all equipment can be factory-wired for 50 Hz at 115, 230, or 250 v. The power source must maintain the nominal voltage to ± 10% and the frequency to + 0.5 Hz. The electrical characteristics of individual components are given in tables 15-1 through 15-3.

## CABLING REQUIREMENTS

Most PDP-9/L systems will require 115v, 30 amp, Hubbel Twistlock flush receptacles (or their equivalent) to mate with equipment power cables. Exceptionally large PDP-9/L systems may require 50 amp sources. If in doubt, consult tables 15-1 through 15-3 or your DEC representative.

All free-standing cabinets, magnetic tape transports, card readers, line printers, etc.) require independent 115v (or equivalent) receptacles. Power on/off, however, will be under control of the PDP-9/L console switch.

Cables are connected to cabinets through drop panels in the cabinet bottoms. Sub-flooring is not necessary because the cabinets are elevated

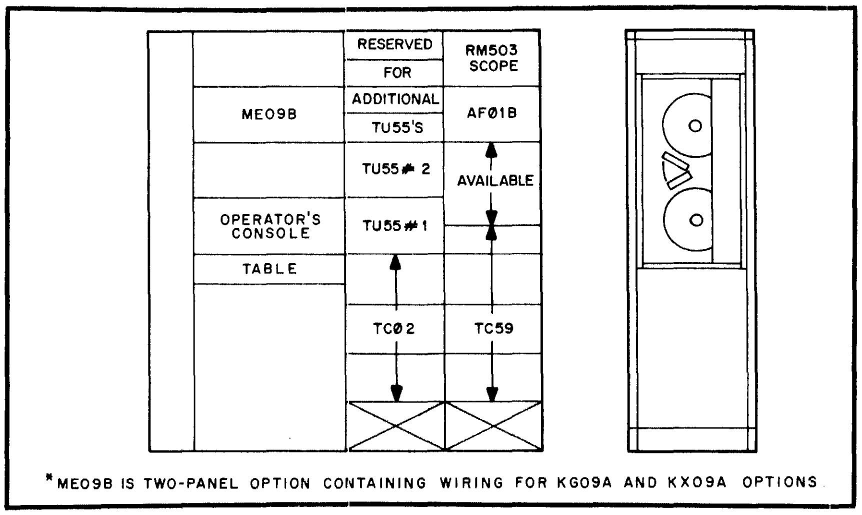

RESERVED FOR ADDITIONAL TU55'S
RM503 SCOPE
ME09B
AF01B
TU55 # 2
AVAILABLE
OPERATOR'S CONSOLE
TU55 # 1
TABLE
TC02
TC59
*ME09B IS TWO-PANEL OPTION CONTAINING WIRING FOR KG09A AND KX09A OPTIONS

Figure 15-9 Typical PDP-9/L System

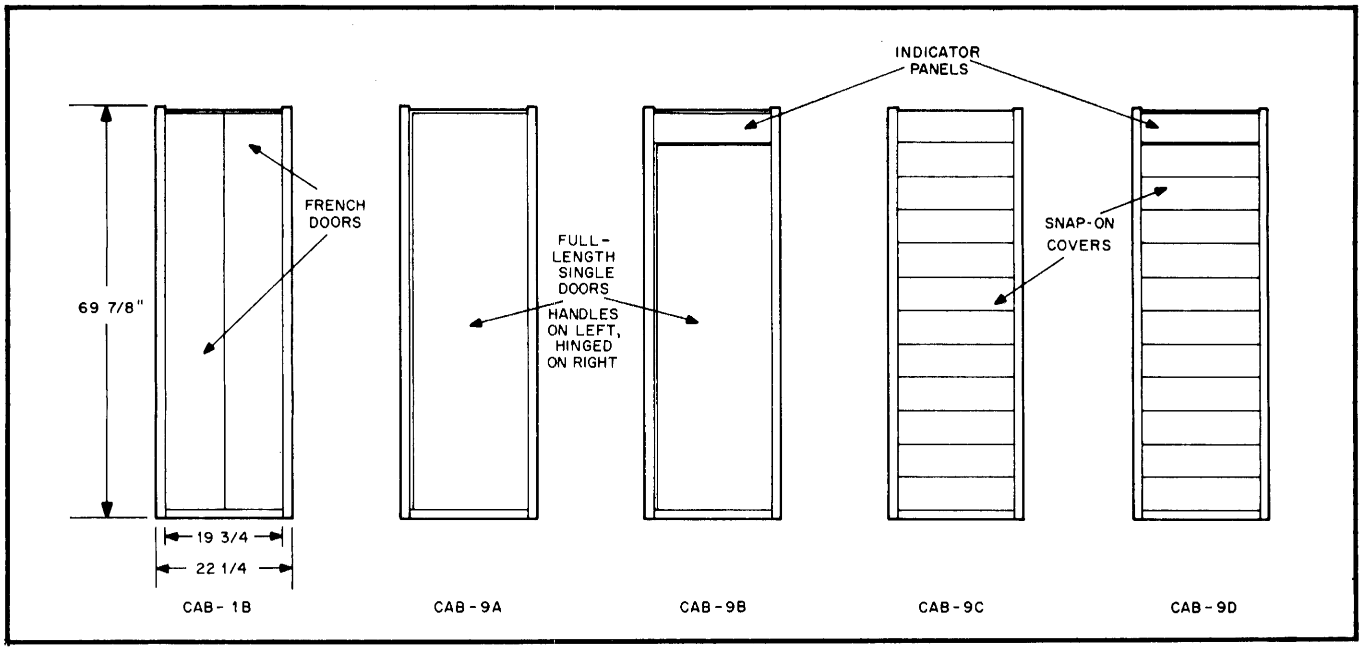


Figure 15-10 Cabinet Configurations

sufficiently by feet or casters to allow clearance for the cables.

## ADDING SPECIAL INTERFACES

Special interfaces may be constructed by using compatible FLIP CHIP modules and mounting hardware (see Digital Logic Handbook, C-105, for details).

A choice of cabinets is available for use with PDP-9/L systems. All hold standard FLIP CHIP mounting hardware (19-in. panels) and are available with or without end penels. Rotary fans are contained in the bottom, and power supplies may be purchased for mounting on the rear plenum doors.

The available cabinet options are illustrated in figure 15-10. A brief description of each is as follows:

| | |
|---|---|
| CAB-1B: | French doors front and rear; without indicator panel. |
| CAB-9A: | single full doors front and rear; without indicator panel. |
| CAB-9B: | single full doors front and rear; *with* indicator panel. |
| CAB-9C: | black snap-on covers front, full single door rear; without indicator panel. |
| CAB-9D: | black snap-on covers front, full single door rear; *with* indicator panel. |

All special interfaces should be designed to interface to the PDP-9/L I/O bus by using I/O bus cables, Type BC09A.

TABLE 15-1 PDP-9/L, EXTRA MEMORY, FREE-STANDING OPTIONS AND THEIR CONTROLS

| Name | Cabinet Dimensions Height (in.) | Cabinet Dimensions Width (in.) | Cabinet Dimensions Depth (w/table) | Options Required | Weight (lb.) | Service Clearance Front (in.) | Service Clearance Rear (in.) | Height of Interface (19 in. logic) | AC Current Nominal (amps) | AC Current Surge (amps) | Heat Dissipation (btu/hr) | Power Dissipation (kw) | Cable Length | Comments |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Standard PDP-9/L | 69-1/8 | 33 | 53 | ----- | 900 | Chair Space | 30 | ---- | 17 | 34 | | | | |
| Magnetic Tape Transports, Type TU20 TU20A | 69-1/8 | 22-1/4 | 27-1/16 | TC59 | 400 | 19 | 19 | See TC59 | 8 | 12 | 2300 | 0.62 | 10 ft | Op. Temp. 55-100°F Humidity 25-95% |
| 200 cpm Card Reader, Type CR02B | 50 | 30 | 17 | DA09A* | 200 | -- | 6-5/8 | ---- | 1.3 | 7.0 | 495 | 0.15 | 10 ft | |
| Line Printer, Type 647 | 52-57 | 56 | 30-1/4 | DA09A* | 1350 | 24 | 26 | ---- | 13 | -- | 2700 | 1.56 | 10 ft | |
| Incremental Plotter, | | | | | | | | | | | | | | |
| Calcomp Model 563 | 9-3/4 | 39-3/8 | 14-3/4 | 350 | 53 | -- | -- | See 350 | 1.12 | 2 | 425 | 0.125 | 10 ft | |
| Calcomp Model 565 | 9-3/4 | 18 | 14-3/4 | 350 | 53 | -- | -- | Control | 1.5 | 3 | 580 | 0.17 | 10 ft | |
| Precision CRT, Type 343 Type 30D | 69-1/8 | 22-1/4 | 51 | DA09A* | 350 | 9 | 36 | 5-1/4 | 8 | 8 | 3140 | 0.9 | 25 ft | |
| Slave Display, Type 343 | 69-1/8 | 22-1/4 | 51 | ---- | 350 | 9 | 36 | ---- | 6 | 10 | 2350 | 0.69 | 25 ft | |
| Data Communications System, Type 680 | (See 680 Handbook) | | | DB98A | -- | -- | -- | See DB98A | -- | -- | --- | -- | 12 ft | |
| Empty Cabinet | 69-1/8 | 22-1/4 | 27-1/16 | ---- | 100 | 19 | 19 | ---- | -- | -- | --- | -- | 12 ft | |

*No charge for required options marked with an asterisk (*).

TABLE 15-2 WIRED-IN OPTIONS

| Name | Included In | AC Current (amps) | | Heat Dissipation | Power Dissipation |
|---|---|---|---|---|---|
| | | Nominal | Surge | (btu/hr) | (kw) |
| Extended Arithmetic Element, Type KE09A | PDP-9 CPU | | | 108 | 0.032 |
| Automatic Priority Interrupt, Type KF09A | PDP-9 I/O Logic | | | 61 | 0.018 |
| Power Failure Detection, Type KP09A | PDP-9 I/O Logic | Included in Basic PDP-9/L Power System | | 69 | 0.021 |
| Oscilloscope Display Control Type 34H | PDP-9 I/O Logic | | | 408 | 0.012 |
| Memory Extension Control Type KG09A | ME09B | | | 94 | 0.030 |
| Memory Protection Option, Type KX09A | ME09B | | | 25 | 0.008 |

TABLE 15-3 HARDWARE AND LOGIC OPTIONS FOR 19-INCH CABINETS

| Name | Logic Height (in.) | Number of Mounting Panels | Options Required | Approx Weight (lb.) | AC Current (amps) Nominal | AC Current (amps) Surge | Heat Dissipation (btu/hr) | Power Dissipation (kw) | Comments |
|---|---|---|---|---|---|---|---|---|---|
| Parity-Extension-Protection Chasis, Type ME09B | 10-1/2 | 2 | — | 25 | | | | | |
| I/O Bus Adapter, Type DA09A | 10-1/2 | 2 | — | | 0.44 | 0.8 | 156 | 0.046 | |
| IPB, Type DB99A | 10-1/2 | 2 | — | | 0.39 | 0.72 | 133 | 0.042 | |
| IPB, Type DB98A | | | | | | | | | |
| PDP-9 End | 10-1/2 | 2 | — | | 0.39 | 0.72 | 133 | 0.042 | |
| PDP-8 End | 10-1/2 | 2 | — | | 0.31 | 0.6 | 111 | 0.032 | |
| IPB, Type DB97A | | | | | | | | | |
| PDP-9/L End | 5-1/4 | 1 | — | 13 | 0.29 | 0.54 | 98 | 0.031 | |
| PDP-7 End | 5-1/4 | 1 | — | | | | | | |
| Output Relay Buffer, Type DR09A | 5-1/4 | 1 | — | | 0.47 | 0.86 | 170 | 0.050 | |
| DECtape Control, Type TC02 | 15-3/4 | 3 | — | 38 | 0.60 | 1.0 | 207 | 0.058 | |
| DECtape Transport, Type TU55 | 10-1/2 | 2 | TC02 | 35 | 0.5 | 3.0 | 410 | 0.170 | |
| Magnetic Tape Control, Type TC59 | 21 | 4 | — | 50 | 2.0 | 5.0 | 1000 | 0.23 | |
| Incremental Plotter Control, Type 350 | 10-1/2 | 2 | — | 25 | | | | | |
| Teletype Control, Type LT09A | 10-1/2 | 2 | — | 25 | | | 47 | 0.014 | |
| Line Unit, Type LT09B | – | – | LT09A | | 0.13 | 0.24 | – | – | |
| Bit Sync Data Comm. System, Type 637 (DP09A, DP01B) | 10-1/2 | 2 | — | 25 | 0.27 | 0.50 | 100 | 0.029 | |

# APPENDIX 1
## INSTRUCTION SUMMARY

### MEMORY REFERENCE INSTRUCTIONS

| Mnemonic Symbol | Octal Code | Machine Cycles | Operation Executed |
|---|---|---|---|
| CAL | 00 | 2 | Call subroutine. The address portion of this instruction is ignored. The action is identical to JMS 20. |
| DAC Y | 04 | 2 | Deposit AC. The content of the AC is deposited in the memory cell at location Y. |
| JMS Y | 10 | 2 | Jump to subroutine. The content of the PC and the content of the L are deposited in memory cell Y. The next instruction is taken from cell Y + 1. |
| DZM Y | 14 | 2 | Deposit zero in memory. Zero is deposited in memory cell Y. |
| LAC Y | 20 | 2 | Load AC. The content of Y is loaded into the AC. |
| XOR Y | 24 | 2 | Exclusive OR. The exclusive OR is performed between the content of Y and the content of the AC, with the result left in the AC. |
| ADD Y | 30 | 2 | Add (1's complement). The content of Y is added to the content of the AC in 1's complement arithmetic and the result is left in the AC. |
| TAD Y | 34 | 2 | Two's complement add. The content of Y is added to the content of the AC in 2's complement arithmetic and the result is left in the AC. |
| XCT Y | 40 | 1+ | Execute. The instruction in memory cell Y is executed. |
| ISZ Y | 44 | 2 | Increment and skip if zero. The content of Y is incremented by one in 2's complement arithmetic. If the result is zero, the next instruction is skipped. |
| AND Y | 50 | 2 | AND. The logical operation AND is performed between the content of Y and the content of the AC with the result left in the AC. |

MEMORY REFERENCE INSTRUCTIONS (continued)

| Mnemonic Symbol | Octal Code | Machine Cycles | Operation Executed |
|---|---|---|---|
| SAD Y | 54 | 2 | Skip if AC is different from Y. The content of Y is compared with the content of the AC. If the numbers are different, the next instruction is skipped. |
| JMP Y | 60 | 1 | Jump to Y. The next instruction to be executed is taken from memory cell Y. |

EAE INSTRUCTION LIST

| Mnemonic Symbol | Octal Code | Operation Executed |
|---|---|---|
| EAE | 640000 | Basic EAE command. No operation. |
| LRS | 640500 | Long right shift. |
| LRSS | 660500 | Long right shift, signed (AC sign = link). |
| LLS | 640600 | Long left shift. |
| LLSS | 660600 | Long left shift, signed (AC sign = L). |
| ALS | 640700 | Accumulator left shift. |
| ALSS | 660700 | Accumulator left shift, signed (AC sign = L). |
| NORM | 640444 | Normalize, unsigned. Maximum shift is $44_8$. |
| NORMS | 660444 | Normalize, signed (AC sign = L). |
| MUL | 653122 | Multiply, unsigned. The number in the AC is multiplied by the number in the next core memory address. |
| MULS | 657122 | Multiply, signed. The number in the AC is multiplied by the number in the next core memory address. |
| DIV | 640323 | Divide, unsigned. The 36-bit content of both the AC and MQ is divided by the number in the next core memory location. |
| DIVS | 644323 | Divide, signed. The content of both the AC and MQ as a 1's complement signed number is divided by the number in the next core memory location. |

EAE INSTRUCTION LIST (continued)

| Mnemonic Symbol | Octal Code | Operation Executed |
|---|---|---|
| IDIV | 653323 | Integer divide, unsigned. Divide the number in the AC as an 18-bit unsigned integer by the number in the next core memory location. |
| IDIVS | 657323 | Integer divide, signed. Same as IDIV but the content of the AC is a 17-bit signed number. |
| FRDIV | 650323 | Fraction divide, unsigned. Divide the 18-bit fraction in the AC by the 18-bit fraction in the number in the next core memory location. |
| FRDIVS | 654323 | Fraction divide, signed. Same as FRDIV, but the content of the AC is a 17-bit signed number. |
| LACQ | 641002 | Replace the content of the AC with the content of the MQ. |
| LACS | 641001 | Replace the content of the AC with the content of the SC. |
| CLQ | 650000 | Clear MQ. |
| ABS | 644000 | Place absolute value of AC in the AC. |
| GSM | 664000 | Get sign and magnitude. Places AC sign in the link and takes the absolute value of AC. |
| OSC | 640001 | Inclusive OR the SC into the AC. |
| OMQ | 640002 | Inclusive OR AC with MQ and place results in AC. |
| CMQ | 640004 | Complement the MQ. |
| LMQ | 652000 | Load MQ. |

INPUT/OUTPUT TRANSFER INSTRUCTIONS

| Mnemonic Symbol | Octal Code | Operation Executed |
|---|---|---|
| | | <u>Program Interrupt</u> |
| IOF | 700002 | Interrupt off. Disable the PIC. |
| ION | 700042 | Interrupt on. Enable the PIC. |

INPUT/OUTPUT TRANSFER INSTRUCTIONS (continued)

| Mnemonic Symbol | Octal Code | Operation Executed |
|---|---|---|
| | | Real Time Clock |
| CLSF | 700001 | Skip the next instruction if the clock flag is set to 1. |
| CLOF | 700004 | Clear the clock flag and disable the clock. |
| CLON | 700044 | Clear the clock flag and enable the clock. |
| | | Perforated Tape Reader |
| RSF | 700101 | Skip if reader is a 1. |
| RCF | 700102 | Clear reader flag, then inclusively OR the content of reader buffer into the AC. |
| RRB | 700112 | Read reader buffer. Clear reader flag and AC, and then transfer content of reader buffer into AC. |
| RSA | 700104 | Select reader in alphanumeric mode. One 8-bit character is read into the reader buffer. |
| RSB | 700144 | Select reader in binary mode. Three 6-bit characters are read into the reader buffer. |
| | | Perforated Tape Punch |
| PSF | 700201 | Skip if the punch flag is set to 1. |
| PCF | 700202 | Clear the punch flag. |
| PSA or PLS | 700204 700206 | Punch a line of tape in alphanumeric mode. |
| PSB | 700244 | Punch a line of tape in binary mode. |
| | | I/O Equipment |
| IORS | 700314 | Input/output read status. The content of given flags replace the content of the assigned AC bits. |
| TTS | 703301 | Test Teletype and skip if KSR 33 is connected to computer. |
| CAF | 703302 | Clear all flags. |
| SKP7 | 703341 | Skip if processor is a PDP-7 or PDP-9 |

INPUT/OUTPUT TRANSFER INSTRUCTIONS (continued)

| Mnemonic Symbol | Octal Code | Operation Executed |
|---|---|---|
| | | Teletype Keyboard |
| KSF | 700301 | Skip if the keyboard flag is set to 1. |
| KRB | 700312 | Read the keyboard buffer. The content of the buffer is placed in AC10-17 and the keyboard flag is cleared. |
| | | Teletype Teleprinter |
| TSF | 700401 | Skip if the teleprinter flag is set. |
| TCF | 700402 | Clear the teleprinter flag. |
| TLS | 700406 | Load teleprinter buffer. The content of AC10-17 is placed in the buffer and printed. The flag is cleared before transmission takes place and is set when the character has been printed. |
| | | Types 30 and 34 Oscilloscope and Precision CRT Displays |
| DXC | 700502 | Clear the X-coordinate buffer. |
| DYC | 700602 | Clear the Y-coordinate buffer. |
| DXL | 700506 | Load the X-coordinate buffer from AC8-17. |
| DYL | 700606 | Load the Y-coordinate buffer from AC8-17. |
| DXS | 700546 | Load the X-coordinate buffer and display the point specified by the XB and YB. |
| DYS | 700646 | Load the Y-coordinate buffer and display the point specified by the XB and YB. |
| DSF | 700501 | Skip if display flag = 1. |
| DCF | 700702 | Clear display flag. |
| DLB | 700706 | Load the brightness register from AC15-17. (for Type 30) |
| -- | 700704 | Load the brightness register from AC16-17. (for Type 34) |

INPUT/OUTPUT TRANSFER INSTRUCTIONS (continued)

| Mnemonic Symbol | Octal Code | Operation Executed |
|---|---|---|
| | | Type AF01B Analog-to-Digital Converter and Multiplexer |
| ADSM | 701103 | Select MX channel. The content of AC12-17 is placed in MAR. |
| ADIM | 701201 | Increment channel address. The content of the MAR is incremented by 1. Channel 0 follows channel $77_8$. |
| ADRM | 701212 | Read MAR into AC12-17. |
| ADSF | 701301 | Skip if converter flag is set. |
| ADSC | 701304 | Select and convert. The converter flag is cleared and a conversion is initiated. |
| ADRB | 701312 | Read converter buffer. Places the content of the buffer into the AC. |
| | | Type 139E General Purpose Multiplexer Control |
| ADSM | 701103 | Select MX channel. The content of AC12-17 is placed in the MAR. |
| ADIM | 701201 | Increment channel address. The content of the MAR is incremented by 1. Channel 0 follows channel $77_8$. |
| ADRM | 701212 | Read MAR into $AC_{12-17}$. |
| | | Type 138E Analog-to-Digital Converters |
| ADSF | 701301 | Skip if converter flag is set. |
| ADSC | 701304 | Select and convert. The converter flag is cleared and a conversion is initiated. |
| ADRB | 701312 | Read converter buffer. Places the content of the buffer in the AC. |
| | | Type DR09A Relay Buffer |
| ORC | 702101 | Clear output relay buffer flip-flop reigster. |
| ORS | 702104 | Set output relay buffer flip-flop register to correspond with the contents of the accumulator. |

INPUT/OUTPUT TRANSFER INSTRUCTIONS (continued)

| Mnemonic Symbol | Octal Code | Operation Executed |
|---|---|---|
| | | Type 350 Incremental Plotter and Control |
| PLSF | 702401 | Skip if plotter flag is a 1. |
| PLCF | 702402 | Clear plotter flag. |
| PLPU | 702404 | Plotter pen up. Raise pen off of paper. |
| PLPR | 702421 | Plotter pen right. |
| PLDU | 702422 | Plotter drum (paper) upward. |
| PLDD | 702424 | Plotter drum (paper) downward. |
| PLPL | 702441 | Plotter pen left. |
| PLUD | 702442 | Plotter drum (paper) upward. |
| PLPD | 702444 | Plotter pen down. Lower pen on to paper. |
| | | Type KF09A Automatic Priority Interrupt |
| SPI | 705501 | Skip on priorities inactive. |
| ISA | 705504 | Initiate selected activity. |
| DBK | 703304 | Debreak. |
| DBR | 703344 | Debreak and restore. |
| | | Type 647 Line Printer |
| LSDF | 706501 | Skip if the DONE flag is set. |
| LPCB | 706502 | Clear the DONE flag, clear control print buffer, enable DONE interrupt, initiate a clear sequence in the hue printer, set the DONE flag when the clear sequence is finished. |
| *LPD1 | 706504 | Disable DONE flag interrupt. |
| | 706522 | Clear DONE flag. |

*These instructions have been added to the Line Printer command set to allow enabling and disabling of the interrupt. Since power clear returns the system to the interrupt enabled condition programs generated for the PDP-7 line printer (647B), which does not have these instructions, will run correctly.

INPUT/OUTPUT TRANSFER INSTRUCTIONS (continued)

| Mnemonic Symbol | Octal Code | Operation Executed |
|---|---|---|
| | 706542 | Clear DONE flag. |
| | 706562 | Clear DONE flag. |
| LPL2 | 706526 | Load printing buffer with two characters; clear DONE FLAG: THE CONTENTS OF AC6-11 and AC12-17 are transferred to the printing buffer as 6-bit bytes in that order. The DONE flag will be set when the load sequence is finished. |
| LPLD | 706546 | Load the printing buffer with three characters. The DONE flag is cleared; the contents of AC 0-5, 6-11, and 12-17 are transferred as 6-bit bytes into the printing buffer in that order. The DONE flag is set at the completion of the load sequence. |
| LPL1 | 706566 | Load the printing buffer with one character; clear DONE flag; the contents of AC12-17 are transferred as a 6-bit byte into the printing buffer. The DONE flag is set at the completion of the load sequence. |
| LPEF | 706601 | Skip if the ERROR flag is set. |
| LPCF | 706602 | Clear DONE flag. |
| | 706622 | Clear DONE flag. |
| | 706642 | Clear DONE flag. |
| | 706662 | Clear DONE flag. |
| LPPB | 706606 | Select printer and initiate printing. The DONE flag is cleared; the contents of the printing buffer are printed; the printing buffer is cleared; the DONE flag is set when the printing sequence is completed. |
| LPLS | 706626 | Load spacing buffer and space; the DONE flag is cleared; the contents of AC15-17 are transferred into the spacing buffer; the paper is spaced vertically according to the format selected; the spacing buffer is cleared; the DONE flag is set. |
| LPPS | 706646 | Print and space. This instruction accomplishes the combined actions of LPPB and LPLS instructions. The DONE flag is cleared; the contents of AC15-17 are transferred to the spacing buffer; the contents of the printing buffer are printed; the paper is spaced vertically; the printing and spacing buffers are cleared; the DONE flag is set upon completion. |

| Mnemonic Symbol | Octal Code | Operation Executed |
|---|---|---|
| *LPEI | 706664 | The DONE flag interrupt is enabled. |
| | | **Type CR02B Card Reader** |
| CRSB | 706744 | Select and read a card in binary mode. A card is started through the reader and 80 columns are read as 12-bit numbers. The card done flag is cleared. |
| CRSA | 706704 | Select and read a card in alphanumeric mode. A card is started through the reader and 80 columns are read in 6-bit BCL codes. |
| CRRB | 706712 | Read the card reader buffer into AC bits 6-17. The column flag is cleared. |
| CRSF | 706701 | Skip on column data ready flag. |
| CRSD | 706721 | Skip on card done flag. |
| CRSR | 706741 | Skip on reader ready condition. |
| CREF | 706761 | Skip on reader EOF flag. |
| CRCD | 706724 | Clear done flag. |
| | | **Type TC59 Tape Control IOT Instructions** |
| MTSF | 707301 | Skip on error flag or magnetic tape flag (EF and MTF). |
| MTCR | 707321 | Skip on tape control ready (TCR). |
| MTTR | 707341 | Skip on tape transport ready (TTR). |
| MTAF | 707322 | Clear status and command registers and EF and MTF. |
| | 707324 | Inclusively OR content of $AC_{0-11}$ into command register. |
| MTLC | 707326 | Load content of $AC_{0-11}$ into command register. |

*These instructions have been added to the Line Printer command set to allow enabling and disabling of the interrupt. Since power clear returns the system to the interrupt enabled condition programs generated for the PDP-7 line printer (647B), which does not have these instructions, will run correctly.

| Mnemonic Symbol | Octal Code | Operation Executed |
|---|---|---|
| | | Type TC59 Tape Control IOT Instructions (continued) |
| MTCC | 707356 | Terminate write continuous mode. |
| | 707342 | Inclusively OR content of status register into $AC_{0-11}$. |
| MTRS | 707352 | Read content of status register into $AC_{0-11}$. |
| | 707302 | Inclusively OR content of command register into $AC_{0-11}$. |
| MTRC | 707312 | Read command register into $AC_{0-11}$. |
| MTGO | 707304 | Set "go" bit to execute command in command register. |
| | | TC02 DECtape Control IOT Instructions |
| DTCA | 707541 | Clear status register A. |
| DTRA | 707552 | Read status register A. |
| DTXA | 707544 | XOR status register A. |
| DTLA | 707545 | Load status register A. |
| DTEF | 707561 | Skip on error flag. |
| DTRB | 707572 | Read status B. |
| DTDF | 707601 | Skip on DECtape flag. |
| | | KG09A Memory Extension Control |
| SEM | 707701 | Skip if in extend mode. |
| EEM | 707702 | Enter extend mode. |
| LEM | 707704 | Leave extend mode. |
| | | KX09A Memory Protection |
| MPSNE | 701741 | Skip on NonExistent Memory Flag |
| MPSK | 701701 | Skip on Violation Flag |
| MPEV | 701742 | Enter Protect Mode |
| MPCV | 701702 | Clear Violation Flag |

INPUT/OUTPUT TRANSFER INSTRUCTIONS (continued)

| Mnemonic Symbol | Octal Code | Operation Executed |
|---|---|---|
| | | <u>KX09A Memory Protection (continued)</u> |
| MPCNE | 701744 | Clear NonExistent Memory Flag |
| MPLD | 701704 | Load boundary register from $AC_{3-7}$ |
| | | <u>KP09A Power Failure Protection</u> |
| | 703201 | Skip if Power-Low Flag is set |

OPERATE INSTRUCTIONS

| Mnemonic Symbol | Octal Code | Event Time | Operation Executed |
|---|---|---|---|
| OPR or NOP | 740000 | --- | Operate group or no operation. Causes a 1-cycle program delay. |
| CMA | 740001 | 3 | Complement accumulator. Each bit of the AC is complemented. |
| CML | 740002 | 3 | Complement link. |
| OAS | 740004 | 3 | Inclusive OR ACCUMULATOR switches. The word set into the ACCUMULATOR switches is OR combined with the content of the AC, the result remains in the AC. |
| RAL | 740010 | 3 | Rotate accumulator left. The content of the AC and L are rotated one position to the left. |
| RAR | 740020 | 2 | Rotate accumulator right. The content of the AC and L are rotated one position to the right. |
| HLT | 740040 | --- | Halt. The program is stopped at the conclusion of the cycle. |
| SMA | 740100 | 1 | Skip on minus accumulator. If the content of the AC is negative (2's complement) number the next instruction is skipped. |
| SZA | 740200 | 1 | Skip on zero accumulator. If the content of the AC equals zero (2's complement), the next instruction is skipped. |
| SNL | 740400 | 1 | Skip on non-zero link. If the L contains a 1, the next instruction is skipped. |

OPERATE INSTRUCTIONS (continued)

| Mnemonic Symbol | Octal Code | Event Time | Operation Executed |
|---|---|---|---|
| SKP | 741000 | 1 | Skip. The next instruction is unconditionally skipped. |
| SPA | 741100 | 1 | Skip on positive accumulator. If the content of the AC is zero (2's complement) or a positive number, the next instruction is skipped. |
| SNA | 741200 | 1 | Skip on non-zero accumulator. If the content of the AC is not zero (2's complement), the next instruction is skipped. |
| SZL | 741400 | 1 | Skip on zero link. If the L contains a 0, the next instruction is skipped. |
| RTL | 742010 | 2,3 | Rotate two left. The content of the AC and the L are rotated two positions to the left. |
| RTR | 742020 | 2,3 | Rotate two right. The content of the AC and the L are rotated two positions to the right. |
| CLL | 744000 | 2 | Clear link. The L is cleared. |
| STL | 744002 | 2,3 | Set link. The L is set to 1. |
| RCL | 744010 | 2,3 | Clear link, then rotate left. The L is cleared, then the L and AC are rotated one position left. |
| RCR | 744020 | 2,3 | Clear link, then rotate right. The L is cleared, then the L and AC are rotated one position right. |
| CLA | 750000 | 2 | Clear accumulator. Each bit of the AC is cleared. |
| CLC | 750001 | 2,3 | Clear and complement accumulator. Each bit of the AC is set to contain a 1. |
| LAS | 750004 | 2,3 | Load accumulator from switches. The word set into the ACCUMULATOR switches is loaded into the AC. |
| GLK | 750010 | 2,3 | Get link. The content of L is set into AC17. |
| LAW N | 76XXXX | --- | Load the AC with 76XXXX. |

# APPENDIX 2
# PDP-9 I/O CODES

## 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 |
| 0 | 260 | → | 337 |
| 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 | * Ignored by the operating system | |
| 9 | 271 | | |

● = HOLE PUNCHED = MARK = 1 BIT
☐ = NO HOLE PUNCHED = SPACE = 0 BIT

MOST SIGNIFICANT BIT
LEAST SIGNIFICANT BIT

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | **8** | **7** | **6** | **5** | **4** | **S** | **3** | **2** | **1** |
| | @ | * | SPACE | | NULL (IDLE) | | | | | O | O | | O | O | O |
| | A | | ! | * | START OF MESSAGE | | | | | O | O | | O | O | ● |
| | B | | " | * | END OF ADDRESS (EOA) | | | | | O | O | | O | ● | O |
| | C | | # | * | END OF MESSAGE (EOM) | | | | | O | O | | O | ● | ● |
| | D | | $ | * | END OF TRANSMISSION (EOT) | | | | | O | O | | ● | O | O |
| | E | | % | * | WHO ARE YOU (WRU) | | | | | O | O | | ● | O | ● |
| | F | | & | * | ARE YOU (RU) | | | | | O | O | | ● | ● | O |
| | G | | ' | * | BELL | | | | | O | O | | ● | ● | ● |
| | H | | ( | * | FORMAT EFFECTOR | | | | | O | ● | | O | O | O |
| | I | | ) | * | HORIZONTAL TAB | | | | | O | ● | | O | O | ● |
| | J | | * | * | LINE FEED | | | | | O | ● | | O | ● | O |
| | K | | + | * | VERTICAL TAB | | | | | O | ● | | O | ● | ● |
| | L | | , | | FORM FEED | | | | | O | ● | | ● | O | O |
| | M | | – | | CARRIAGE RETURN | | | | | O | ● | | ● | O | ● |
| | N | | . | | SHIFT OUT | | | | | O | ● | | ● | ● | O |
| | O | | / | | SHIFT IN | | | | | O | ● | | ● | ● | ● |
| | P | | 0 | | DCO | | | | | ● | O | | O | O | O |
| | Q | | 1 | | READER ON | | | | | ● | O | | O | O | ● |
| | R | | 2 | | TAPE (AUX ON) | | | | | ● | O | | O | ● | O |
| | S | | 3 | | READER OFF | | | | | ● | O | | O | ● | ● |
| | T | | 4 | | (AUX OFF) | | | | | ● | O | | ● | O | O |
| | U | | 5 | | ERROR | | | | | ● | O | | ● | O | ● |
| | V | | 6 | | SYNCHRONOUS IDLE | | | | | ● | O | | ● | ● | O |
| | W | | 7 | | LOGICAL END OF MEDIA | | | | | ● | O | | ● | ● | ● |
| | X | | 8 | | SO | | | | | ● | ● | | O | O | O |
| | Y | | 9 | | S1 | | | | | ● | ● | | O | O | ● |
| | Z | | : | | S2 | | | | | ● | ● | | O | ● | O |
| | [ | * | ; | | S3 | | | | | ● | ● | | O | ● | ● |
| ACK | \ | * | < | * | S4 | | | | | ● | ● | | ● | O | O |
| ALT MODE | ] | * | = | * | S5 | | | | | ● | ● | | ● | O | ● |
| | ↑ | * | > | * | S6 | | | | | ● | ● | | ● | ● | O |
| RUB OUT | ← | * | ? | * | S7 | | | | | ● | ● | | ● | ● | ● |

| | | | | |
|---|---|---|---|---|
| NON-TYPING | ● | O | O | SAME |
| | ● | O | ● | SAME |
| | ● | ● | O | SAME |
| NON-TYPING | ● | ● | ● | SAME |

* OBTAINED WITH SHIFT KEY

## TELETYPE CODE COMPARISON

| Character Name | Flexowriter FIODEC Code | 28 KSR Baudot Code | 33 or 35 KSR ASCII Code |
|---|---|---|---|
| | 0-9 | 0-9 | 0-9 |
| | a-z | A-Z | A-Z |
| | A-Z | $A-$Z | A-Z |
| period | . | . | . |
| | , | , | , |
| minus sign | — | — | — |
| | / | / | / |
| center dot, period | : | : | : |
| center dot, comma | ; | ; | ; |
| | ( | ( | ( |
| | ) | ) | ) |
| | + | & | + |
| | ↑ | ! | ↑ |
| multiply | x | # | * |
| | " | $" | " |
| | ' | $' | ' |
| | = | $: | = |
| | [ | $( | [ |
| | ] | $) | ] |
| | < | $- | < |
| | > | $& | > |
| | ~ | $? | ← |
| | ⊃ | $, | % |
| | ∨ | $/ | ! |
| | ∧ | $# | & |
| | → | $; | @ |
| vertical stroke | \| | $! | \ |
| underbar | _ | " | $ |
| center | · | $. | none |
| overbar | ‾ | ' | # |
| | stop code | none | form feed |
| | tab | ↓ bell | ↓ tab |

## LINE PRINTER ASCII CODE IN OCTAL FORM

| Character | 6-Bit Trimmed Code (in octal) | Character | 6-Bit Trimmed Code (in octal) |
|---|---|---|---|
| A | 01 | 6 | 66 |
| B | 02 | 7 | 67 |
| C | 03 | 8 | 70 |
| D | 04 | 9 | 71 |
| E | 05 | ! | 41 |
| F | 06 | " | 42 |
| G | 07 | # | 43 |
| H | 10 | $ | 44 |
| I | 11 | % | 45 |
| J | 12 | & | 46 |
| K | 13 | ' | 47 |
| L | 14 | ( | 50 |
| M | 15 | ) | 51 |
| N | 16 | * | 52 |
| O | 17 | + | 53 |
| P | 20 | , | 54 |
| Q | 21 | - | 55 |
| R | 22 | . | 56 |
| S | 23 | / | 57 |
| T | 24 | : | 72 |
| U | 25 | ; | 73 |
| V | 26 | < | 74 |
| W | 27 | = | 75 |
| X | 30 | > | 76 |
| Y | 31 | ? | 77 |
| Z | 32 | @ | 00 |
| 0 | 60 | [ | 33 |
| 1 | 61 | \ | 34 |
| 2 | 62 | ] | 35 |
| 3 | 63 | ↑ | 36 |
| 4 | 64 | ← | 37 |
| 5 | 65 | Space | 40 |

## TYPE CR01E CARD READER, INTERNAL ALPHANUMERIC CODES

| 12 11 0 / 1-9 | ZONE | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | No Punch | | | 0 | | | 11 | | | 12 | | |
| | Internal | 029 | 026 | Internal | 029 | 026 | Internal | 029 | 026 | Internal | 029 | 026 |
| No Punch | 00* | Blank | Blank | 20* | 0 | 0 | 40 | - | - | 60 | & | +[&] |
| 1 | 01 | 1 | 1 | 21 | / | / | 41 | J | J | 61 | A | A |
| 2 | 02 | 2 | 2 | 22 | S | S | 42 | K | K | 62 | B | B |
| 3 | 03 | 3 | 3 | 23 | T | T | 43 | L | L | 63 | C | C |
| 4 | 04 | 4 | 4 | 24 | U | U | 44 | M | M | 64 | D | D |
| 5 | 05 | 5 | 5 | 25 | V | V | 45 | N | N | 65 | E | E |
| 6 | 06 | 6 | 6 | 26 | W | W | 46 | O | O | 66 | F | F |
| 7 | 07 | 7 | 7 | 27 | X | X | 47 | P | P | 67 | G | G |
| 8 | 10 | 8 | 8 | 30 | Y | Y | 50 | Q | Q | 70 | H | H |
| 9 | 11 | 9 | 9 | 31 | Z | Z | 51 | R | R | 71 | I | I |
| 8-2 | 12 | : | | 32 | ‡ † | | 52 | ! | | 72 | ¢ | |
| 8-3 | 13 | # | =[#] | 33 | , | , | 53 | $ | $ | 73 | . | . |
| 8-4 | 14 | @ | '[@] | 34 | % | ([%] | 54 | * | * | 74 | < | )[¤] |
| 8-5 | 15 | ' | | 35 | _ | | 55 | ) | | 75 | ( | |
| 8-6 | 16 | = | | 36 | > | | 56 | ; | | 76 | + | |
| 8-7 | 17 | " | | 37 | ? | | 57 | ¬ | | 77 | \| | |

* A blank column appears as code 00 and a 0-zone punch (alone) appears as 20. To transform this to IBM compatible tape BCD, a programmed reversal of these two codes must take place.

+ Non-printing character.

TYPE CR02B CARD READER CODE (HOLLERITH) IN OCTAL FORM

| Character | Octal Code | Character | Octal Code | Character | Octal Code | Character | Octal Code |
|---|---|---|---|---|---|---|---|
| A | 61 | M | 44 | Y | 30 | + | 60 |
| B | 62 | N | 45 | Z | 31 | — | 40 |
| C | 63 | O | 46 | 0 | 12 | / | 21 |
| D | 64 | P | 47 | 1 | 01 | = | 13 |
| E | 65 | Q | 50 | 2 | 02 | , | 33 |
| F | 66 | R | 51 | 3 | 03 | $ | 53 |
| G | 67 | S | 22 | 4 | 04 | . | 73 |
| H | 70 | T | 23 | 5 | 05 | ' | 14 |
| I | 71 | U | 24 | 6 | 06 | ( | 34 |
| J | 41 | V | 25 | 7 | 07 | * | 54 |
| K | 42 | W | 26 | 8 | 10 | ) | 74 |
| L | 43 | X | 27 | 9 | 11 | | |
| | | | | | | blank | 00 |

TYPE CR02B CARD READER CODE (HOLLERITH) IN BINARY FORM

| Low order bits | High order bits | | | |
|---|---|---|---|---|
| | 00 | 01 | 10 | 11 |
| 0000 | | blank | — | + [ &] |
| 0001 | 1 | / | J | A |
| 0010 | 2 | S | K | B |
| 0011 | 3 | T | L | C |
| 0100 | 4 | U | M | D |
| 0101 | 5 | V | N | E |
| 0110 | 6 | W | O | F |
| 0111 | 7 | X | P | G |
| 1000 | 8 | Y | Q | H |
| 1001 | 9 | Z | R | I |
| 1010 | 0 | | | |
| 1011 | = [ # ] | , | $ | . |
| 1100 | ' [@] | ( [ % ] | * | ) [ □] |

## HOLLERITH CARD CODE - TYPE 26 PUNCH

| digit | Zone | | | |
|---|---|---|---|---|
| | no zone | 12 | 11 | 0 |
| no punch | blank | + [ &] | — | 0 |
| 1 | 1 | A | J | / |
| 2 | 2 | B | K | S |
| 3 | 3 | C | L | T |
| 4 | 4 | D | M | U |
| 5 | 5 | E | N | V |
| 6 | 6 | F | O | W |
| 7 | 7 | G | P | X |
| 8 | 8 | H | Q | Y |
| 9 | 9 | I | R | Z |
| 8-3 | = [ #] | . | $ | , |
| 8-4 | ' [@] | ) [□] | * | ( [ % ] |

## HOLLERITH CARD CODE - TYPE 29 PUNCH

| digit | Zone | | | |
|---|---|---|---|---|
| | no zone | 12 | 11 | 0 |
| no punch | blank | & | — | 0 |
| 1 | 1 | A | J | / |
| 2 | 2 | B | K | S |
| 3 | 3 | C | L | T |
| 4 | 4 | D | M | U |
| 5 | 5 | E | N | V |
| 6 | 6 | F | O | W |
| 7 | 7 | G | P | X |
| 8 | 8 | H | Q | Y |
| 9 | 9 | I | R | Z |
| 8-2 | : | ¢ | ! | |
| 8-3 | # | . | $ | , |
| 8-4 | @ | < | * | % |
| 8-5 | \| | ( | ) | _ |
| 8-6 | = | + | ; | > |
| 8-7 | " | \| | | ? |

# APPENDIX 3

# 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

**Addition** **Multiplication**

**Binary Scale**

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

$$0\times 0=0 \qquad 0\times 1=1\times 0=0 \qquad 1\times 1=1$$

**Octal Scale**

| 0 | 01 | 02 | 03 | 04 | 05 | 06 | 07 |
|---|---|---|---|---|---|---|---|
| 1 | 02 | 03 | 04 | 05 | 06 | 07 | 10 |
| 2 | 03 | 04 | 05 | 06 | 07 | 10 | 11 |
| 3 | 04 | 05 | 06 | 07 | 10 | 11 | 12 |
| 4 | 05 | 06 | 07 | 10 | 11 | 12 | 13 |
| 5 | 06 | 07 | 10 | 11 | 12 | 13 | 14 |
| 6 | 07 | 10 | 11 | 12 | 13 | 14 | 15 |
| 7 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |

| 1 | 02 | 03 | 04 | 05 | 06 | 07 |
|---|---|---|---|---|---|---|
| 2 | 04 | 06 | 10 | 12 | 14 | 16 |
| 3 | 06 | 11 | 14 | 17 | 22 | 25 |
| 4 | 10 | 14 | 20 | 24 | 30 | 34 |
| 5 | 12 | 17 | 24 | 31 | 36 | 43 |
| 6 | 14 | 22 | 30 | 36 | 44 | 52 |
| 7 | 16 | 25 | 34 | 43 | 52 | 61 |

## MATHEMATICAL CONSTANTS IN OCTAL SCALE

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

## POWERS OF TWO

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

# OCTAL-DECIMAL INTEGER CONVERSION TABLE

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

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

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

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

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

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

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

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

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

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

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

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

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

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

# OCTAL-DECIMAL INTEGER CONVERSION TABLE (continued)

| 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 (continued)

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

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

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

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

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 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 | 3584 |
| to | to |
| 7777 | 4095 |
| (Octal) | (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 (continued)

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

## OCTAL-DECIMAL FRACTION CONVERSION TABLE (continued)

| 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 | .001201 | .000573 | .001445 | .000673 | .001689 | .000773 | .001934 |
| .000474 | .001205 | .000574 | .001449 | .000674 | .001693 | .000774 | .001937 |
| .000475 | .001209 | .000575 | .001453 | .000675 | .001697 | .000775 | .001941 |
| .000476 | .001213 | .000576 | .001457 | .000676 | .001701 | .000776 | .001945 |
| .000477 | .001216 | .000577 | .001461 | .000677 | .001705 | .000777 | .001949 |