DDP-416
USERS GUIDE

Honeywell

COMPUTER CONTROL
DIVISION

Doc.No. 130071630
M-1040

DDP-416
USERS GUIDE

June 1967

Honeywell

COMPUTER CONTROL
DIVISION

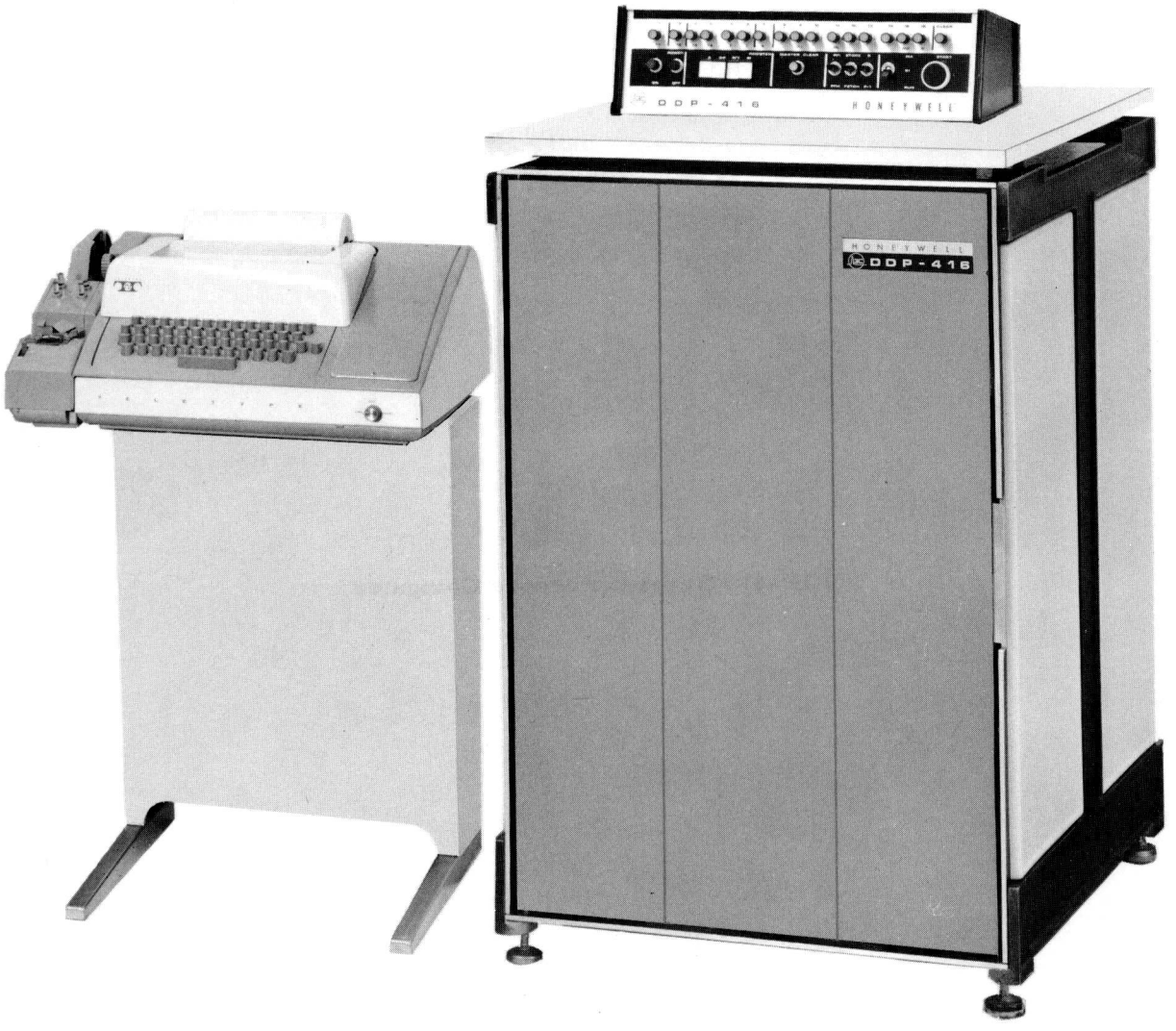
Published by the Publications Department,
Honeywell Inc., Computer Control Division

CONTENTS

CONTENTS (Cont)

ILLUSTRATIONS

TABLES



DDP-416 General Purpose Computer

SECTION I
INTRODUCTION

This User's Guide is intended to familiarize you with the operation of your DDP-416 computer. From a user's point of view, the DDP-416 computer is a relatively simple device to operate. Its movable control console is designed to allow complete operator freedom which provides for easy, error-free operation. The comprehensive software package supplied with the DDP-416 includes a DAP-16 assembler and a variety of utility, input/output, and mathematical routines. Furthermore, with each DDP-416, Honeywell provides a complete library of instruction manuals and program listings that tell the programmer/operator all he needs to know about preparing and using DAP-16 programs.

The DAP-16 Manual (3C Doc. No. 130071629) presents complete instructions for preparing programs in the DDP-416 source language. The Programmer's Reference Manual (3C Doc. No. 130071628) lists and describes all the instructions in the DDP-416 repertoire.

In the User's Guide you will find detailed instructions for using your computer and its software package. The guide is sectionalized for easy referencing such that information you will refer to again and again can be found easily. At the same time, you can progressively increase your knowledge of the DDP-416 by reading the sections in the order they appear. Sections II and III will be of greatest interest to you initially. Section II explains the function of each of the controls on the control console and presents a few simple procedures for using them to perform basic operations. Section III lists, by type and function, all of the programs in your software package with the mnemonic, document number and equipment required. Section IV contains generalized operating procedures for system tapes and for some of the more frequently used utility programs. The remainder of the User's Guide contains reference material designed to help you make effective use of your DDP-416 and its software package.

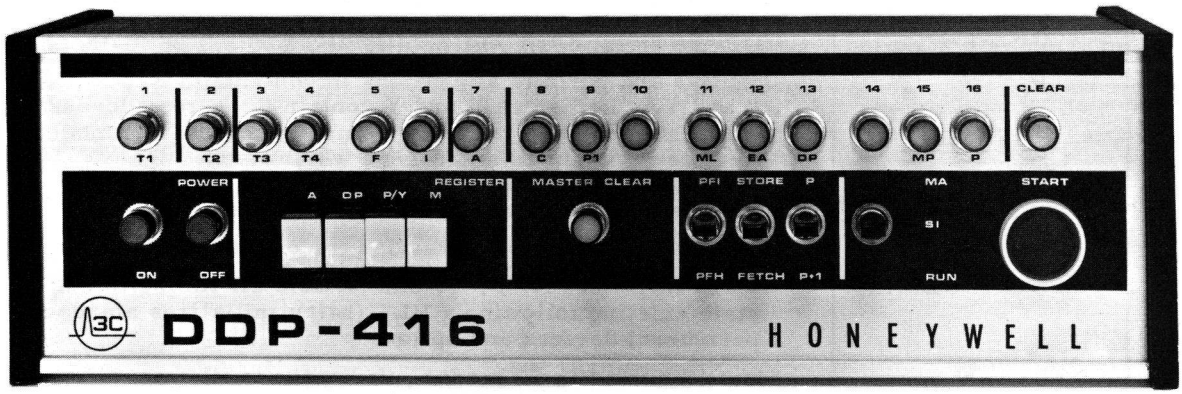
SECTION II
CONTROL CONSOLE

The basic DDP-416 computer system includes a main frame, a control console and a typewriter. In addition to this standard equipment, your installation may include any or all of the following peripheral devices:

High speed paper tape reader
High speed paper tape punch
Card reader
Magnetic tape units

The DDP-416 Programmer's Reference Manual (3C Doc. No. 130071628) and the appropriate peripheral device option manuals contain operating instructions for each device, standard or optional. The software package contains the input/output subroutines required for console control of the devices included in your installation.

The control console (Figure 2-1) includes the controls and indicators for normal system operation. These controls and indicators are listed and described in Tables 2-1 through 2-3.



3760

Figure 2-1. DDP-416 Control Console, Control Panel

Table 2-1.
Registers Displayed on DDP-416 Control Panel

Register	Bits	Display
A	1-16	Register A (primary arithmetic and logic register).
OP	1-16	State of key flip-flops in system. (Refer to Table 2-4.)
P/Y	1-16	Register Y (memory address register). When the computer is operating in the memory access (MA) mode, Registers P and Y contain the same address. When in the single instruction (SI) or RUN mode Register P contains an address which is one greater than the address contained in Register Y.
M	1-16	Register M (memory buffer register). Contains the contents of that memory location specified by Register Y.

Table 2-2.
Function of Control Panel Pushbuttons and Switches

Pushbutton	Function
16 Indicator	Enter or display data. A ONE is indicated when pushbutton/indicator is illuminated. A ZERO is indicated when pushbutton/indicator is extinguished.
CLEAR	Clears displayed registers (A, P/Y or M only).
POWER-ON	Applies power to main frame. The indicator is illuminated when power is applied.
POWER-OFF	Removes power from main frame.
A	Displays and permits alterations in contents of Register A.
OP	Displays state of key system flip-flops.
P/Y	Displays and permits alterations in contents of Registers P and Y. (Registers P and Y contain the same address when the computer is in the memory access (MA) mode. Register P contains an address which is one higher than that contained in Register Y when the computer is in the single instruction (SI) or RUN modes.)
M	Displays and permits alterations in contents of Register M.
MSTR CLEAR	Clears Registers A, P, Y, and M; sets all timing to that state existing following a HLT (halt); initializes standard peripheral devices and options.
START	Starts machine in all modes. Indicator is illuminated when machine is in RUN mode.

Table 2-3.
Function of Control Panel Selector Switches

Switch	Position	Function
PFI/PFH (Power Fail Interrupt/ Power Fail Halt)	PFI	Allows the machine to cause a program interrupt when power fails.
	PFH	Allows the machine to halt when power fails.

Table 2-3. (Cont)
Function of Control Panel Selector Switches

Switch	Position	Function
STORE/FETCH	STORE	Enables data to be written into memory when the computer is in the MA mode.
	FETCH	Enables reading the contents of Registers A, Y, and M.
P/P+1	P	Enables data to be read out of memory when the computer is in the MA mode.
	P+1	Enables accessing a specific memory location when the computer is in the MA mode.
MA/SI/RUN (Memory Address/ Single Instruction/ Run)	MA	Enables accessing consecutive memory locations when the computer is in the MA mode.
	SI	Enables data to be written into or read from a location in memory and disables the protection for locations $1\text{-}17_8$.
	RUN	Enables normal operating mode.

Data within memory and several of the main frame registers can be monitored on the control panel. Data can also be manually entered into memory and several registers from the control panel. Procedures for initializing the system, reading out and entering data into memory and the main frame registers are provided in the following paragraphs.

SYSTEM INITIALIZATION

The system can be initialized by depressing the CLEAR pushbutton. This operation has no effect on the contents of memory. The MASTER CLEAR function is also performed when power is applied to the system via the POWER-ON pushbutton.

REGISTER DISPLAY

The contents of Register A, OP, P/Y, or M can be displayed by depressing the appropriate REGISTER pushbutton. The contents of the selected register is displayed on the 16 pushbutton/indicators. Refer to Table 2-4 for the significance of the OP display.

Table 2-4.
OP Display

Bit	Significance	Bit	Significance
1	T1, timing level 1	9	PI, permit interrupt
2	T2, timing level 2	10	Unassigned
3	T3, timing level 3	11	ML, memory lockout (option)
4	T4, timing level 4	12	Unassigned
5	F, fetch cycle	13	Unassigned
6	I, indirect cycle	14	Unassigned
7	A, execute cycle	15	MP, memory parity error
8	Unassigned	16	P, I/O parity error

REGISTER LOAD

Data is entered into Register A, P/Y or M as follows.

a. Depress the appropriate REGISTER pushbutton.

b. Depress the CLEAR pushbutton.

c. Enter the desired data by depressing the appropriate pushbutton/indicators (1 through 16). Refer to the paragraph on Memory Load in this section for the procedure to enter data into memory.

MEMORY DISPLAY

The contents of any memory location is displayed as follows.

a. Set MA/SI/RUN switch to MA.

b. Set FETCH/STORE switch to FETCH.

c. Set P/P+1 switch to P.

d. Depress REGISTER-P/Y pushbutton.

e. Depress CLEAR pushbutton.

f. Depress the appropriate pushbutton/indicators (1 through 16) to designate the octal address of the memory location containing the data to be displayed.

g. Depress REGISTER-M pushbutton.

h. Depress START pushbutton. The contents of the addressed memory location will be displayed on the pushbutton/indicators (1 through 16).

The contents of successive memory locations can be displayed as follows.

i. Set P/P+1 switch to P+1.

j. Depress the START pushbutton. Each time the START pushbutton is depressed the contents of the next memory location is displayed.

MEMORY LOAD

Data is entered into any memory location as follows.

a. Set MA/SI/RUN switch to MA.

b. Set FETCH/STORE switch to STORE.

c. Set P/P+1 switch to P.

d. Depress REGISTER-P/Y pushbutton.

e. Depress CLEAR pushbutton.

f. Depress the appropriate pushbutton/indicators (1 through 16) to designate the octal address of the memory location to be loaded.

g. Depress REGISTER-M pushbutton.

h. Depress CLEAR pushbutton.

i. Depress the appropriate pushbutton/indicators (1 through 16) to enter the desired data into the addressed location.

j. Depress START pushbutton.

The desired data is now in the addressed location. Successive memory locations can be loaded as follows.

k. Set the P/P+1 switch to P+1.

l. Repeat steps h through j for each successive memory location to be loaded.

SINGLE INSTRUCTION OPERATION

A program is executed in the single-instruction mode as follows.

a. Set MA/SI/RUN switch to SI. (When the switch is not in the MA position, the FETCH/STORE and P/P+1 switches are disabled.)

b. Depress MASTER CLEAR pushbutton.

c. Enter initial parameters into Register A, or P/Y as required. (Refer to the Register Load procedure described in this section.)

d. Depress START pushbutton.

The first instruction is fetched from memory and placed in Register M and may be examined by depressing the REGISTER-M pushbutton. Thereafter, each time the START pushbutton is depressed, the previously fetched instruction is executed, the next instruction is fetched, and the computer halts. If the P/Y pushbutton is depressed, the address from which the new instruction was fetched is displayed on the pushbutton/indicators (1 through 16). During execution (run operation) the SI position may be used at any time to aid in program debugging.

RUN OPERATION

A program is executed in the run mode as follows.

a. Set MA/SI/RUN switch to RUN.

b. Depress MASTER CLEAR pushbutton.

c. Enter initial parameters into Register A, or P/Y as required. (Refer to the procedure for Register Load described in this section.)

d. Depress START pushbutton.

The program will run until a HALT is executed or until the MA/SI/RUN switch is set to SI.

KEY-IN LOADER

The octal instructions listed below normally occupy memory locations 1_8 through 17_8 and enable the loading of "self-loading" paper tapes via the teletype or high-speed paper tape reader.

Octal Address	Octal Instruction	Meaning	Octal Address	Octal Instruction	Meaning
1	010057	STA '57	11	002010	JMP *-1
2	03000X	OCP '000X	12	041470	LGL 8
3	13100X	INA '100X	13	13000X	INA '000X
4	002003	JMP *-1	14	002013	JMP *-1
5	101040	SNZ	15	110000	STA* 0
6	002003	JMP *-3	16	024000	IRS 0
7	010000	STA 0	17	100040	SZE
10	13100X	INA '100X			

The value of "X" in the above instructions is dependent upon the input device used to read the paper tape. A "1" specifies the high-speed paper tape reader and "4" specifies the teletype.

The hardware protects memory locations 1_8 through 17_8 from modification by a stored program. Therefore, under normal conditions, the key-in loader should remain intact in these locations. However, when operating in the MA mode, locations 1_8 through 17_8 are unprotected, therefore care must be taken to avoid inadvertently destroying the key-in loader while loading memory. The key-in loader is entered into memory as follows.

a. Depress MASTER CLEAR pushbutton.

b. Set MA/SI/RUN switch to MA. (This unlocks the protected area, addresses $1\text{-}17_8$.)

c. Set STORE/FETCH switch to STORE.

d. Set P/P+1 switch to P+1.

e. Enter the first instruction (010057_8) into Register M.

f. Depress START pushbutton. The program counter (Register P) will be incremented by one and the first instruction will be entered into location 000001_8.

g. Repeat steps f. and g. for each of the remaining instructions to be loaded. Each time the START pushbutton is depressed, Register P will be incremented by one and the instruction in Register M will be loaded into memory.

CHECKING THE KEY-IN LOADER

The following instructions are performed to ensure that the key-in loader has been loaded correctly into the designated memory locations.

a. Depress MASTER CLEAR pushbutton.

b. Set MA/SI/RUN switch to MA.

c. Set STORE/FETCH switch to FETCH.

d. Set P/P+1 switch to P+1.

e. Depress Register-M pushbutton.

f. Depress START pushbutton. Register P will be incremented by one and the contents of memory location 000001_8 (010057) will be displayed on the pushbutton/indicators (1 through 16).

g. Repeat step f. for each of the remaining memory locations to be monitored. Each time the START pushbutton is depressed, Register P will be incremented by one and the contents of the addressed memory will be displayed.

SECTION III
SOFTWARE PACKAGE

This section consists of a table that lists utility, mathematical, and input/output routines in the DDP-416 software package. The mnemonic, document number, and equipment required are given for each routine listed. All routines are supplied in DAP-object format.

Table 3-1.
DDP-416 Software Package

Type and Function	Mnemonic	Doc. No.	Equipment Required*
UTILITY			
Assembly System			
Assembler	DAP-16	180275000	
I/O Selectors			
ASR only	IOS-16A	180560000	
Expanded I/O	IOS-16B	180324000	High-speed paper tape reader
Debug Package (one sector)	DEBUG	180430000	
Loaders			
Expanded			
ASR Input	LDR-A	180335000	
High-Speed Paper Tape Input	LDR-P	180336000	High-speed paper tape reader
Standard			
ASR Input	SLDR-A	180341000	
High-Speed Paper Tape Input	SLDR-P	180342000	High-speed paper tape reader
Object Program Dump and Load	PAL-AP	180311000	High-speed paper tape reader and punch (optional)
MATHEMATICAL			
Single Precision Arithmetic			
Add	ADD	180422000	
Subtract	SUB	180422000	
Multiply	MPY	180423000	
Variable Length Multiply	VMP	180423000	
Divide	DIV	180424000	
Remainder Entry	DIVR	180424000	

*Equipment required is basic unless otherwise specified.

Table 3-1. (Cont)
DDP-416 Software Package

Type and Function	Mnemonic	Doc. No.	Equipment Required*
MATHEMATICAL (Cont)			
Single Precision Standard Functions:			
Square Root	SQRX1	180425000	
Sine	SINX1	180426000	
Cosine	COSX1	180426000	
Arctangent	ATNX1	180427000	
Logarithm Base 2	LG2X1	180428000	
Logarithm Base e	LGEX1	180428000	
Logarithm Base 10	LG10X1	180428000	
Exponential Base e	EXEX1	180429000	
Exponential Base 2	EX2X1	180429000	
Exponential Base 10	EX10X1	180429000	
Double Precision Arithmetic Package:		180453000	
Add	DADD		
Subtract	DSUB		
Multiply	DMPY		
Divide	DDIV		
Twos Complement	DTCA		
Load Simulated Double Accumulator	DLDA		
Store Simulated Double Accumulator	DSTA		
Skip on Overflow	DSOV		
Skip on No Overflow	DSNO		
INPUT/OUTPUT			
Interrupt Executive Routine	X$EXEC	180472000	
Dummy Executive Routine		180472000	
Decode I/O Status Word	STAT	180529000	
Interrupt Linkage Setup	XILOC	180528000	
ASR-33/35			
Reader			
ASCII	I$AAS	180527000	
Binary	I$ABS	180526000	

*Equipment required is basic unless otherwise specified.

Table 3-1. (Cont)
DDP-416 Software Package

Type and Function	Mnemonic	Doc. No.	Equipment Required*
INPUT/OUTPUT (Cont)			
Punch			
ASCII	O$AAS	180530000	
Binary	O$ABS	180531000	
End of Message	O$ASS	180530000 and 180531000	
Leader	O$ALRS		
Teleprinter			
Listing	O$LLS	180551000	
Heading	O$LHS		
Page Eject	O$LES		
High-Speed Paper Tape			
Reader			
ASCII	I$PAS	180533000	High-Speed Paper Tape Reader
Binary	I$PBS	180534000	
Punch			
ASCII	O$PAS	180539000	High-Speed Paper Tape Punch
Binary	O$PBS	180532000	
End of Message	O$PSS	180539000 and 180532000	
Leader	O$PLRS		
Listing	O$PLS		
Heading	O$PHS	180547000	
Page Eject	O$PES		
Card Reader, Hollerith	I$CAS	180537000	Card Reader
Magnetic Tape			
Read			Magnetic Tape Transport and Control Unit
BCD (2 characters/word)	I$MA1S	180548000	
Binary			
(2 characters/word)	I$MB1S		
(3 characters/word)	I$MC1S		
Write			
BCD (2 characters/word)	O$MA1S	180552000	
Binary			
(2 characters/word)	O$MB1S		
(3 characters/word)	O$MC1S		
End of File	O$ME1S		

*Equipment required is basic unless otherwise specified.

Table 3-1. (Cont)
DDP-416 Software Package

Type and Function	Mnemonic	Doc. No.	Equipment Required*
INPUT/OUTPUT (Cont)			
Conversion			
ASCII to IBM Tape Code	C$8T06	180536000	Magnetic Tape Transport and Control Unit
IBM Tape Code to ASCII	C$6T08	180535000	
Control			
Rewind	C$MR1S		
Forwardspace			
One Record	C$FR1S		
One File	C$FF1S	180549000	
Backspace			
One Record	C$BR1S		
One File	C$BF1S		
Physical and Logical Magnetic Tape Correlation	M$UNIT	180538000	

*Equipment required is basic unless otherwise specified.

SECTION IV
OPERATING PROCEDURES

This section presents detailed procedures for loading and assembling programs and for punching self-loading paper tapes.

LOAD

The following paragraphs contain procedures for loading both self-loading and object programs on paper tape.

Load Self-Loading Programs

a. Depress the MASTER CLEAR pushbutton.

b. Load 1_8 into Register P.

c. Insert the DAP self-loading tape into the appropriate input device.

d. Depress the START pushbutton. (When loading with an ASR-33 teletype unit, the manual START switch on the device must be activated. When loading with an ASR-35 teletype unit, the MODE switch must be set to KT.) The program will be loaded into the locations it occupied when it was punched out. Loading any self-loading tape destroys the contents of locations 00020_8 through 00057_8 inclusive.

Load Object Programs

a. Perform the procedure for self-loading programs, described above, using the appropriate self-loading loader program.

b. Load $XX000_8$, the starting location of the loader, into Register P (XX is the highest sector of memory). (If the object program is relocatable (REL), perform steps c. and d. If the object program is absolute (ABS), skip steps c. and d., and continue at step e.)

c. Load Register A with the octal address of the location at which loading is to begin. If Register A is clear, a starting location of 1000_8 will be assumed.) To force a starting location of 00000_8, load 100000_8 into Register A.

d. Press START. The computer will halt.

e. Load Register A with the octal address of the location at which the cross-sector indirect address word table is to begin. (If Register A is clear, a starting location of 100_8 will be assumed.)

NOTE

The cross-sector indirect word table must be located in sector 0 for DDP-416 computers without the Memory Lockout Option if the expanded loaded LDR is used, or when using the standard loader SLDR. The table may be in any sector for computers with the option. The sector may be changed at load time by the pseudo-operation SETB.

f. Insert the object program into the appropriate input device.

g. Depress the START pushbutton. The object program will be loaded into memory until a halt occurs and a loader message is produced on the teletype. Refer to Table 4-1 for the significance of the loader messages.

Table 4-1.
Loader Messages

Message	Meaning	Action Required
LC	Loading complete	Depress the START pushbutton to begin program execution.
MR	More subroutines required	Insert the required subroutine tapes into the appropriate input device and depress the START pushbutton to continue loading.
CK	Checksum error in the last block read	Depress the START pushbutton to ignore the block and continue loading.
BL	Block too large or improperly formatted	Depress the START pushbutton to ignore the block and continue loading.
MO	Memory overflow due to program attempting to overwrite the loader	Depress the START pushbutton to obtain a memory map. No recovery is possible from this error.

The loader cannot detect a program overlaying the indirect word tables or the tables overlaying the program. You must obtain a memory map to determine whether overlap exists. The loader will detect base sector overflow and produce a MO message.

Options Following a Loader Halt

When a loader halt occurs and a message is produced, there are several options you can perform other than those specified in Table 4-1. The options allow you to perform the following functions.

a. Reload the object tape by repeating the procedure "Loading Object Programs" described above.

b. Recover from a missing end-of-tape block. The procedure is as follows.

(1) Momentarily set the MS/SI/RUN switch to SI to stop the computer.

(2) Load $XX001_8$ into Register P.

(3) Depress the START pushbutton.

c. Print a memory map. The procedure is as follows.

(1) Load $XX002_8$ into Register P.

(2) Depress the START pushbutton. A memory map will be produced on the teletype.

d. Set a program break. The procedure is as follows.

(1) Load $XX003_8$ into Register P.

(2) Load Register A with the address of the location at which loading is to continue. If Register A is cleared, the origin for loading remains unchanged.

(3) Depress the START pushbutton and loading will continue.

e. Force load a subprogram. The procedure is as follows.

(1) Load $XX004_8$ into Register P.

(2) Insert a tape into the appropriate input device if required.

(3) Depress the START pushbutton. The tape will be loaded into memory.

f. Begin executing the object program. The procedure is as follows.

(1) Load $XX005_8$ into Register P.

(2) Depress the START pushbutton and the program will be executed.

ASSEMBLE DAP-16 SOURCE PROGRAMS

This procedure enables you to perform an assembly, using a self-loading DAP-16 system tape. To perform the assembly, proceed as follows.

a. Depress the MASTER CLEAR pushbutton.

b. Load 1_8 into Register P.

c. Insert the DAP-16 system tape into the appropriate input device.

d. Depress the START pushbutton and the system tape will be loaded into memory.

e. Select a bit pattern from Table 4-2 to designate a one or two pass assembly and to designate the I/O devices to be used during the assembly.

f. Load the selected bit pattern into Register A.

g. Load 400_8 into Register P. There are five standard starting options available for the DAP-16 assembler.

(1) 400_8 - Start assembly

(2) 401_8 - Continue assembly

(3) 402_8 - Start subroutine tape assembly

(4) 403_8 - Terminate assembly

(5) 404_8 - Restart pass two to produce additional object tapes

h. Insert a DAP-16 source tape into the appropriate input device and turn on your input/output equipment.

NOTE

If you are using the high-speed paper tape equipment or an ASR-35 to assemble, perform step i to complete the procedure. If you are using the ASR-33 to assemble, skip step i and perform steps j through n.

i. Depress the START pushbutton and the assembly will be executed.

j. Depress the START pushbutton. A portion of the source tape will be read and a halt will occur.

k. Depress the ON pushbutton on the ASR-33 paper tape punch.

l. Depress the START pushbutton. A length of object tape will be punched and a halt will occur.

m. Depress the OFF pushbutton on the ASR-33 paper tape punch.

n. Repeat steps j through m until the complete source tape has been read and a complete object tape has been punched.

After the assembly is completed, you will have an object tape and a listing of your program (if both were requested). When operating in the two-pass assembly mode, the output is generated during the second pass. The second pass is accomplished by repeating steps h. and i. after the source tape has been read in the first time. If the assembly was successfully completed, Register A will contain all ONE's when the computer halts. If a MOR pseudo-operation was encountered, Register A will contain all ZEROs.

The DAP-16 assembler indicates coding errors by typing or printing error flags in the left-hand margin of the listing. (See Table 8-1.) Such errors do not interfere with the assembly process. Undefined symbols are automatically defined by the assembler and are listed at the end of each pass.

Table 4-2.
Assembly and I/O Device Selection

Set Bit	Assembly or I/O Device Selected
1	Two pass assembly. If bit 1 is not set, a one pass assembly is executed.
	Input Device:
2	Teletype Unit
3	High-Speed Paper Tape Reader
4	Card Reader
5	Magnetic Tape Unit
6	Teletype Unit with halts for manual inputs
	Output Device:
7	Teletype Unit
8	High-Speed Paper-Tape Punch
9	Card Punch
10	Magnetic Tape Unit
11	No Object Output

Table 4-2. (Cont)
Assembly and I/O Device Selection

Set Bit	Assembly or I/O Device Selected
	Listing Device:
12	Teletype Unit
13	High-Speed Paper-Tape Punch
14	Magnetic Tape Unit
15	Line Printer
16	No Listing

NOTE

If any of the five-bit I/O groups are all ZEROs, a standard device, depending on the configuration, will be selected.

PUNCH SELF-LOADING OBJECT TAPES

A program is provided in your software package which enables you to punch self-loading object tapes of any segment of memory. This punch and load program (PAL-AP) is supplied in self-loading form. The following procedures describe the method for loading the program and executing the program once it is loaded.

Load the Self-Loading PAL-AP

a. Depress the MASTER CLEAR pushbutton.

b. Load 00001_8 into Register P.

c. Depress the START pushbutton. The tape will be loaded into the highest 1200_8 locations of memory. (The program overlays the relocating loader if it is currently in the high sectors of memory and locations 00020_8 through 00057_8 are destroyed.)

Execute PAL-AP

a. Depress the MASTER CLEAR pushbutton.

b. Load ZZ100 into Register P. (ZZ is the sector into which PAL-AP was loaded.)

c. Load into Register A the starting address of the data to be punched out. (If the teletype is being used, set the high-order bit of Register A and turn on the punch unit on the teletype.)

d. Depress the START pushbutton. The program will halt at location $ZZ117_8$.

e. Load into Register A the ending address of the data to be punched out.

f. Depress the START pushbutton. A self-loading object tape will be punched out.

SECTION V
UTILITY LIBRARY

This section contains short descriptions of the routines in the DDP-416 Utility Library. The purpose, storage requirements, and usage are given for each routine. In some cases more detailed information contained either in the program listing or other 3C documents is referenced.

DAP ASSEMBLER (DAP-16)

Purpose

The DAP Assembler is designed to convert DAP-coded source language programs to object form for execution on the DDP-116, 416, or 516. DAP-16 is a one- or two-pass assembler; additional passes, however, may be made to produce extra copies of the program in object form. The program is loaded with the standard bootstrap loader. There are five starting options available.

400 Start assembly
401 Continue assembly (for restart after a reading check, etc.)
402 Start subroutine tape assembly (do not punch EOF at end of tape)
403 Terminate assembly (dump buffers, punch trailer, punch EOF)
404 Restart pass 2 (to produce additional tapes)

The contents of Register A control the number of passes and device selection. The sign bit (bit 1) must be set to establish a two-pass mode. If the 4K I/O Selector (IOS-16A) is used, the device selection is limited by the system configuration. If the 8K I/O Selector (IOS-16B) is used, the remaining 15 bits of Register A are used to determine device selection as follows.

a. Source File Selection

Bit 2 - ASR-33/35 input
Bit 3 - High-speed paper tape reader input
Bit 4 - Card reader input
Bit 5 - Magnetic tape input
Bit 6 - Halts for manual control of ASR-33/35 paper tape punch will occur

b. Object File Selection

Bit 7 - ASR-33/35 output
Bit 8 - High-speed paper tape punch output

Bit 9 - Card punch output

Bit 10 - Magnetic tape output

Bit 11 - No object output

c. List File Selection

Bit 12 - ASR-33/35 listing

Bit 13 - High-speed paper tape punch listing

Bit 14 - Magnetic tape listing

Bit 15 - Line printer listing

Bit 16 - No listing

If all bits of one of the 5-bit groups are ZERO, a standard device, depending on system configuration, will be selected.

Storage Requirements

A minimum of 4K using IOS-16A (preselected I/O) or 8K using IOS-16B (selectable I/O).

Usage

Refer to the DAP-16 manual for complete operating procedures and details on usage.

INPUT/OUTPUT SUPERVISOR (IOS-16B)

Purpose

IOS-16B provides the necessary supervision and control of DAP-16 assembler input/output device selection. Source, object, and listing devices are selected by presetting Register A as indicated in the preceding DAP-16 description. IOS-16B has 15 entries that are used by the DAP-16 Assembler, as required. Each entry returns control to the assembler when its function has been accomplished. The 15 entries, and the purpose of each, are as follows.

D$IN	To interrogate Register A and preset source, object, and listing library subroutine. Calls for the requested device.
D$OL	To output a line of data on a listing device.
D$BH	To output a heading line on a listing device.
D$RD	To read a record from input device to core buffer and compute record sequence number.
D$ADV	To advance the character pointer to the next output field.
D$LG	To place an error flag in the first four positions of the output buffer for the given line of output.
D$OB	To process binary output for object device and to output the buffer if it is full or if the new block type is not the same as the previous block type.

D$OPT	To set octal machine op-code to ASCII code and store in buffer.
D$CHR	To read a character from either the left or right half of a word.
D$SGN	To insert a minus sign in column 17 of the output buffer.
D$STR	To insert double asterisk in columns 23 and 24 of the output buffer.
D$EJ	To eject page for listing device.
D$CKSZ	To fetch buffer counter.
PAUSE	To halt if not magnetic tape input. Return on restart.
HALT	When using magnetic tapes, wrap up on output pass. If input pass, reset magnetic tape input, halt. Return on restart.

Storage Requirements

IOS-16B requires 766_{10} (1376_8) locations (Rev. A).

EXPANDED LOADERS (LDR-A, LDR-P)

Purpose

To load either absolute or relocatable main programs and subprograms produced by the DAP-16 assembler into the DDP-416 memory. LDR-A requires only the ASR-33/35 for input, and LDR-P requires the high-speed paper tape reader. The binary output of the DAP-16 assembler includes a 15-bit or 16-bit address for each memory reference so that progams may be written for computers with the extended memory option. Nine-bit address instructions are desectored by the loader before execution whether in extended-memory mode or not. It is the programmer's responsibility to ensure that code is loaded in the mode in which it will be executed. This should be done prior to assembly by use of the pseudo-ops EXD (enter extend-mode desectoring) and LXD (leave extend-mode desectoring).

Storage Requirements

LDR-A and LDR-P each require approximately 3-1/2 sectors of memory.

Usage

Refer to Section IV of this manual for instructions on using the loaders.

STANDARD LOADERS (SLDR-A, SLDR-P)

Purpose

SLDR-A and SLDR-P load either absolute or relocatable main programs and subprograms produced by the DAP-16 assembler into the DDP-416 memory. The standard loaders are essentially only limited versions of the expanded loaders (LDR-A and LDR-P). The standard loaders do not allow for extended memory addressing, relocatable sector zero, or forward references. SLDR-A requires the ASR-33/35 for input, and SLDR-P requires the high-speed paper tape reader.

Storage Requirements

SLDR-A and SLDR-P each require more than two sectors of memory.

Usage

Refer to Section IV of this manual for instructions on using the loaders.

PUNCH AND LOADER PROGRAM (PAL3)

Purpose

PAL3 punches self-loading object tapes for execution on the DDP-416 computer. This routine is made up of a punch section and a load section which is of the bootstrap variety. PAL3 punches out its own loader in 8-8 format followed by 12 in. of leader. The desired program is then punched in PAL format, which is recognized by the loader. The self-loading ability of a PAL program is apparent: the loader on the front of the tape will first load itself, and then the PAL-format program. Data is punched in blocks of 50 words each with six frames of leader between blocks. Refer to Section IX of this manual for details on how data is punched on paper tape.

Storage Requirements

PAL3 requires less than one sector of core.

Usage

Refer to Section IV of this manual for complete instructions on preparing PAL-format tapes.

DDP-416 DEBUG PACKAGE

Purpose

Aids in debugging DDP-416 programs by allowing the following functions.

a. Entering values into memory locations
b. Providing breakpoints
c. Typing a given location each time a breakpoint is executed
d. Typing memory between two limits each time a breakpoint is executed
e. Scanning memory between given limits to locate an octal value
f. Clearing one or more locations in memory by inserting zeros.

This program will operate on any standard DDP-416 computer, but it must not be loaded into sector zero.

Storage Requirements

Debug requires 506_{10} (772_8) locations (Rev. A). Location 777 is reserved exclusively for use by this package.

Usage

The program starts at load point +1. The operations defined in Table 5-1 may be performed after the program has started and then pauses to allow input from the ASR-33/35 keyboard. All numeric inputs will be interpreted as octal. The first input character must be a valid command code. Any non-octal input serves to separate octal fields. A carriage return terminates input. During typed input, a slash (/) enables restart of input.

NOTE

LOC = memory location
OV = octal value
(CR) = carriage return

Table 5-1.
Debug Operations

Code	Operation
A (CR)	Type the value in (A)
A OV (CR)	Set the value in (A) to OV
B (CR)	Clear all breakpoints
B LOC3 (CR)	Set a breakpoint at LOC3
B LOC3, LOC1 (CR)	Set a breakpoint at LOC3 and type the contents of LOC1 each time LOC3 is executed

Table 5-1. (Cont)
Debug Operations

Code	Operation
B LOC3, LOC1, LOC2 (CR)	Set a breakpoint at LOC3 and type the contents of each location between LOC1 and LOC2 inclusive each time the breakpoint is executed.
	NOTE
	A maximum of four breakpoints may be specified.
C (CR)	Continue in test program from breakpoint.
C OV (CR)	Continue in test program from breakpoint OV times.
D LOC1 (CR)	Dump value at LOC1.
(CR)	LOC1 = LOC1 +1. Dump value at LOC1.
OV (CR)	OV replaces value at LOC1. LOC1 = LOC1 +1. Dump value at LOC1.
D LOC1, LOC2 (CR)	Dump values between LOC1 and LOC2 inclusively.
	NOTE
	(CR) and OV (CR) are valid only when the previous command begins with a "D".
M LOC1, LOC2, OV1, OV2 (CR)	Scan memory between LOC1 and LOC2 inclusively for word OV1 using mask OV2. Type matching location and its contents.
S LOC1 (CR)	Start program at LOC1.
S (CR)	Start program at location specified in the last S LOC1 (CR) operation.
Z LOC1 (CR)	Replace value at LOC1 with zeros.
Z LOC1, LOC2 (CR)	Replace values between LOC1 and LOC2 with zeros.

Program output on the ASR-33/35 typewriter resulting from valid operations includes the following.

a. A dollar sign ($) that terminates valid operations and indicates that further input may be entered.

b. A question mark (?) that follows invalid input and indicates that further input may be entered.

Errors

Some of the more common errors that may occur while debugging are as follows.

a. Undefined operation

b. Use of commands in illogical sequence (e.g., an initial C command when no breakpoint has been executed.

c. Memory locations out of range.

d. LOC1 greater than LOC2.

SECTION VI
MATHEMATICAL LIBRARY

All mathematical routines in the DDP-416 Library are listed in Table 6-1. Each routine is listed categorically according to the function that it performs. Information given for each routine includes the main routine identification, calling sequence, errors, accuracy, timing, storage, and other routines used. The routine identification in column 2 is not necessarily the entry for the function in column 1, but rather the identification of the routine that contains it.

Single-precision routines require that the first argument (ARG1) be loaded into Register A prior to entry.

Double precision routines require the loading of the simulated double precision accumulator with the first argument (ARG1) prior to calling the routine. This is accomplished by calling the double precision load routine DLDA. After calling the desired double precision arithmetic routine, the double precision results may subsequently be stored by calling the double precision store routine DSTA.

All routines operate in a real, fixed-point mode.

Table 6-1.
Mathematical Library

Function	Routine Identification	Calling Sequence	Errors	Accuracy (Bits)	Timing (μsec)	Storage (Decimal)	Other Routines Used
SINGLE PRECISION Arithmetic Routines							
Add	ADD	CALL ADD, DAC ARG2	Overflow	16	25.92	40	None
Subtract	ADD	CALL SUB, DAC ARG2	Overflow	16	32.64	40	None
Multiply	MPY	CALL MPY, DAC ARG2	None	15	292.22	82	None
Multiply, Variable Length	MPY	CALL VMPY, DEC L, DAC ARG2	None	variable	Note 3	82	None
Divide	DIV	CALL DIV, DAC ARG2	ARG1 ≥ ARG2; ARG = - 1 BO	15	336.96	75	None
Divide, Remainder Entry	DIV	CALL DIVR	None	15	3.84	75	None
Standard Functions							
Arctangent	ATNX1	CALL ATNX1	None	15	1825.32	31	MPY
Cosine	SINX1	CALL COSX1	None	15	1502.38	55	MPY
Exponential (Base e)	EXEX1	CALL EXEX1	None	15	1333.24	92	MPY, DIV
Exponential (Base 2)	EXEX1	CALL EX2X1	None	15	1633.20	92	MPY, DIV
Exponential (Base 10)	EXEX1	CALL EX10X1	None	15	1931.12	92	MPY, DIV
Logarithm (Base e)	LGEX1	CALL LGEX1	ARG1 < .5 BO, ARG1 ≥ 1 BO	15	1566.32	61	MPY, DIV
Logarithm (Base 2)	LGEX1	CALL LG2X1	ARG1 < .5 BO, ARG1 ≥ 1 BO	15	1263.54	61	MPY, DIV
Logarithm (Base 10)	LGEX1	CALL LG10X1	ARG1 < .5 BO, ARG1 ≥ 1 BO	15	1566.32	61	MPY, DIV
Sine	SINX1	CALL SINX1	None	15	1495.66	55	MPY
Square Root	SQRX1	CALL SQRX1	None	15	1031.42	64	MPY, DIV

Table 6-1. (Cont)
Mathematical Library

Function	Routine Identification	Calling Sequence	Errors	Accuracy (Bits)	Timing (μsec)	Storage (Decimal)	Other Routines Used
DOUBLE PRECISION Arithmetic Routines							
Add	DPAP	CALL DADD, DAC ARG2	Overflow	30	69.60		
Subtract	DPAP	CALL DSUB, DAC ARG2	Overflow	30	74.88		
Multiply	DPAP	CALL MPY, DAC ARG2	Overflow	30	1180.00		DTCA, DLDA
Divide	DPAP	CALL DDIV, DAC ARG2	Divisor ≤ Dividend	30	1445.76		DTCA, DLDA
Twos Complement	DPAP	CALL DTCA	None	30	16.32 (min) 21.12 (max)	325	
Load Simulated Double Accumulator	DPAP	CALL DLDA, DAC ARG2	None	30	46.08 (min)		
Store Simulated Double Accumulator	DPAP	CALL DSTA, DAC ARG2	None	30	46.08 (min)		
Skip on Overflow	DPAP	CALL DSOV	Overflow	30	11.52 (min) 14.40 (max)		
Skip on No Overflow	DPAP	CALL DSNO	Overflow	30	11.52 (min) 14.40 (max)		

Notes:
1. Unless otherwise specified, timing given is average and accuracy is plus or minus a rounding factor.
2. ARG1 and ARG2 are real, fixed-point, fractional binary numbers.
3. $50.00 + 11.904\ L + 6.72\ M + 2.88\ N$

   where:

   L = fixed width (including sign)
   M = number of ones (not including sign)
   N = number of zeros (not including sign)

SECTION VII
INPUT/OUTPUT LIBRARY

All routines in the DDP-416 Input/Output Library are described in this section. The purpose, storage requirements, and calling sequence are given for each routine. Errors are also given where applicable. Standard I/O routines are intended to operate under control of an I/O Interrupt Executive Routine (X$EXEC) to which control is returned if an I/O operation cannot be honored. Each routine, however, can also be used in a non-interrupt mode by using the Dummy I/O Executive Routine.

When using I/O routines, the programmer should first call the system initialization entry to the executive routine. No task should be considered complete until all I/O operations requested have been executed.

Mnemonic conventions used for most I/O Library routines consist of five characters as follows.

a. The first character is either an I (for input) or an O (for output).

b. The second character (dollar sign) identifies the routine as a library routine.

c. The third character designates the device as follows.

A	ASR-33/35	P	Paper Tape Reader or Punch
C	Card Reader	M	Magnetic Tape Transport

d. The fourth character specifies the mode or function as follows.

A	ASCII	S	End of Message
B	Binary	F	File
E	Eject (or End of File)	I	Initialize
		L	Listing
H	Heading	P	Punch

e. The fifth character indicates that the routine operates under control of the I/O Interrupt Executive Routine.

The magnetic tape control routines and several special-purpose I/O routines are an exception to the general rule for I/O routine mnemonics.

Refer to the individual program listing for each of the I/O routines for more detailed descriptions and method used. Document numbers for the program listing for each of the I/O routines may be found in Section III of this manual. Storage requirements listed in this section are subject to change. Refer to the applicable program listing for up-to-date information.

I/O INTERRUPT EXECUTIVE ROUTINE (X$EXEC)

Purpose

X$EXEC monitors I/O interrupts, determines which I/O program is to receive control when an interrupt occurs, and transfers control to the proper program. The X$EXEC entry is used only when an interrupt occurs and is never used under program control. The purpose of other entries to the executive routine is as follows.

X$ENB enables interrupt (after an I/O routine is finished) and returns control to the calling program.

X$EXCI initializes the executive routine. This entry must be used prior to use of any I/O routines.

X$P1-17 sets up the program register table entry for the appropriate level. This table is used by the executive routine to determine the location to which control should be transferred.

X$ST1 allows external access to ST1 (active I/O routine status).

X$ST2 allows external access to ST2 (enabled I/O device status).

Storage Requirements

The executive routine requires 88_{10} (130_8) locations.

Calling Sequence

X$ENB: CALL X$ENB
(Normal return)

X$EXCI: CALL X$EXCI
(Normal return)

X$P1-17: CALL X$PN (N= level of program setting the register table)

DUMMY I/O EXECUTIVE ROUTINE

Purpose

This routine is a dummy executive used to allow operation of I/O library routines in a non-interrupt mode. The initialization entry (XEXCI) clears the two status words (ST1 and ST2). All other entries do nothing but immediately return control to the calling program.

Storage Requirements

This routine requires 10_{10} (12_8) locations.

Calling Sequence

The dummy entries are called in the same manner as entries to the real executive routine.

DECODE I/O STATUS WORD (STAT)

Purpose

STAT routine decodes the I/O status word generated by some I/O routines. Status words can indicate busy, end-of-file, end-of-tape, record unreadable, and number of words. The significance of the first four status-word bits is as follows.

Bit 1: Busy flag
Bit 2: End-of-file (EOF) flag, or end-of-message if paper tape
Bit 3: End-of-tape (magnetic tape only)
Bit 4: Record unreadable (magnetic tape only)

The magnetic tape read routines keep count of the number of characters read in the low-order eight bits of the status word. This routine leaves the low-order eight bits of the status word unchanged in the A-register. The high-order eight bits are set to ZERO.

Storage Requirements

STAT requires 22_{10} (26_8) locations.

Calling Sequence

```
LDA     Status word
CALL    STAT
JMP     BUSY
JMP     EOF
JMP     EOT
JMP     Record unreadable
Normal returns (no status bits on)
```

INTERRUPT LINKAGE SETUP (XILOC)

Purpose

XILOC provides a memory location for linkage with the I/O interrupt executive routine. XILOC can also be used in defining other parameters that may vary with the system. A constant within this routine can be referenced with an external address constant (XAC) pseudo-operation. This provides the capability of changing externally referenced data without reassembly of the using program.

Storage Requirements

XILOC requires one location.

ASR PAPER TAPE READER, ASCII (IAAS, IAIS)

Purpose

I$AAS reads ASCII coded paper tape using the ASR-33/35 Paper Tape Reader. The I$AIS entry is used to initialize the read routine (I$AAS). The read routines will assume, if not initialized by I$AIS, that the input buffer is 40 words long and that there are three tab settings corresponding to character positions 6, 12, and 30 (DAP-16 source format). If I$AIS has been used for non-standard initialization, and subsequently it is desired to use the standard tab setup, the routine must be reinitialized accordingly.

Storage Requirements

I$AAS and I$AIS require 200_{10} (310_8) locations.

Calling Sequence

```
Initialization -  CALL      I$AIS
                  Busy return
                  DEC       (number of words in input buffer)
                  DEC       (number of tabs in following table)
                  DEC       T1 (character position of first tab)
                   :         :
                  DEC       Tn (character position of nth tab)
                  (Normal return)

Read Data      -  CALL      I$AAS
                  Busy return
                  Status word
                  DAC       (address of input buffer)
                  (Normal return)
```

Errors

The user may provide a routine having the entry C$ASRS to perform any action he wishes, otherwise a default entry (NO OP) is used and no action is taken.

Reading of an end-of-message character causes the routine to terminate with an end-of-file status (040000 octal).

ASR PAPER TAPE READER, BINARY (IABS, IABIS)

Purpose

I$ABS reads binary records from paper tape using the ASR-33/35 Paper Tape Reader. The I$ABIS entry is used to initialize the tape-read routine (I$ABS) by clearing the start-of-message flag before proceeding to the I$ABS code.

Storage Requirements

This routine requires 191_{10} (277_8) locations.

Calling Sequence

```
Initialization  -  CALL      I$ABIS
                   Busy return
                   Status word
                   DAC       (address of input data buffer)
                   (Normal return)

Read Data       -  CALL      I$ABS
                   Busy return
                   Status word
                   DAC       (address of input data buffer)
                   (Normal return)
```

Errors

Reading an X-OFF character followed by an end-of-message character causes the end-of-file status to be stored in the calling sequence status word (040000 octal). Reading more than 60 data words causes a halt with the A-register set to all ONEs. Pressing the start button causes program execution to proceed.

ASR PAPER TAPE PUNCH, ASCII (OAAS, OAIS, OASS, OALRS)

Purpose

O$AAS punches ASCII coded paper tape using the ASR-33/35 Paper Tape Punch. The purpose of each entry is as follows.

O$AIS initializes the punch routines for other than normal operations.
O$AAS punches paper tape in the ASCII mode.
O$ASS punches an end-of-message code.
O$ALRS passes ten inches of leader.

The punch routine O$AAS will assume, if not initialized by O$AIS, that the data buffer is 40 words long and that there are three tab settings corresponding to character positions 6, 12, and 30. If O$AIS has been used for non-standard initialization, and subsequently it is desired to use the standard tab setup, the routine must be reinitialized accordingly.

Storage Requirements

This routine requires 218_{10} (332_8) locations.

Calling Sequence

```
Initialization -  CALL     O$AI
                  Busy return
                  DEC      (number of words in output buffer)
                  DEC      (number of tabs in the following table)
                  DEC      T1 (character position of first tab)
                   :        :
                  DEC      Tn (character position of nth tab)

Punch Data     -  CALL     O$AAS
                  Busy return
                  Status word
                  DAC      (address of output data buffer)
                  (Normal return)

Punch EOM      -  CALL     O$ASS
                  Busy return
                  Status word
                  (Normal return)

Punch Leader   -  CALL     O$ALRS
                  Busy return
                  Status word
                  (Normal return)
```

ASR PAPER TAPE PUNCH, BINARY (OABS, OASS, O$ALRS)

Purpose

O$ABS punches binary-coded paper tape using the ASR-33/35 Paper Tape Punch. The purpose of each entry is as follows.

O$ABS punches paper tape in the binary mode
O$ASS punches an end-of-message code
O$ALRS passes ten inches of leader

Storage Requirements

This routine requires 199_{10} (307_8) locations.

Calling Sequence

```
Punch Data -  CALL     O$ABS
              Busy return
              Status word
              DAC      (Address of the output data buffer. First word in
                        buffer indicates number of words to be punched.)
              (Normal return)
```

Punch End-of-Message	-	CALL	O$ASS
		Busy return	
		Status word	
		(Normal return)	
Punch Leader	-	CALL	O$ALRS
		Busy return	
		Status word	
		(Normal return)	

ASR TELEPRINTER OUTPUT (OLLS, OLHS, O$LES)

Purpose

These routines print a heading line, a data line, or eject the page on the ASR teleprinter unit depending on which entry is used as follows.

O$LLS prints a thirty-five word data line
O$LHS prints a thirty-word header and page number
O$LES ejects page

Storage Requirements

These routines require 280_{10} (430_8) locations.

Calling Sequence

Punch Data Line or Header	-	CALL	O$LLS (or O$LHS)
		Busy return	
		Status word	
		DAC	(starting address of buffer)
		(Normal return)	
Page Eject	-	CALL	O$LES
		Busy return	
		Status word	
		(Normal return)	

HIGH-SPEED PAPER TAPE READER, ASCII (IPAS, IPIS)

Purpose

I$PAS reads ASCII coded paper tape using the high-speed paper tape reader. The I$PIS entry is used to initialize the read routine (I$PAS). The read routine will assume, if not initialized by I$PIS, that the input buffer is 40 words long and that there are three tab settings corresponding to character positions 6, 12, and 30. If I$PIS has been used for non-standard initialization, and subsequently it is desired to use the standard tab setup, the routine must be reinitialized accordingly.

Storage Requirements

This routine requires 191_{10} (277_8) locations.

Calling Sequence

```
Initialization -  CALL      I$PIS
                  Busy return
                  DEC       (number of words in input buffer)
                  DEC       (number of tabs in following table)
                  DEC       T1 (character position of first tab)
                   :         :
                  DEC       Tn (character position of nth tab)
                  (Normal return)

Read Data      -  CALL      I$PAS
                  Busy return
                  Status word
                  DAC       (address of input data buffer)
                  (Normal return)
```

Errors

Reading an end-of-message character causes the routine to return with the status word containing an end-of-file status (040000 octal).

HIGH-SPEED PAPER TAPE READER, BINARY (IPBS, IPBIS)

Purpose

I$PBS reads binary records from paper tape using the high-speed paper tape reader. The I$PBIS entry is used to initialize the read routine (I$PBS), and to read the first record on the tape.

Storage Requirements

This routine requires 172_{10} (254_8) locations.

Calling Sequence

```
Initialization -  CALL      I$PBIS
                  Busy return
                  Status word
                  DAC       (address of input data buffer)
                  (Normal return)
```

Read Data - CALL I$PBS
Busy return
Status word
DAC (address of input data buffer)
(Normal return)

Errors

Reading an X-OFF character followed by an end-of-message character causes the end-of-file status to be stored in the calling sequence status word (040000 octal). Reading more than 60 data words causes a halt with the A-register set to all ONEs. Pressing the start button causes program execution to proceed.

HIGH-SPEED PAPER TAPE PUNCH, ASCII (OPAS, OPIS, OPSS, OPLRS)

Purpose

These routines are designed to punch paper tape in ASCII format using the high-speed paper tape punch.

O$PIS initializes the punch routines for other than normal operation
O$PAS punches paper tape in the ASCII mode
O$PSS punches end-of-message code
O$PLRS passes ten inches of leader

The punch routine will assume, if not initialized by O$PIS, that the output buffer is 40 words long and has three tab settings corresponding to character positions 6, 12, and 30. If O$PIS has been used for non-standard initialization, and subsequently it is desired to use the standard tab setup, the routine must be reinitialized accordingly.

Storage Requirements

This routine requires 217_{10} (331_8) locations.

Calling Sequence

Initialization - CALL O$PIS
Busy return
DEC (number of words in output buffer)
DEC (number of tabs in following table)
DEC T_1 (character position of first tab)
⋮ ⋮
DEC T_n (character position of nth tab)

Punch Data - CALL O$PAS
Busy return
Status word
DAC (address of output data buffer)
(Normal return)

Punch End-of-Message - CALL O$PSS
Busy return
Status word
(Normal return)

Punch Leader - CALL O$PLRS
Busy return
Status word
(Normal return)

HIGH-SPEED PAPER TAPE PUNCH, BINARY (OPBS, OPSS, O$PLRS)

Purpose

O$PBS punches binary-coded paper tape using the high-speed paper tape punch. The purpose of each entry is as follows.

O$PBS punches paper tape in the binary mode
O$PSS punches an end-of-message code
O$PLRS passes ten inches of leader

Storage Requirements

This routine requires 197_{10} (305_8) locations.

Calling Sequence

Punch Data - CALL O$PBS
Busy return
Status word
DAC (Address of output data buffer. First word in buffer contains number of words to punch.)
(Normal return)

Punch End-of-Message - CALL O$PSS
Busy return
Status word
(Normal return)

Punch Leader - CALL O$PLRS
Busy return
Status word
(Normal return)

HIGH-SPEED PAPER TAPE PUNCH, HEADING, LISTING, EJECT (OPHS, OPLS, O$PES)

Purpose

This routine punches a heading, listing, or page eject code on paper tape using the high-speed paper tape punch. The purpose of each entry is as follows.

O$PHS punches a fifty-five word header
O$PLS punches a sixty-word data line
O$PES punches a page eject code

Storage Requirements

This routine requires 307_{10} (463_8) locations.

Calling Sequence

Punch Header - or Listing	CALL	O$PHS (or O$PLS)
	Busy return	
	Status word	
	DAC	(starting address of buffer)
	(Normal return)	
Punch Eject Code -	CALL	O$PES
	Busy return	
	Status word	
	(Normal return)	

CARD READER, HOLLERITH (ICAS, ICAIS)

Purpose

I$CAS reads one card in the Hollerith mode, converts the 6-bit Hollerith code for each character to 8-bit ASCII code, and packs the data two-characters per word. The I$CAIS entry is used to initialize the card read routine (I$CAS) for other than an 80-column read.

Storage Requirements

This routine requires 136_{10} (208_8) locations.

Calling Sequence

Initialization -	CALL	I$CAIS
	DEC	(number of columns to be read)
	(Normal return)	

```
Read Data   -   CALL      I$CAS
                Busy return
                Status word
                DAC       (address of the data buffer into which the
                            characters are to be read)
                (Normal return)
```

MAGNETIC TAPE READ ROUTINES (I$MA1S, I$MB1S, I$MC1S)

Purpose

These routines are designed to read magnetic tape in any of three available modes as follows.

I$MA1S: BCD mode, two characters per word

I$MB1S: Binary mode, two characters per word

I$MC1S: Binary mode, three characters per word

In either two-character-per-word mode, 6-bit tape recording characters are read in pairs and stored in bit positions 1 through 12. In the three-character-per-word mode, the first two characters are stored in bit positions 1 through 12 and the least significant four bits (channels 3-6) are stored in bit positions 13 through 16.

Storage Requirements

These routines require 151_{10} (227_8) locations.

Calling Sequence

```
Read Data   -   CALL      I$MA1S, I$MB1S, or I$MC1S
                Busy return
                Status word
                DAC       (address of input buffer)
                DEC       (number of words in input buffer)
                DEC       (logical tape unit)
                (Normal return)
```

Errors

The record-unreadable status (010000 octal) is indicated if on 10 consecutive attempts to read a parity error is indicated. The data read on the tenth attempt is stored in the data buffer and the tape is positioned to read the next record. The end-of-tape status (020000 octal) is indicated if the reflective marker near the end of the tape has been reached while reading a record. Data is unaffected and is stored in the buffer.

The end-of-file status (040000 octal) is indicated if an end-of-file indicator is detected instead of a data record.

MAGNETIC TAPE WRITE ROUTINES (O$MA1S, O$MB1S, O$MC1S)

Purpose

These routines are designed to write magnetic tape in any of three available modes or to write end-of-file depending on which entry is used as follows.

O$MA1S:	BCD mode, two characters per word
O$MB1S:	Binary mode, two characters per word
O$MC1S:	Binary mode, three characters per word
O$ME1S:	End-of-file

In either two-character-per-word mode, 6-bit tape recording characters are taken in pairs from bit positions 1 through 12 of the computer word and written in two consecutive frames on tape. In the three-character-per-word mode, characters are taken in groups of three from bit positions 1 through 6, 7 through 12, and 13 through 16. The first two characters are recorded in two consecutive frames and are followed by a frame containing the last 4 bits of the computer word recorded in channel positions 3 through 6.

Storage Requirements

This routine requires 172_{10} (254_8) locations.

Calling Sequence

Write Data -

```
CALL      O$MA1S, O$MB1S, or O$MC1S
Busy return
Status word
DAC       (address of output buffer)
DEC       (number of words in output buffer)
DEC       (logical tape unit)
(Normal return)
```

End-of-File -

```
CALL      O$ME1S
Busy return
Status word
DEC       (logical tape unit)
(Normal return)
```

Errors

The end-of-tape status (020000 octal) is indicated if the reflective marker near the end of the tape has been reached while writing a record. Writing is unaffected by the presence of the end-of-tape marker.

CONVERT IBM MAGNETIC TAPE CODE TO ASCII (C$6T08)

Purpose

C$6T08 is designed to convert IBM 6-character magnetic tape code to equivalent 8-character ASCII code.

Storage Requirements

C$6T08 requires 81_{10} (121_8) locations.

Calling Sequence

```
CALL        C$6T08
DAC         (address of data buffer containing characters to be
              converted)
DEC         (number of words in data buffer)
(Normal return)
```

CONVERT ASCII TO IBM MAGNETIC TAPE CODE (C$8T06)

Purpose

C$8T06 converts 8-character ASCII code to the equivalent IBM 6-character magnetic tape code.

Storage Requirements

C$8T06 requires 80_{10} (120_8) locations.

Calling Sequence

```
CALL        C$8T06
DEC         (address of data buffer containing characters to be
              converted)
DEC         (number of words in data buffer)
(Normal return)
```

MAGNETIC TAPE CONTROL PACKAGE (C$MR1S, C$FR1S, C$BR1S, C$FF1S, C$BF1S)

Purpose

This routine provides any of a number of magnetic tape control functions depending on the entry used as follows.

C$MR1S:	Rewind magnetic tape
C$FR1S:	Forwardspace magnetic tape one complete record
C$BR1S:	Backspace magnetic tape one complete record
C$FF1S:	Forwardspace magnetic tape one complete file
C$BF1S:	Backspace magnetic tape one complete file

Storage Requirements

This routine requires 181_{10} (265_8) locations.

Calling Sequence

```
CALL          C$XX1S (where XX = MR, FR, BR, FF, or BF)
Busy return
Status word
DEC           Logical unit number
(Normal return)
```

Errors

If a file mark is encountered when spacing forward one record (C$FR1S), an end-of-file will be indicated.

PHYSICAL AND LOGICAL MAGNETIC TAPE UNIT CORRELATION (M$UNIT)

Purpose

M$UNIT converts logical tape unit numbers to physical device numbers by direct table lookup and stores the result in the A-register upon exit.

Storage Requirements

This routine requires 15_{10} (17_8) locations.

Calling Sequence

```
LDA           (location of logical tape unit number)
CALL          M$UNIT
(Normal return)
```

SECTION VIII
ERROR MESSAGES

LOADING MESSAGES

A message is always typed when loading halts to indicate the reason for the halt. See Table 8-1 for an explanation of these messages.

DAP-16 ASSEMBLY PROGRAM

DAP-16 is able to detect many types of clerical errors commonly made in coding programs. These errors are indicated by an appropriate error code printed in the left margin of the assembly listing. Examples of errors that are detected are shown in Table 8-1. Errors in a field will generally result in that field being assembled as a zero. In the case of multiply defined symbols, the first symbol definition is used.

Table 8-1.
Error Messages Generated by the DAP-16
Assembly Program

Error Message	Condition
M	Multiply defined symbol
C	Erroneous conversion of a constant or a variable field in improper format
A	Address field missing where normally required, or error in address format
O	Operation code missing or in error
L	Location symbol missing where required, or error in location symbol
S	Address of variable field expression not in sector being processed or sector zero (applicable only in load mode)
R	Relocation assignment error
X	Symbol table or literal table exceeded
F	Major formatting error
V	Unclassified error in variable field of multiple field pseudo-operator (i.e., DEC, OCT, etc.)
T	Improper use of or error in index field
N	Missing name (main program or subroutine)

SECTION IX
PAPER TAPE FORMATS

This section describes the format of paper tapes that are used as a principal input/output medium for the DDP-416 computer. Data is recorded on paper tape by groups of holes arranged in a definite format along the length of the tape. Paper tape is a continuous recording medium, as opposed to cards which are fixed in length, and the length of data records is limited only by the input/output requirements of the system. A vertical column of holes extending across the tape is referred to as a frame. A horizontal row of holes extending the length of the tape is referred to as a channel. For paper tapes punched and read by the DDP-416 system, there are eight channel-hole positions per frame, and one small sprocket hole. (See Figure 9-1.)

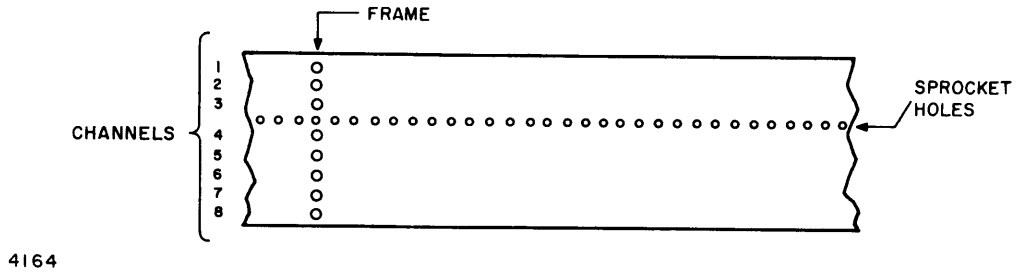


Figure 9-1. General Format of Paper Tape

The format descriptions given in this section apply to tapes punched by the high-speed paper tape punch as well as those punched by the ASR-33/35 Paper Tape Punch. Paper tape formats used with DDP-416 systems fall into two main categories: an ASCII format (used for punching source code), and a 4/6/6 format (used for punching object code).

ASCII FORMAT

ASCII format is an octal code that uses eight channels to define one character per frame. Each frame is read from channel 8 to channel 1 in bit groups of 2-3-3 as illustrated in Figure 9-2. Two ASR-33/35 typewriter control codes, '212 for line feed and '215 for carriage return, are represented in Figure 9-2 to illustrate use of the ASCII format.

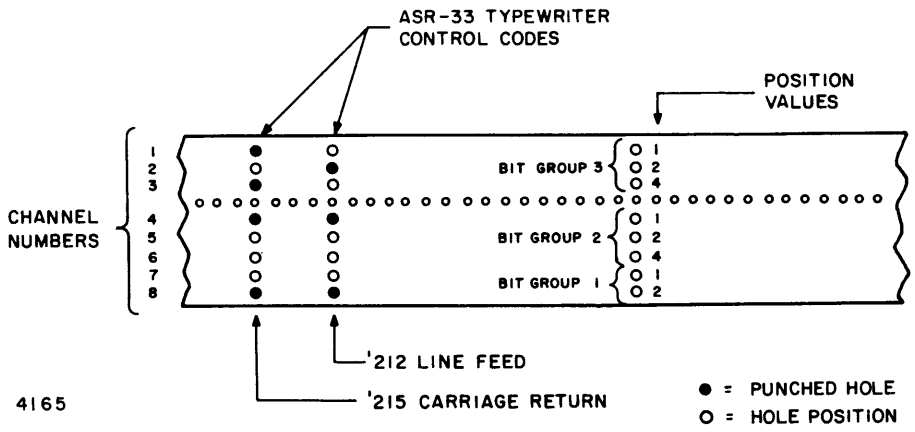


Figure 9-2. ASCII Format

SOURCE TAPE PREPARATION

A DAP-16 source-program data line is recorded on paper tape in ASCII format as follows.

.. LINE FEED... TEXT ... X-OFF ... CARRIAGE RETURN ...

The text string between the LINE FEED at the beginning of the line and the X-OFF at the end of the line is read into the input buffer. The LINE FEED, X-OFF, and CARRIAGE RETURN are control characters and are not input as part of the text string. The X-OFF at the end of the line (preceding the CARRIAGE RETURN) is necessary to be compatible with the ASR-33/35 input routines. (The ASR-35 also requires a RUBOUT following the CARRIAGE RETURN.) If the tapes to be read by the paper tape read subroutine are never to be read by the ASR-33/35 tape read routine, the X-OFF (and RUBOUT) may be omitted.

When preparing a source tape using the ASR-33, depress the LINE FEED key, type the desired ASCII record (maximum of 72 characters), and then depress the X-OFF and CARRIAGE RETURN keys. Repeat this process for each record. If a character is punched erroneously, depress the backspace and RUBOUT keys and proceed with the rest of the line. If a line is punched erroneously, depress the Left Arrow, X-OFF, and CARRIAGE RETURN keys. Source tapes punched using the ASR-35 are prepared in a similar manner except that the maximum number of characters per record is 75, and the RUBOUT key must be punched after the RETURN key.

Tabs may be used to compress the data much as tabs are used on a typewriter. The backslash character (\) '334 is used as a tab code. The backslash is punched on the ASR-33/35 as an upper-case L (FORM). Tabs may be used whenever a string of spaces precedes a "tab stop." The tab is punched in place of the spaces. Another way to describe the backslash is that it is used as a field delimiter. For example, the backslash is ordinarily used when the location, operation, or variable field is not present.

The ASCII paper tape read subroutines will assume, if not initialized, that there are three tab positions corresponding to character positions 6, 12, and 30 (DAP-16 source code format).

The END OF MESSAGE (EOM) record has the following format.

... X-OFF ... EOM (EOM is '203 and is punched by using the CONTROL KEY with the letter C).

The paper tape read subroutine will read one line of data per entry. If an END OF MESSAGE code ('203) is encountered at any point in the line, the END OF MESSAGE return will be taken, otherwise, the normal return is taken when a carriage return is encountered. The END OF MESSAGE code is usually used as a unit record following the last data line to be read.

If the data line exceeds the length of the data buffer, those characters in excess will be ignored. No error indication will be given. Similarly, if a tab is encountered after the last tab-stop has been passed, the tab will be treated as a space. No error indication will be given. The data line may be shorter than the length of the data buffer in which case, the buffer is filled out with spaces.

Figure 9-3 shows a portion of an actual assembly listing (DAP-Test Program) along with one line of corresponding source code punched on paper tape.

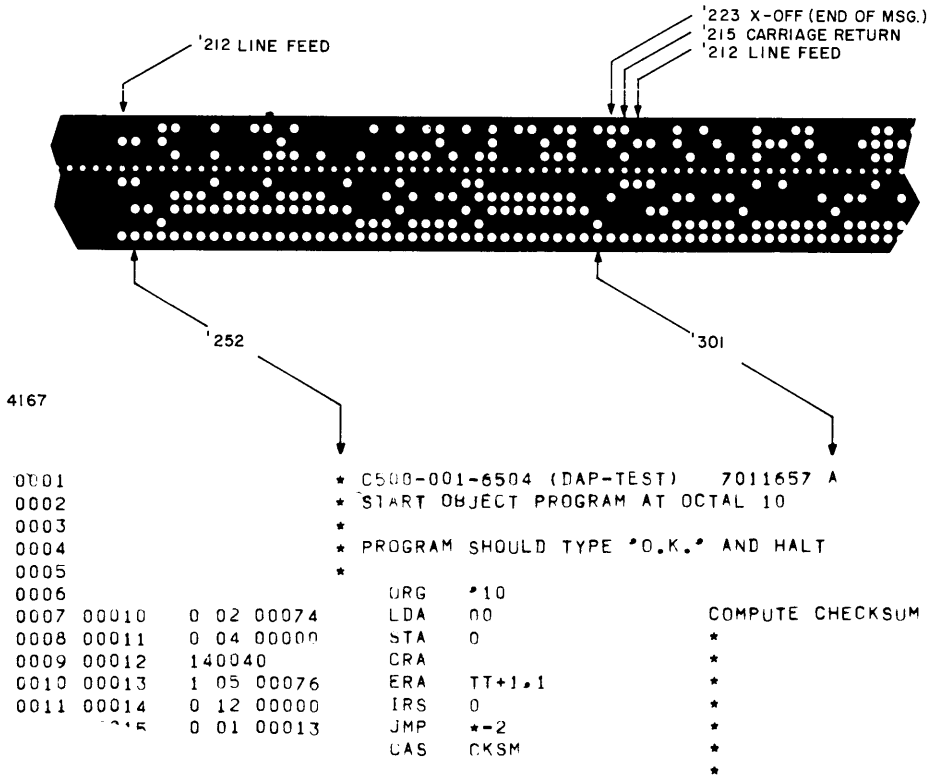


Figure 9-3. Source Code Punched in ASCII Format

4/6/6 FORMAT

The 4/6/6 format is a binary code that uses three frames to define one 16-bit word. As illustrated in Figure 9-4, bits 1-4 of a 16-bit word are encoded to make up the first frame, 5-10 the second frame, and 11-16 the third frame. Control codes that are punched in ASCII format precede and follow each block of 16-bit words encoded in 4/6/6 format and use only one frame each. Each data block begins with an SOM ('201) and ends with a DC4 (or X-OFF '223).

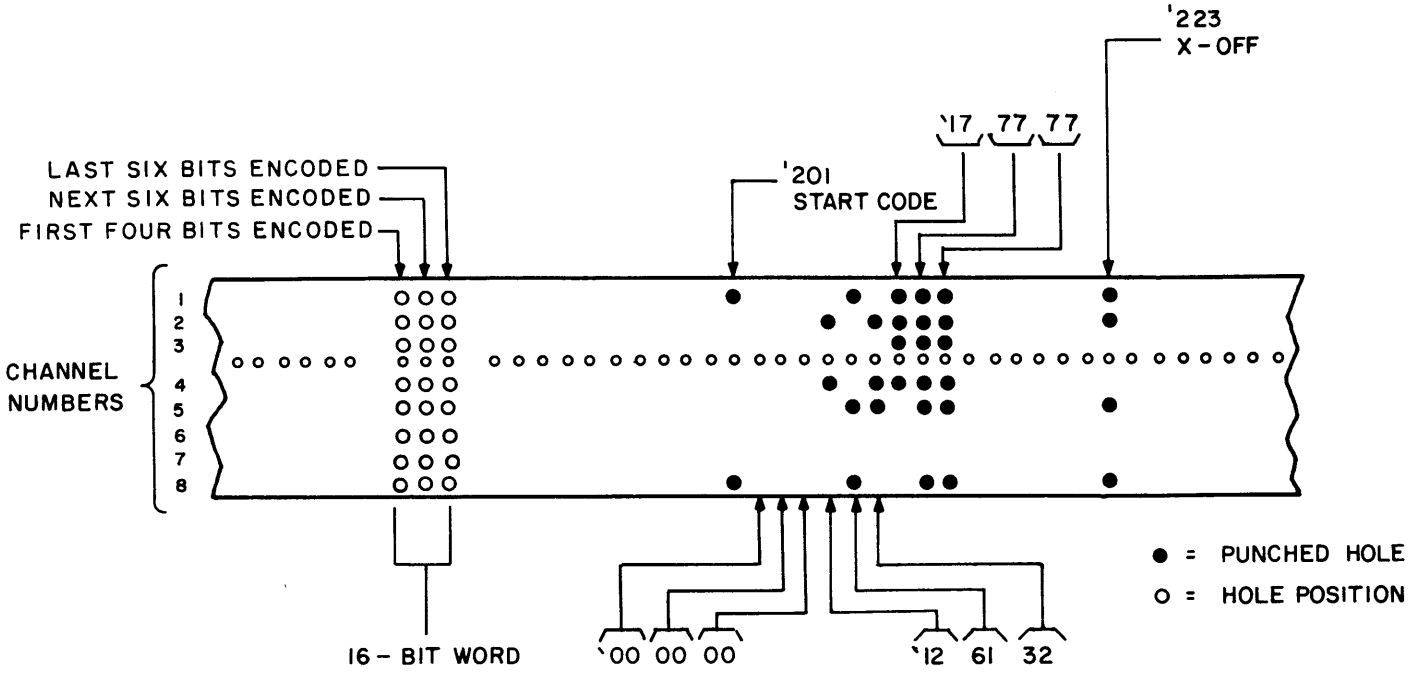


Figure 9-4. 4/6/6 Format

When a frame is punched, channels 6 and 7 are ordinarily left blank. This is done so that a frame will not print when read by the ASR. However, certain six-bit patterns correspond to control functions, which if executed by the ASR, would interfere with further reading of this tape. Therefore, these (eight) six-bit groups are translated according to Table 9-1 before being punched. A tape punched using this translation will not type anything when read in by the ASR. Hence, the 4/6/6 format is sometimes called the "Invisible Format."

Table 9-1.
4/6/6 Translations

Intended 4 or 6 Bits	Is Converted to Frame	Possible Ambiguity with ASR-33/35 Control
05 or 45	174 or 374, respectively	WRU
12 or 52	175 or 375	LF
21 or 62	176 or 376	X-ON
23 or 63	177 or 377	X-OFF

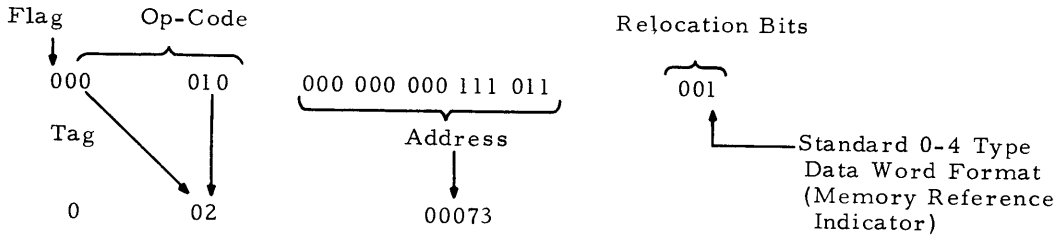
Object tapes that are produced by the DAP assembly process are punched in the binary 4/6/6 format. Within each block of object code, there is a variable-length sequence of data words. There are a number of different types of blocks, and they are defined in Section IV of the DAP-16 Manual, 3C Doc. No. 130071629. Object tapes generated in the 4/6/6 format are accepted by and compatible with both the DDP-416 standard and expanded loaders.

Figure 9-5 shows a portion of an actual assembly listing (DAP-TEST Program) along with corresponding object code punched in paper tape. As explained in the DAP Manual, the first 4/6/6 group after the start code defines the block type. This particular block type is 0-4 (000400), or a data block. The next 4/6/6 group specifies the number of data words in the data block. In this case there are 72_8 words in the data block. The words are interpreted as per the format for this block type given in the DAP Manual. In block type 0-4 the remaining bits are grouped into groups of 24 bits each. These groups are formed by using 16 bits of one 4/6/6 group and the first 8 bits of a second 4/6/6 group. The remaining 8 bits of the second 4/6/6 group and 16 bits of a third 4/6/6 group are used to form the second 24-bit group.

In the example given in Figure 9-5, the first 24-bit group is

000010000000000111011001

The DAP-16 Manual (3C Doc. No. 130071629) further explains the meaning of each 24-bit group, based on the last three bits. This one converts to the LDA instruction at address 00010 as follows.



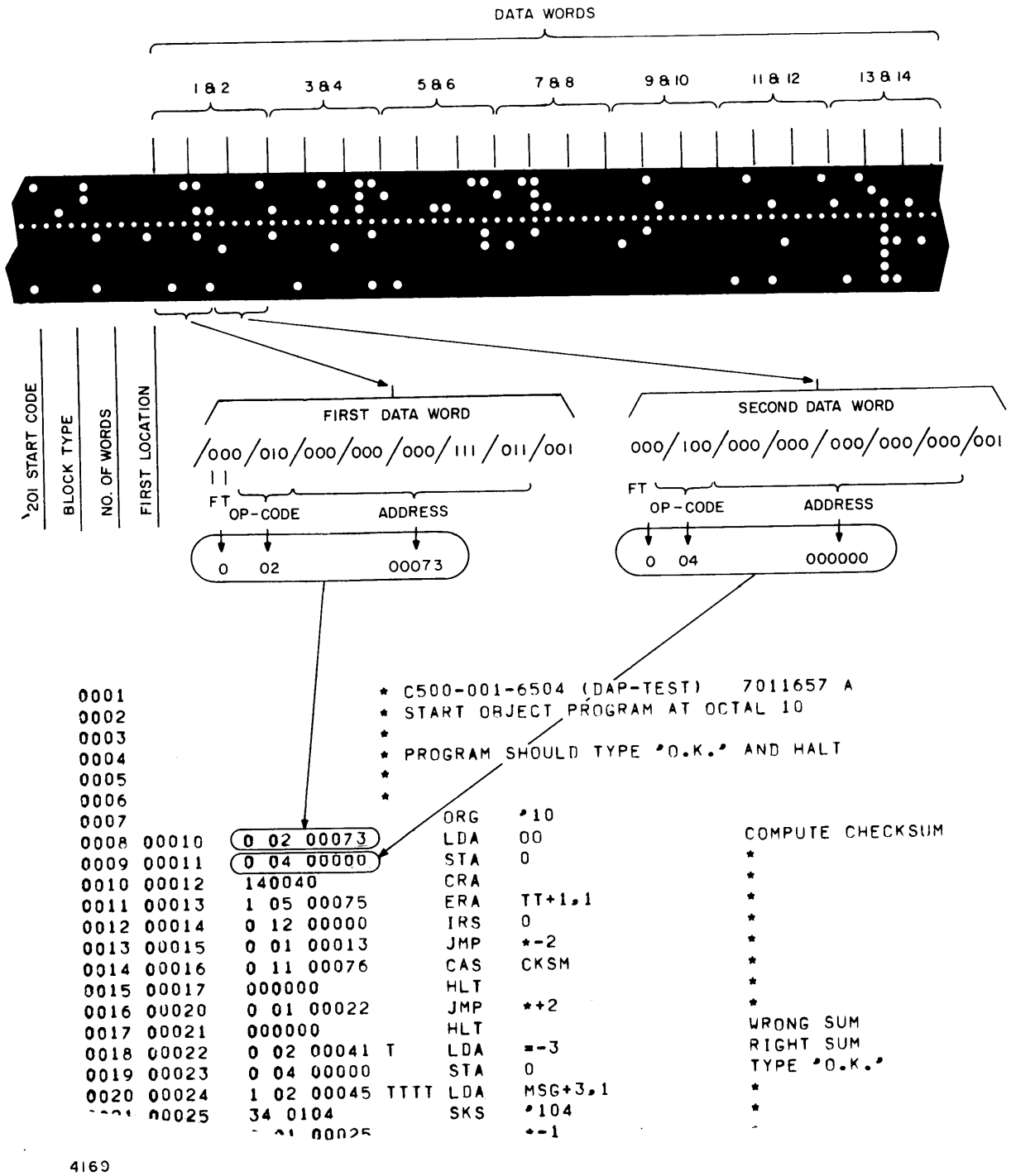


Figure 9-5. Object Code Punched in 4/6/6 (Invisible) Format

HONEYWELL, COMPUTER CONTROL DIVISION, FRAMINGHAM, MASS. 01701