COGAR
SYSTEM 4®

PROGRAMMER'S REFERENCE MANUAL

PROGRAMMER'S REFERENCE MANUAL

This publication is designed to be used as a reference manual by programmers using the Cogar System 4® Processor. The manual is divided into three parts. Part I defines the unique features of the machine which are relative to the programmer, as well as providing a machine specification summary. Part II provides general information on the usage of each group of instructions in the instruction set repertoire. Part III defines each instruction in detail, and provides the timing and an example of how each instruction may be used in context with surrounding instructions, in both Source and Object coding. A summation of all the instructions in the repertoire is contained on the Cogar System 4 Instruction Reference Card.

Other publications relating to software for the Cogar System 4 are:

Batch Assembler Operating Instructions; which contains the step-by-step instructions for creating a self-loading program tape, which has been assembled as part of an Object-String background.

Standard Cogar Library Functions; which contains descriptions and operating instructions for the Language Base Library and the I/O Libraries.

The programmer should be familiar with the content and design objectives of the above documents in order to make full use of the capabilities of the Cogar System 4 Processor.

PROGRAMMER'S REFERENCE MANUAL

Table of Contents

COGAR INSTRUCTION SET INDEX

COGAR INSTRUCTION SET INDEX

SPECIFICATION SUMMARY

Size	10 inches high (25 cm) 18.5 inches wide (47 cm) 24 inches deep (60 cm)
Weight	60 pounds (27 kg)
Power	115 VAC ±10%, 220 VAC ±10% 48 to 62 Hz 2.5 amps average
Environment	10% to 80% relative humidity without condensation 60°F to 95°F Operating Temperature 0°F to 150°F Storage Temperature
Ventilation	30 cubic feet per minute air flow 4 inches air flow clearance on all sides 1000 BTU per hour heat dissipation
Processor	45 instruction types plus I/O 3 to 6 μs instruction cycle time 1 Accumulator 7 Index Registers per 2K of memory 16 Member Instruction Address Stack Hardware Bootstrap Loader
Memory	16K bytes capacity Random Access Read/Write Non-Destructive Read-Out Monolithic Semiconductor
Keyboard	Software configurable Hall effect keys N-Key rollover capability Audible cue
Visual Display	5 inch CRT 4 or 8 line display, with interleave capability 32 characters per line 5 x 8 matrix under program control
Tape System	10 ips write tape speed 1600 bpi density, phase modulation 2 mechanically independent transports Read after Write, CRC, phase checks Automatic threading Write interlock switch Rewind: 40 ips rewind and forward or rewind search
Tape Cartridges	100 ft. computer grade tape 900 records of 136 characters each Write/Erase Protection

SECTION I. GENERAL

1. SYSTEM FEATURES

The Cogar System 4 is a compact, operator-oriented, general purpose data processing system. It combines, in a single unit, an input keyboard, magnetic tape transports, CRT visual display, I/O interface, solid state memory and a versatile processor. The System architecture closely integrates the functioning of all sub-systems and features transparency of graphics and coding. All major system functions are under program control. The processor structure is designed to optimize byte handling and interpretation, and provides automatic threading of recursive subroutines.

The nature of the processor design and its relationship to the other system components make the Cogar 4 heavily dependent on software. This means that the system is uniquely flexible in the jobs it can perform and is especially adaptable for various operator and interfact applications. It also means that software is an essential ingredient that must be as fully and carefully integrated into the System as the other components.

The Cogar 4 is a binary machine using 8-bit bytes in its memory organization and most hardware data paths. Its operations are highly memory oriented and are designed to take advantage of the performance of its semiconductor storage.

2. LANGUAGE FEATURES

The language base for the Cogar System 4 is flexible, easy to learn and use, yet permits the programmer to take full advantage of the System 4's power. The Cogar Language Base is comprised of a comprehensive set of "Pre-packaged" functions to facilitate modular program construction. The Cogar Assembler provides linkage between these functions and the specialized routines necessary to a given application.

Programs are written and assembled in symbolic notation, with the final stage of the assembly effecting a merge of the specialized routines and the pre-packaged background functions. This method of assembly allows easy and rapid modification or correction of programs or the re-configuration of a program to accomodate different peripheral devices or the selection of a new or modified graphic set, or keyboard configuration.

The DPL-1 instructions for the Cogar 4 are machine level instructions that are directly executed while the DPL-2 commands are executed interpretively by a resident software monitor. DPL-1 instructions are two bytes long and must occur on even byte boundaries. DPL-2 commands are four bytes long and should also occur on even boundaries. When DPL-1 and DPL-2 are intermixed, a new language is formed called DPL-3. The batch assembler for DPL-3 is known as DPL-3B. A subset of the DPL-2 monitor that handles I/O function is known as the I/O Supervisor or IOS. This manual describes DPL-1 and IOS as assembled on DPL-3B.

In order to be able to tailor the system for optimum use with particular applications, many device functions have been designed for program control. The codes generated by the keyboard, for example, correspond not to the key character, but to the key location. A translate table is located in the processor memory and is used to convert a key code into a character code. The user program can easily modify the translate table and can thus produce any desired code for any key.

The visual display uses a 5 x 8 dot matrix to form each display character and has cursor control with each character. The dot matrix is stored in the processor memory so that any possible 5 x 8 combination may be generated by the user program to be displayed for any character code. The standard dot pattern uses a 5 x 7 dot matrix to form the desired character. This provides for a space between the character and the cursor.

The Cogar 4 provides an unusually efficient subroutine control mechanism that is easy to use, yet offers powerful capabilities.

3. IOS FEATURES

Cogar has designed an Input/Output Supervisor to provide easy access for the user to a set of standard I/O routines. The flexibility of the system peripheral device operations is still available for special applications, but most I/O operations can be accommodated by the I/O Supervisor. IOS is a memory resident software monitor that is accessed using the ENT:IOS pseudo command. It performs a complete single operation and automatically returns control to the user program.

4. ASSEMBLER FEATURES

Computer programs must always eventually be expressed in machine language. The machine only understands binary numbers and programs so expressed are called Object programs. There are some circumstances when it is desirable for the system user to be able to write Object instructions directly. Most of the time, however, it is much more efficient to use an instruction language that is easily interpreted by the user. The mnemonic expressions used to represent the Object language form a Symbolic language. An Assembler is a program that translates a Symbolic program into an Object program.

Since the programmer spends much of his time communicating with the Assembler, it is useful to supply commands that control the operations of the Assembler itself. These commands are called Pseudo instructions and normally do not result in any Object coding. Another class of Pseudo instructions used in the Cogar 4 Assembler to control executive monitor operations does generate Object coding.

The Cogar Batch Assembler, known as DPL-3B, provides many features designed to streamline the programming process. Comments may be inserted in the Symbolic program to help identify the operations taking place. Instructions, data, constants and locations may all be referred to symbolically. Diagnostics are generated to help identify errors in the program. Editing, display and printing of both Object and Symbolic programs are available as part of the DPL-3B package.

The Cogar Assembler also handles the appropriate translations, controls, and linkages for the IOS and DPL-3 monitors.

5. DISPLAY

Keyboard Transparency:

The Cogar System 4 is designed to provide code hardware transparency. Any keyboard character may be automatically translated to any desired code and any dot matrix pattern may be displayed for a given character code. These functions are directly under software control and are thus available to the programmer.

Selective Blanking:

The commonly used internal key and character codes in standard Cogar software are shown in Table 1. Notice that the high order octal digit is always zero. This digit corresponds to the bits six and seven of the character byte. These two bits are used to provide added features for the CRT display. If a 1 is inserted in bit 7 (changing the code for A, for example, from 015 to 215) of a character in the CRT buffer area, that character will be displayed on the screen as a blank.

Cursor Underscore:

If a 1 is inserted in bit 6 (changing the code for A, for example, from 015 to 115) of a character in the CRT buffer area, that character may be displayed with an underline. The underline feature must be enabled by adding octal 1 to the second octal digit of the display base enable function codes. Thus, to permit underlines in display base 2 the normal display enable of IOC, C#3; 023 becomes IOC, C#4; 033. The underline feature is a convenient means of providing a cursor.

Selective Interlace:

Memory areas displayed are program selectable from any one of 16 memory Pages (256 bytes per Page), with provision for half Page (128 bytes) display only or for selective interlace of half-Pages.

6. KEYBOARD

When a character key is depressed on the keyboard after a Transfer Byte IOC, it causes a key code to be loaded into the accumulator. The NUM (numeric), CTRL (control) and ALPHA (alphabetic) are three special keys that act on bits 6 and 7 of the key code for any key pressed while one of them is held down. NUM turns on bit 6, CTRL turns on bit 7, and ALPHA turns on both 6 and 7. If none of the special keys are activated, bits 6 and 7 remain off. The following procedure may be used to translate the key code residing in the accumulator into a character code.

a. The 6th and 7th bits are taken care of as follows:

6th bit on:	do not change
7th bit on:	turn 7th bit off (reset after translation, if desired).
6th and 7th bits on:	turn 6th and 7th bits off (reset after translation, if desired).

b. Store the result in an index register
c. Add to the index register the displacement within the page of the beginning of the translate table. The standard translate table in page 05, for example, starts at location decimal 064, therefore, add decimal 064 to the value of the index register containing the key code before translation.
d. Load the Accumulator using indexed addressing and the page where the translate table resides. The Accumulator now contains the character code for the key that was depressed. The translate table may be designed by the user to supply any desired 8 bit character code including ASCII, EBCDIC, etc.

7. CARTRIDGE TAPES

The resident software I/O Supervisor provides for the actual reading, writing and tape positioning of the Mini-Tape. The user will often want to test the status of the tape drives for his own purposes. For example, to check the presence of a cartridge on a particular tape drive, first execute a Status instruction (IOC, C#N; Ø16), then test with a mask of Ø2Ø (TMJ, +NN; OCT:Ø2Ø). If the condition is satisfied, the cartridge is not present. Any of the status byte conditions may be tested by first loading the status of the device in question into the accumulator, and then testing it against the literal mask specified.

		00	01	02	03	04	05	06	07	10	11	12	13	14	15	16	17	20	21	22	23	24	25	26	27	30	31	32	33	34	35	36	37	40	41	42	43	44	45	46	47	50	51	52	53	54	55	56	57	60	61	62	63	64	65	66	67	70	71	72	73	74	75	76	77
			SCRATCH AREA															SEQUENCE STORAGE AREA								START BOOTSTRAP LOAD LABEL FIELD																																																					OCTAL NOTATION
PAGE 00	000		XR1	XR2	XR3	XR4	XR5	XR6	XR7	CRC	1	2	3	4	5	6	7																	SP0		SP1		SP2		SP3		SP4		SP5		SP6		SP7		SP8		SP9		SP10		SP11		SP12		SP13		SP14		SP15	
	100																																																																									
	200																					END BOOTSTRAP LOAD →																																																															
	300																																																																									
PAGE 01	000																																																																									
	100																																																																									
	200																																																																								
	300																																																																								
PAGE 02	000		* MAIN PROGRAM STARTS																																																																								
	100																																																																								
	200																																																																								
	300																																																																								
PAGE 03	000		* INTERRUPT																																																																							
	100																																																																								
	200																																																																							
	300																																																																							
PAGE 04	000	SP 000	– 010	+ 010	ø 076	1 000	2 162	3 042	4 030	5 047	6 074	7 141	8 066	9 106	A 174	B 177	C 076	D 177	E 177	F 177	G 076	H 177	I 000	J 060	K 177	L 177	M 177	N 177	O 177	P 177	Q 076	R 177	S 042	T 001	U 007	V 007	W 177	X 143	Y 007	Z 141	, 000	# 024	@ 014	– 010	% 143	$ 044	* 052	. 000	< 000	> 101	/ 040	(034	) 000	? 000	¢ 034	= 024	'' 000	! 000	' 000	: 000	; 000	— 100	¬ 002	& 060	↑ 000
	100	000	010	010	101	102	111	101	024	105	112	021	111	111	022	111	101	101	111	011	101	010	101	100	010	100	002	006	101	011	101	011	105	001	100	030	040	024	010	121	130	167	062	010	023	052	034	140	010	042	020	042	000	002	042	024	007	000	007	000	000	100	002	116	000
	200	000	010	176	101	177	111	111	022	105	111	011	111	111	021	111	101	101	111	011	101	010	177	100	024	100	014	010	101	011	121	031	111	177	100	140	030	010	170	111	070	000	052	010	010	177	076	140	024	024	010	101	101	001	177	024	000	137	007	066	133	100	002	131	177
	300	000	010	010	101	100	111	111	177	105	111	005	111	051	022	111	101	101	101	001	111	010	101	100	042	100	002	060	101	011	041	051	121	001	100	030	040	024	010	105	000,	167	072	010	144	052	034	000	042	010	004	000	042	131	042	024	007	000	000	066	073	100	002	046	177
	000	000	010	010	076	000	106	066	020	071	060	003	066	036	174	066	042	076	101	001	171	177	000	077	101	100	177	177	177	006	136	106	042	001	007	007	177	143	007	103	000	024	074	010	143	022	052	000	101	000	002	000	034	006	000	024	000	000	000	000	100	016	120	000
PAGE 05	100	000	START 201	@ 054	% 053	* 055	< 057	MINUS 206	DUP 207	— 001	/ 061	P.SEL 212	REL 213	BSR 214	E.O.F. 215	Q 035	W 043	E 021	R 036	T 040	Y 045	U 041	I 025	O 033	P 034	BSF 230	L/Z 231	ERR 232	000	000	000	000	A 015	S 037	D 020	F 022	G 023	H 024	J 026	K 027	L 030	SKIP 250	HOME 251	BLK 000	Z 046	000	000	X 044	000	C 017	V 042	B 016	N 032	M 031	, 047	056	COR 267	SPACE 000	EOJ 271							
KEYPUNCH TRANSLATIE TABLE →	200	000	START 201	# 050	, 047	$ 054	PERIOD 056	MINUS 206	DUP 207	— 001	ø 003	P.SEL 212	REL 213	BSR 214	E.O.F 215	÷ 002	– 074	) 063	¢ 065	T 040	↑ 077	1 004	2 005	3 006	& 076	BSF 230	L/Z 231	ERR 232	000	000	000	000	A 015	> 060	. 072	, 073	¬ 075	' 071	4 007	5 010	6 011	SKIP 250	HOME 251	BLK 000	Z 046	000	000	? 064	000	067	= 066	! 070	(062	7 012	8 013	9 014	COR 267	SPACE 000	EOJ 271							
	300																																																																						
PAGE 06	000																																																																								
	100																																																																								
	200																																																																					
	300																																																																					
PAGE 07	000																																																																		
	100																																																																					
	200																																																																					
	300																																																																					
		00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
																																																																							DECIMAL NOTATION

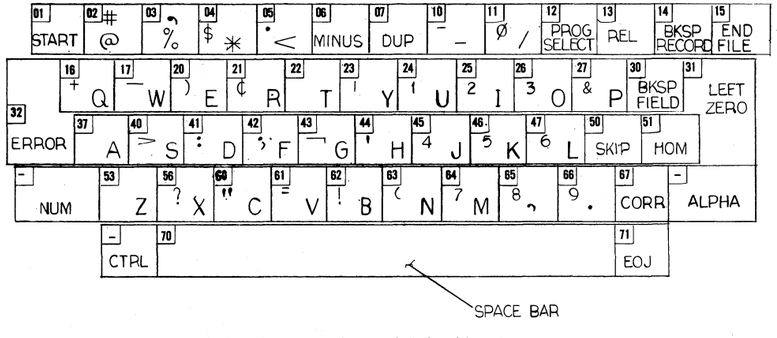
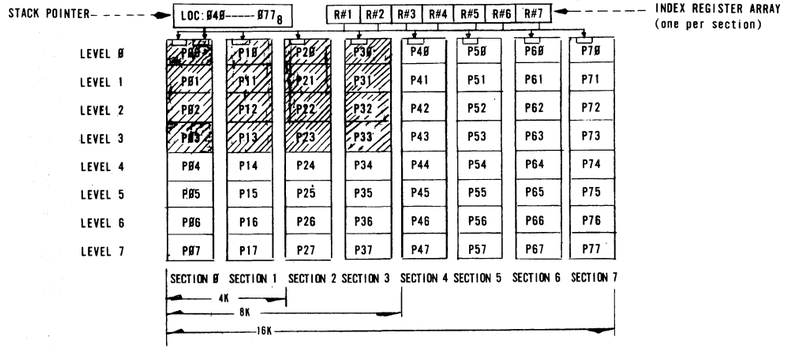


Figure 1b. Standard Keypunch Keyboard Layout.



*SHADED AREA = DISPLAYABLE PAGES

Figure 1c. Memory Map Layout.

TABLE I. KEY AND CHARACTER CODES FOR COGAR 4 KEYPUNCH KEYBOARD.

KEY	KEY CODE	CHAR CODE	KEY	KEY CODE	CHAR CODE	KEY	KEY CODE	CHAR CODE	KEY	KEY CODE	CHAR CODE	KEY	KEY CODE	CHAR CODE
Space	070	000	A	037	015	N	063	032	,	065	047	)	120	063
-	010	001	B	062	016	O	026	033	,	103	047	?	156	064
+	116	002	C	060	017	P	027	034	#	102	050	¢	121	065
Ø	111	003	D	041	020	Q	016	035	@	002	051	=	161	066
1	124	004	E	020	021	R	021	036	%	003	053	"	160	067
2	125	005	F	042	022	S	040	037	$	104	054	!	162	070
3	126	006	G	043	023	T	022	040	*	004	055	'	144	071
4	145	007	H	044	024	U	024	041	.	105	056	:	141	072
5	146	010	I	025	025	V	061	042	.	066	056	;	142	073
6	147	011	J	045	026	W	017	043	<	005	057	-	117	074
7	164	012	K	046	027	X	056	044	>	140	060	¬	143	075
8	165	013	L	047	030	Y	023	045	/	011	061	&	127	076
9	166	014	M	064	031	Z	053	046	(	163	062	|	123	077

CONTROL KEYS

KEY	KEY CODE	CHAR CODE	KEY	KEY CODE	CHAR CODE	KEY	KEY CODE	CHAR CODE
START	001	201	BKSP RECORD	014	214	HOM	051	251
MINUS	006	206	END FILE	015	215	CORR	067	267
DUP	007	207	BKSP FIELD	030	230	EOJ	071	271
PROG SELECT	012	212	ERROR	032	232	LEFT ZERO	031	231
REL	013	213	SKIP	050	250			

Write Pin Enable

A Write Pin Sensor in the SYSTEM 4 requires that if a tape is to be written on, the write plug must be in the proper position. Otherwise, tape will not move and no write operation can be performed on that deck until a cartridge is inserted with the write pin in place.

Physical End of Tape Sensing

The SYSTEM 4 tape cartridges contain a reflective spot to notify the program that during a write operation, the Physical End of Tape is approaching. The user may write beyond this point if so desired. The Mini-tape write Software function detects this condition and provides the tape status for the user to test. Once the EOT is detected, this condition remains set until a Rewind operation is initiated.

8. OPERATOR CONTROLS

A Switch Well located beneath the CRT screen contains eight sense switches, a Program Load/Program Interrupt switch, and a System Reset switch.

Sense Switches

These eight toggle switches may be manually set by the user to any combination of eight bits. The setting of these switches may then be tested by the user program at selected times, to control specialized applications.

Program Load/Program Interrupt Switch

This toggle switch initiates a tape load cycle when pushed toward the CRT (Momentary position), or initiates a Program Interrupt when set in the ON position (away from the CRT screen).

With the switch set to ON, the user program may test the condition to provide automatic linkage to the Interrupt Routine. Return to the point of interrupt will occur after the interrupt routine has been completed, and an Exit instruction to the Stack Level established by the interrupt has been executed.

System Reset Switch

When this push button switch is pressed, a System Reset pulse is generated which resets the Stack Pointer to Stack Level 1 and forces the instruction address to P02-000 where processing is then initiated.

SECTION II. INSTRUCTION USAGE

1. SUBROUTINE CONTROL:

The Instruction Address Stack (IAS) is located in memory and consists of sixteen Instruction Address Words (IAW) of two bytes each. Access to the Stack is under control of a four-bit register called the Stack Pointer. The current instruction address is contained in the IAW indicated by the Stack Pointer.

During sequential instruction operations, the Instruction Address is retrieved from the IAW, used to locate the current instruction, incremented by two, and inserted back into the IAW. For branch operations, a new Instruction Address is inserted into the current IAW and execution continues with the new address.

To enter a subroutine, the Stack Pointer is incremented so that it now points to a new IAW location and the subroutine address is inserted in the Stack as the new IAW. Normal sequential operation then proceeds. Note that the content of the previous IAW has not been disturbed and may be returned to by simply decrementing the Stack Pointer with an Exit instruction. Thus it is not necessary to provide space in the sub-routine for return address storage. If more than 16 levels of stack and branching has occured an automatic wrap-around to stack level 1 will be initiated.

Figure 2 is a diagram of the IAS and shows the actual octal locations of the stack bytes in page ØØ. Assume that the Stack Pointer is indicating IAW1 as the location of the current Instruction Address. Sequential or Branch operations of the mainline program change the contents of IAW1 but do not affect the Stack Pointer. When the mainline program encounters a Stack and Branch instruction, however, the Stack Pointer is incremented to indicate IAW2 and the Branch address is inserted into IAW2. If the Stack and Branch instruction was located at Page 10, location 52, IAW1 will now contain the coding to indicate Page 10, location 52, and IAW2 will become the current location counter. The subroutine indicated by IAW2 may reference other subroutines in which case IAW3, IAW4, etc. may be used. When the IAW2 subroutine is finished, an Exit instruction is executed which simply decrements the Stack Pointer and returns program control to IAW1 at the instruction following the original Stack and Branch. If the exit instruction was located at Page 13, location 220, IAW2 will be left with the coding for Page 13, location 220. A subsequent mainline Stack and Branch would insert a new Branch address into IAW2.

Note that the low order bit of the location may be on. This bit must be removed,by using the "ANA" instruction if the user desires to use this address after a load processor status operation (See "LPS" instruction)

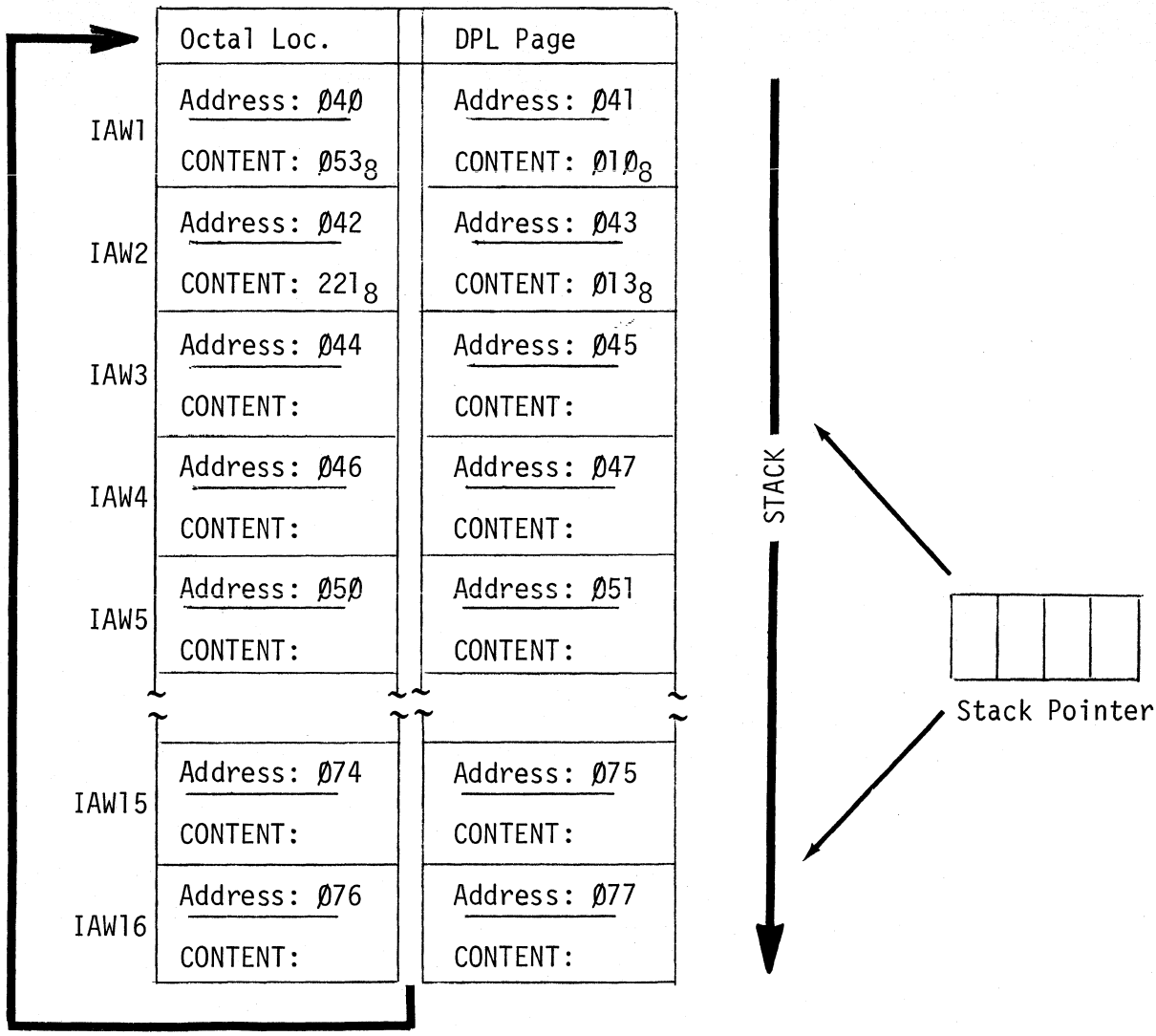
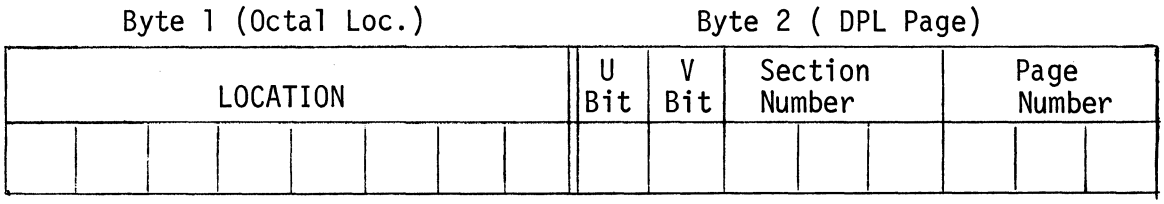


Figure 2. Snapshot of Instruction Address Stack after completion of EXU Instruction (See Example).

EXAMPLE:

```
PPP-LLL: MP1-MP2-MP3-MP4. E SEQ. NO. LAB: VERB  OPERANDS        COMMENTS

P1Ø-Ø5Ø: 2ØØ-Ø24.          Ø1-Ø1Ø.  EAB: LDA,  R#Ø; OCT:Ø24.
P1Ø-Ø52: 123-174.          Ø1-Ø2Ø.       SBU:  DLY.
P1Ø-Ø54: 237-Ø54.          Ø1-Ø3Ø.       STA,  R#7; P11.
P13-174: 23Ø-Ø11.          Ø1-Ø4Ø.  DLY: STA,  R#Ø; L#1.
P13-176: 2Ø3-234.          Ø1-Ø5Ø.       LDX,  R#3; DEC:156.
P13-2ØØ: Ø14-Ø27.          Ø1-Ø6Ø.       TLJ,  +12; (K).
P13-22Ø: 14Ø-ØØØ.          Ø1-Ø7Ø.       EXU:  ØØØ.
```



INSTRUCTION LOCATION

(Octal ØØØ to 376)

MEMORY CONTROL

Ø = Direct Address
within Section Ø, Page Ø.

1 = Direct Address
within current section, Page Ø.

RELOCATABLE BRANCH CONTROL

Ø = Normal processing.

1 = Relocatable Branch functions.
(See SMC Instruction)

SECTION NUMBER

Section Ø through Section 7.

PAGE NUMBER

Page Ø through Page 7 (within a section).

Figure 3. Instruction Address Word Layout.

2. REGISTERS:

The Cogar 4 contains one general purpose accumulator that is eight bits (one byte) long. Almost all of the nonbranch DPL-1 instructions refer to the accumulator. It is the major center for processor activity and the primary pipeline for data flow to and from the memory and the peripheral devices.

The Cogar 4 contains seven one-byte index registers for each memory section available. They are often used as address displacements in indexed addressing, but may also be used as general purpose registers. A few of the DPL-1 instructions act directly on the index registers, but there is much more flexibility than those instructions imply because the registers are located in memory. They may thus be addressed by all memory reference instructions. The accumulator can retrieve, manipulate and restore the contents of any index register.

The hardware condition register contains the results of Test and Compare instructions. It may be set to High, Equal or Low and retains its status until a new Test or Compare is executed. The operation of DPL-1 conditional Branch instructions depends on the status of the hardware condition register.

3. ADDRESSING:

The Cogar 4 contains 4K, 8K or 16K bytes of memory, with an IAW 16 bits long. Indirect addressing may operate anywhere within this range. The total memory capacity is divided into eight Sections of 2Ø48 bytes each, requiring 11 bits to fully address. Branch operations (if not preceded by a "SMS" instruction) may refer only to locations within a Section. Each Section is further divided into eight pages of 256 bytes each, requiring eight bits to fully address. Direct addressing (page Ø of the current control section) or relocatable subroutines (branch operations with page Ø assigned) may refer to one page only.

The object formats shown with the instruction descriptions include the following Binary Notations:

Z = 1 bit frame

Y = 2 bit frame

X = 3 bit frame

JJ = 4 bit frame

Instruction Addressing:

All instruction addressing is relocatable page oriented. The address specification, in octal notation (object), is Pnn-LLL where nn = SL, S is the Section number, L is the Level number and LLL is the byte location within the page.

All instructions are retrieved from memory using the current Instruction Address Word, and all instruction addressing involves modification of the IAW.

For sequential execution of instructions, one of the sixteen IAW's within the Stack is incremented by two during each instruction cycle. Instructions may be executed sequentially within a Section or across Section boundaries. It is important to note that when instructions cross a Section boundary, the branch functions, if executed, will transfer control to the Section that was previously set. Other functions are not affected. A "Set Memory Section" instruction is used to change the section context of the IAW for branch instructions.

A jump to a new instruction location uses relative instruction addressing by adding or subtracting up to 15 instruction locations to or from the current IAW. A Jump may be across a Section Boundary.

ADDRESS NOTATIONS

AAA	DDD = Absolute Address, in decimal notation SSS = Symbolic Address RRR = Symbolic Branch Reference NNN = Address Adjustment for Symbolic Addresses, in decimal notation
PPP	Pnn = Absolute Page Number, in decimal notation SSS = Symbolic Page Number

Data Addressing:

Data is addressed by an instruction in three different modes: Immediate, Direct and Indexed.

When using the Immediate Addressing Mode, the operand itself, instead of the operand address, is assembled within the instruction as a self-defining literal. The literal represents data rather than an address of data. Literals provide a means of entering constants into a program by specifying the constant in the operand of the instruction in which it is used. Immediate Addressing is differentiated from Direct Addressing by the operand form.

Direct Addressing Mode uses the instruction operand as the address of a byte location for all page numbers within level Ø. This mode is utilized by specifying in the operand, any form of Direct Address notation. All DPL-1 functions may take this form of operand except Class Ø and Class 1 Instructions.

The Indexed Addressing Mode provides a method of addressing data anywhere within memory. An Indexed Address is composed of a displacement address contained in a specified index register plus a base address contained in the operand. The register specifies the location within a page and the operand specifies the page within memory. The index register in use may be unchanged, incremented by one or decremented by one following the indexed operation. There are three forms of register notation used to specify this option. X may be any integer from 1 through 7.

R#X = Retain Register Value

I#X = Increment Register after Instruction Execution

D#X = Decrement Register after Instruction Execution

When an overflow occurs (I#X), the overflow bit is lost and the register contains octal ØØØ. When an underflow occurs (D#X), the result is the two's compliment of the underflow count.

4. SYMBOLS:

Program elements, such as instructions or constants, may be referenced in an instruction by specifying the absolute address of the element. The form for this type of reference is Pnn, LLL. Pnn specifies the page in 2 digit decimal notation from ØØ to 63 and LLL specifies the location within the page in 3 digit decimal notation from ØØØ to 255.

It is often more convenient to refer to program elements symbolically. In the DPL-3B Assembler, a symbol is a combination of characters used to represent a program element. Symbols are defined through their use in the label field of an instruction or through the EQU pseudo instruction. A Symbol may be used only once in a label field within one program. When a symbol is used as an instruction operand, it must be defined somewhere in the program. A symbol must be comprised of three non-blank alpha-numeric characters with the first character non-numeric. If the first character is "P", the following characters must be alphabetic. The total number of symbols plus ORG statements plus page boundaries crossed by sequential program operation is limited to a maximum of 128.

Address adjustment may be used for convenience and to cut down on the number of symbols defined. A signed numeric adjustment in decimal bytes from Ø to 255 may be appended to a symbolic reference or may be used relative to the current location. An "*" (asterisk) is used to indicate the location of the first byte of the current instruction.

The I/O Control Instruction micro-codes provide for control, status and data exchange between the processor and its interface devices. Tape channels may be selected, tape motions initiated, and read or write commanded; the keyboard may be read or beeped; the CRT may be enabled or disabled ; the I/O interface transmission may be started or stopped, and data or control bytes written. With the CRT enabled, the data content of any memory page which has a section or level number of less than 5 may be displayed in four-line consecutive mode, eight-line consecutive mode, or eight-line interleaved mode. Several status checks are available for the processor to interrogate. Most normal I/O operations will use the I/O Supervisor, but special purpose routines may be constructed from the IOC instructions and there are several operations, like keyboard beep, that are not available from the IOS.

5. DPL-1 INSTRUCTION CLASSES:

The DPL-1 instruction set includes all hardware instructions and is divided into four general classes covering all types of operations required of a general purpose processor.

Class Ø: Jump and Conditional Exit Instructions

Class 1: Branch, Linkage-Control, and I/O Instructions

Class 2: Data-Transfer and Arithmetic Instructions

Class 3: Boolean and Compare Instructions

Class Ø: Jump Instructions:

Jump instructions transfer control within a context to a location relative to the current instruction location. All Jump Instructions are conditional and depend on the result of a test of the contents of the accumulator. The test comparison, the test mask, the Jump direction and the jump increment are all specified in the instruction. The Jump increment is expressed in the instruction itself as the octal number of two-byte instructions to be jumped. However, the Batch Assembler uses a decimal byte count for the Jump increment. Test results are stored in the hardware condition register. For the TMJ and TMX instructions, an unconditional Jump or Exit, and the setting of the condition register to equal, can be effected by using a test mask of zero.

Class 1: Branch, Linkage-Control, and I/O Instructions:

Branch instructions transfer control outside a context to any section address. Branch instructions replace the current IAW with a new instruction address. Stack and Branch instructions introduce a new instruction address in a new IAW and preserve the contents of the previous IAW for return linkage. Direct Branch instructions may be conditioned by previous test or compare operations. The conditional instructions allow powerful data-dependent decisions to be made. The Exit and the Exit and Branch instructions are used to return from subroutines. They decrement the stack pointer and thus change program control to the next previous IAW.

Class 2: Data Transfer and Arithmetic Instructions:

This class of instructions includes the Load and Store operations that allow data to be moved between memory and the accumulator or index registers. These instructions use immediate, direct, or indexed addressing modes. When loading or storing using indexed addressing, the specified index register may be automatically incremented or decremented.

The arithmetic instructions in this class include Binary add and subtract operations on the accumulator or the index registers. Immediate, direct, or indexed addressing may be used. Automatic increment or decrement of index registers may be specified when using indexed addressing. All operations are available for use with the accumulator. Some operations may also be performed on index registers.

Class 3: Boolean and Compare Instructions:

The Boolean instructions in this class include immediate, direct or indexed addressing of And, Inclusive Or, and Exclusive Or operations. The immediate instructions allow for up to seven right circular shifts of the accumulator prior to operation with the literal.

The Compare instructions compare the contents of the accumulator with a location specified by immediate, direct or indexed addressing. Any index register may be compared with a literal. The comparison results are stored in the condition register and may be tested by any following conditional Branch instruction. In indexed addressing of both Boolean and Compare instructions, the specified index register may be automatically incremented or decremented.

6. DPL PUNCTUATION:

Rather than an implicit syntax, the DPL grammar provides an explicit syntax by use of punctuation. Four punctuation characters are used: the semi-colon, the comma, the colon and the period.

The semi-colon is used as an imperative terminator or a major field delimiter. It usually separates the instruction field from the operand field.

The comma is used as a minor field terminator. It separates multiple field instructions or operands.

The colon is used as a declarative terminator. It follows instruction labels, pseudo instructions and constant designators.

The period is used as a closing terminator and defines the end of the symbolic instruction.

7. LITERAL NOTATIONS:

Literal notations may be classified as explicit terms or as implicit terms. Explicit literals are self-defining because they include the specific value to be used. The four explicit literal forms are Character, Octal, Hexadecimal, and Decimal. They provide a means of specifying values or bit configurations without equating the values to symbols. The value of an explicit literal is assembled into an instruction. The value of a symbolic constant resides in memory and its address is assembled into an instruction.

Literals that are assigned a value by the DPL-3B Assembler use five forms of address constants in which AAA is a symbolic address. These are: ADC:AAA, ADL:AAA, ADP:AAA, IDP:AAA, and DDP:AAA. These address constants are used primarily to define the actual address of a symbolic reference. When the literal form ADP, IDP, or DDP is used in conjunction with an R#Ø or an R#X, instruction, the DPL page value of AAA is assembled as the operand; either with no indexing tag, or with incrementing or decrementing tag, respectively. If the form ADL is used, the address location value within the page is assembled as the operand.

When the literal form ADC is used in conjunction with an R#0 instruction, the DPL page value, in increment form, is assembled as the operand. If used in conjunction with an R#X instruction, the symbolic address location within the page is assembled as the operand.

8. STANDARD C4 PROGRAM RECORD (Mini-Tape)

The Standard Mini-Tape Record is comprised of an 8-byte label, generated by the Mini-Write Software Function, followed by 128 bytes of data. The 8-byte label when read into (or written from) memory resides in Page 00, locations 030_8 thru 037_8. The first byte of the Record Header contains a sequence number. The sequence number is automatically checked by the Mini-Read Software Function to provide a method of automatically bypassing any "CIG" (Character in Gap). This sequence number may also be used to adjust search counters when utilizing the high-speed capability to locate multiple records by continuation. Byte-2 contains the control function. A value other than those specified below may be inserted by the user for specialized functions. Bytes 3 and 4 are not used by the Standard Mini-Read/Write, and can, therefore, contain any value as established by the user.

Bytes 5 through 8 of a program record contain the Segment ID and the Page Designator. Through usage of these bytes, an overlay record can easily be located and loaded into memory. Bytes 5 through 8 are not used in a data file.

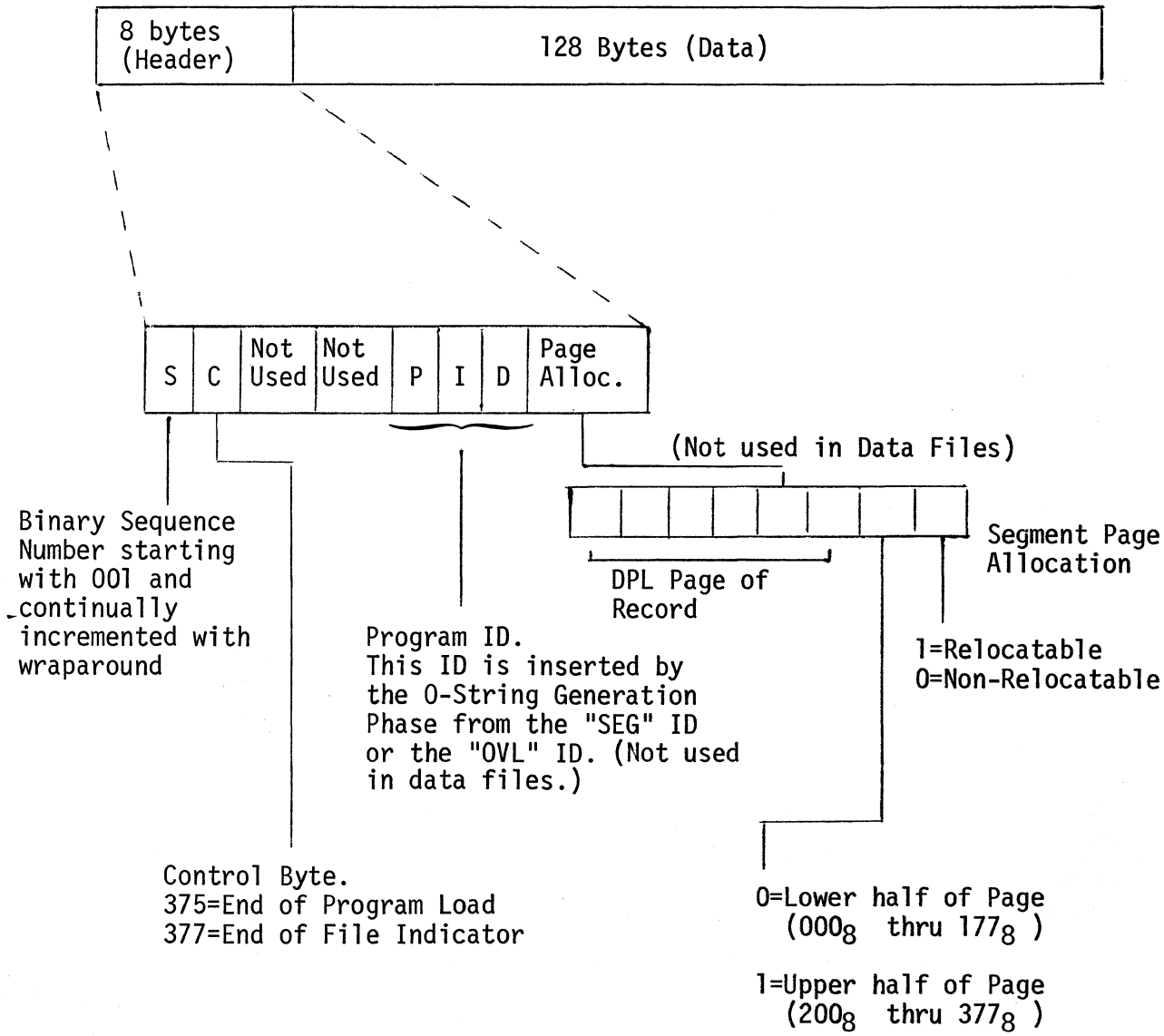


Figure 4. Standard Mini-Tape Record Layout.

9. SUBROUTINE RELOCATABILITY

A method has been provided to allow the user to write subroutines that may be executed within any Page without re-assembling the subroutine for that Page. By executing a SET Memory Control Command that sets the Relocatable Branch Control (RBC) Bit, any Branch, Stack and Branch or Exit and Branch Instruction given with Page 0 specified in the Branch Address will cause the Branch to occur within the current Section and Page of the program. If any Page other than 0 is specified in the Branch Address, the RBC-Bit is Inactive and a normal Branch function will occur.

10. TAPE I/O CHARACTER QUEUE

The SYSTEM 4 tape logic contains an 8-bit chararacter buffer that will hold a character for 512 usec., allowing this much time for other processing before the user must return to the I/O operation.

For DPL-1 instructions that use Immediate Addressing, the following forms may be used in symbolic coding to specify the literal value:

(K)	Where K is a valid keyboard character
OCT:NNN	Where NNN is a one-byte constant in OCTAL notation from ØØØ to 377.
HEX:HH	Where HH is a one-byte constant in HEXA-DECIMAL notation from ØØ to FF.
DEC:NNN	Where NNN is a one-byte constant in DECIMAL notation from ØØØ to 255.
ADP:AAA	Where AAA is an address constant for a PAGE in symbolic notation (without Auto Indexing).
IDP:AAA	Where AAA is an address constant for a PAGE in symbolic notation (with Increment Auto. Indexing).
DDP:AAA	Where AAA is an address constant for a PAGE in symbolic notation (with Decrement Auto. Indexing).
ADL:AAA	Where AAA is an address constant for a LOCATION in symbolic notation.
ADC:AAA	An address constant for labels, in symbolic notation (will generate page or location dependent on the Instruction form).
AAA±NNN	Where AAA is an address constant for a location in symbolic notation, and NNN is offset ± from that location.

SECTION III. INSTRUCTION DESCRIPTIONS

The instructions described in this section of the manual are presented in the same order as they appear on the Cogar System 4 Instruction Reference Card, and fall in the following four categories:

1. DPL-1 Instructions. These instructions perform all the data manipulation and control tasks allowed by the hardware.

2. IOS Commands. These instructions provide access to the standard software I/O routines, using the I/O Supervisor.

3. Pseudo Instructions. These instructions provide programmer control over the DPL-3B Assembler, and the resident monitors.

4. Constants. Byte constants or string constants may be generated using these notations.

CLASS Ø: JUMP TEST LITERAL AND JUMP

TLJ

OBJECT	SOURCE
ØØØJJØ-LLL	TLJ, +NNN; Literal.
ØØØJJ1-LLL	TLJ, -NN; Literal.

WHERE: JJ is the jump count in 4 Bit Binary notation, indicating the number of 2-Byte instructions to be jumped.

AND: LLL is an 8 Bit Literal.

WHERE: NN is the jump count in decimal notation, indicating the number of bytes to be jumped.

NOTE: This jump count must always be an even decimal number (Max:30).

DESCRIPTION:

The Accumulator is compared to the byte of immediate data (literal), and the result is indicated in the condition register. Comparison is binary, and all codes are valid. If the resulting condition register is equal, a jump forward (+) or a jump backward (-) up to 15 two-byte instruction locations is performed. If however the resulting condition register is not equal (high or low), the next sequential instruction is executed. The character in the Accumulator is not changed. Once set, the condition register remains unchanged until modified by the next jump or compare instruction that reflects a different condition code.

NOTE: The condition register contains the true arithmetic condition (high or low) after an unsuccessful jump (unequal condition).

HIGH	ACCUM	>	LITERAL
LOW	ACCUM	<	LITERAL
EQUAL	ACCUM	=	LITERAL

TIMING: 3 Microseconds if the jump is not performed.
4 Microseconds if the jump is performed.

EXAMPLE:

```
PPP-LLL: MP1-MP2-MP3-MP4. E SEQ. NO. LAB: VERB  OPERANDS          COMMENTS

P15-ØØØ: Ø12-Ø15.          Ø1-12Ø.       TLJ,  +1Ø; (A).        JUMP IF
P15-ØØ2: Ø13-Ø16.          Ø1-13Ø.       TLJ,  -1Ø; OCT:Ø16.    ACCUM IS
P15-ØØ4: ØØ6-Ø15.          Ø1-14Ø.       TLJ,  +Ø6; DEC:Ø13.    EQUAL
P15-ØØ6: ØØ4-Ø17.          Ø1-15Ø.       TLJ,  +Ø4; HEX:ØF.
```

CLASS Ø: JUMP TEST MASK AND JUMP

TMJ

OBJECT	SOURCE
ØØ1JJØ-MMM	TMJ, +NN; LT-MASK.
ØØ1JJ1-MMM	TMJ, -NN; LT-MASK.

WHERE: JJ is the jump count in 4 Bit Binary notation, indicating the number of 2-Byte Instructions to be jumped.

AND: MMM is an 8 Bit Literal Mask.

WHERE: NN is the jump count in decimal notation, indicating the number of bytes to be jumped.

NOTE: This jump count must always be an even decimal number (Max:30).

DESCRIPTION:

The state of the Accumulator bits selected by a mask is used to set the condition code.

The byte of Immediate Data (Literal-Mask) is used as an eight-bit mask. The bits of the mask are made to correspond one for one with the bits of the character in the Accumulator. A mask bit of one indicates that the corresponding Accumulator bit is to be tested. When the mask bit is zero, the corresponding Accumulator bit is ignored. When any of the Accumulator bits thus selected are zero, the Condition Register is made unequal. When the selected bits are all-one, the Condition Register is made equal. If the resulting Condition Register is equal, jump forward (+) or jump back (-) up to 15 two-byte instruction locations. On the resulting Condition Register not equal (high or low), execute the next sequential instruction. The character in the Accumulator is not changed. Once set, the Condition Register remains unchanged until modified by an instruction that reflects a different condition code.

NOTE: The content of the Condition Register is unpredictable after an unsuccessful jump (unequal condition).

TIMING: 3 Microseconds if the jump is not performed.
4 Microseconds if the jump is performed.

EXAMPLE:

```
PPP-LLL: MP1-MP2-MP3-MP4. E SEQ. NO. LAB: VERB  OPERANDS          COMMENTS

P15-Ø1Ø: Ø5Ø-Ø16.           Ø2-Ø1Ø.       TMJ,  +Ø8; OCT:Ø16.   JUMP IF
P15-Ø12: Ø51-Ø5Ø.           Ø2-Ø2Ø.       TMJ,  -Ø8; DEC:Ø4Ø.   MASK IS
P15-Ø14: Ø76-377.           Ø2-Ø3Ø        TMJ,  +3Ø; HEX:FF.    EQUAL
```

·DPL-1 CLASS Ø: TEST LITERAL AND EXIT

TLX

OBJECT	SOURCE
ØØØ-LLL	TLX, ØØØ; Literal.

WHERE: LLL is an 8 bit Literal.

DESCRIPTION:

The Accumulator is compared to the byte of immediate data (literal), and the result is indicated in the Condition Register. Comparison is binary, and all codes are valid. If the resulting Condition Register is equal, then a special form of exit, (conditional exit) is performed, which completes the return linkage established by the last executed stack and branch instruction. The stack pointer is decremented to the preceding stack level, which contains the address of the last stack and branch instruction executed. This address is then incremented by 2 bytes, which establishes the address of the instruction following the stack and branch instruction, and a new location counter value. This value is the new instruction address, where processing continues.

The exit function may return within a section or outside a section without any special consideration, since the stack contains the page and location of the return address.

NOTE: The Condition Register contains the true arithmetic condition (high or low) after an unsuccessful Jump (unequal condition).

HIGH	ACCUM	>	LITERAL
LOW	ACCUM	<	LITERAL
EQUAL	ACCUM	=	LITERAL

TIMING: 3 Microseconds if the Jump is not performed.
4 Microseconds if the Jump is performed.

EXAMPLE:

```
PPP-LLL: MP1-MP2-MP3-MP4. E SEQ. NO. LAB: VERB  OPERANDS        COMMENTS

P15-Ø16: ØØØ-Ø17.           Ø2-Ø9Ø.       TLX,  ØØØ; (C).      EXIT IF
P15-Ø2Ø: ØØØ-Ø13.           Ø2-1ØØ.       TLX,  ØØØ; OCT:Ø13.  ACCUM IS
P15-Ø22: ØØØ-Ø16.           Ø2-11Ø.       TLX,  ØØØ; DEC:Ø14.  EQUAL
P15-Ø24: ØØØ-377.           Ø2-12Ø.       TLX,  ØØØ; HEX:FF.
```

·DPL-1 CLASS Ø: TEST MASK AND EXIT

TMX

OBJECT	SOURCE
Ø4Ø-MMM	TMX, ØØØ; LT-MASK.

WHERE: MMM is an 8 bit Literal Mask.

DESCRIPTION:

The state of the Accumulator bits selected by a mask is used to set the condition code.

The byte of Immediate Data (Literal-Mask) is used as an eight-bit mask. The bits of the mask are made to correspond one for one with the bits of the character in the Accumulator. A mask bit of one indicates that the corresponding Accumulator bit is to be tested. When the mask bit is zero, the corresponding Accumulator bit is ignored. When any of the Accumulator bits thus selected are zero, the Condition Register is made unequal. When the selected bits are all one, the Condition Register is made equal. If the resulting Condition Register is equal, then a special form of exit, (conditional exit) is performed, which completes the return linkage established by the last executed stack and branch instruction. The stack pointer is decremented to the preceding stack level, which contains the address of the last stack and branch instruction executed. This address is then incremented by 2 bytes, which establishes the address of the instruction following the stack and branch instruction, and a new location counter value. This value is the new instruction address, where processing continues.

The exit function may return within a section or outside a section without any special consideration, since the stack contains the page and location of the return address.

NOTE: The content of the Condition Register is unpredictable after an unsuccessful Jump (unequal condition).

TIMING: 3 Microseconds if the Jump is not performed.
4 Microseconds if the Jump is performed.

EXAMPLE:

```
PPP-LLL: MP1-MP2-MP3-MP4. E SEQ. NO. LAB: VERB  OPERANDS          COMMENTS

P15-Ø26: Ø4Ø-Ø1Ø.          Ø2-18Ø.       TMX,  ØØØ; OCT:Ø1Ø.   EXIT IF
P14-Ø3Ø: Ø4Ø-31Ø.          Ø2-19Ø.       TMX,  ØØØ; DEC:2ØØ.   MASK IS
P15-Ø32: Ø4Ø-24Ø.          Ø2-2ØØ.       TMX,  ØØØ; HEX:AØ.    EQUAL
```

·DPL-1 CLASS 1: BRANCH BRANCH UNCONDITIONAL

BRU

OBJECT	SOURCE
1ØX-YXYØ	; RRR±NNN. BRU , Pnn; LLL. ; *±NNN.

WHERE: 1ØX is the command, in which X is the page.

AND: YXY is a 7 bit address.

WHERE: RRR is a symbolic address

AND: NNN is a decimal byte displacement.

AND: nn is a decimal page.

AND: LLL is a decimal location

AND: * is the location of the instruction itself.

DESCRIPTION:

The unconditional branch is performed by introducing a branch address as a new instruction address, regardless of the setting of the Condition Register.

The Branch Address may be represented in symbolic notation, as an absolute address; or as a relative address. The Branch Address may be any location within the current section. "OUT-OF-SECTION" branching is achieved by preceding the branch instruction with a SET MEMORY SECTION (SMS) instruction, or a SET memory SECTION & CONTROL (SSC) instruction. "WITHIN-A-PAGE" branching relocatability is achieved by preceding the branch instruction with a SET MEMORY CONTROL (SSC) instruction in which the RELOCATABLE BRANCH CONTROL (RBC) bit is set. (i.e.: C#1 or C#3). The hardware condition register remains unchanged after execution of a branch function.

TIMING: 4 Microseconds.

EXAMPLE:

```
PPP-LLL: MP1-MP2-MP3-MP4. E SEQ. NO. LAB: VERB  OPERANDS        COMMENTS

P15-Ø34: 1Ø5-Ø42.           Ø3-Ø3Ø.       BRU; IN1.            WITHIN A
P15-Ø36: 1Ø5-Ø36.           Ø3-Ø4Ø.       BRU; *+Ø.            SECTION
P15-Ø4Ø: 1Ø7-ØØØ.           Ø3-Ø5Ø.       BRU, P15; ØØØ.
P15-Ø42: 15Ø-ØØØ.           Ø3-Ø6Ø. IN1: SMS; S#Ø.            OUT OF A
P15-Ø44: 1Ø6-144.           Ø3-Ø7Ø.       BRU, PØ6; 1ØØ.       SECTION
```

CLASS 1: BRANCH BRANCH ON EQUAL

BRE

OBJECT	SOURCE
1ØX-YXY1	; RRR+NNN. BRE , Pnn; LLL. ; *+NNN.

WHERE: 1ØX is the command, in which X is the page.
AND: YXY is a 7 bit address.

WHERE: RRR is a symbolic address.
AND: NNN is a decimal byte displacement.
AND: nn is a decimal page.
AND: LLL is a decimal location.
AND: * is the location of the instruction itself.

DESCRIPTION:

The conditional branch instruction, branch on equal, is performed when the condition register, set by a previous compare or test instruction, is found to be equal. If this condition is not satisfied, the next sequential instruction is executed. The conditional branch is performed by introducing a branch address as a new instruction address.

(Refer to "BRU" for Basic Rules of Branching).

TIMING: 3 Microseconds if the branch is not performed.
4 Microseconds if the branch is performed.

EXAMPLE:

```
PPP-LLL: MP1-MP2-MP3-MP4. E SEQ. NO. LAB: VERB  OPERANDS        COMMENTS

P15-Ø46: 1Ø5-Ø51.           Ø3-13Ø       BRE;  IN2.             WITHIN SECT.
P15-Ø5Ø: 15Ø-ØØØ.           Ø3-14Ø.  IN2: SMS;  S#Ø.            OUT OF A
P15-Ø52: 34Ø-2ØØ.           Ø3-15Ø.      CPA,  R#Ø; OCT:2ØØ.    SECT. IF
P15-Ø54: 1Ø6-145.           Ø3-16Ø.      BRE,  PØ6; 1ØØ.        EQUAL-ELSE
P15-Ø56: 15Ø-Ø1Ø.           Ø3-17Ø.      SMS;  S#1.             RESET SECT.
```

·DPL-1 CLASS 1: BRANCH BRANCH ON HIGH

BRH

OBJECT	SOURCE
11X-YXYØ	; RRR±NNN. BRH , Pnn; LLL. ; *±NNN.

WHERE: 11X is the command, in which X is the page.
AND: YXY is a 7 bit address.

WHERE: RRR is a Symbolic address.
AND: NNN is a decimal byte displacement.
AND: nn is a decimal page.
AND: LLL is a decimal location.
AND: * is the location of the instruction itself.

DESCRIPTION:

The conditional branch instruction, branch on high, is performed when the condition register, which has been set by a previous compare or test instruction, is found to be high. If this condition is not satisfied, the next sequential instruction is executed.

(Refer to "BRU" for Basic Rules of Branching).

TIMING: 3 Microseconds if the branch is not performed.
4 Microseconds if the branch is performed.

EXAMPLE:

```
PPP-LLL: MP1-MP2-MP3-MP4. E SEQ. NO. LAB: VERB  OPERANDS          COMMENTS

P15-Ø6Ø: 115-Ø62.          Ø4-Ø3Ø.       BRH;  *+Ø2.             WITHIN SECT.
P15-Ø62: 15Ø-ØØØ.          Ø4-Ø4Ø.  IN3: SMS;  S#Ø.              OUT OF A
P15-Ø64: 34Ø-2ØØ.          Ø4-Ø5Ø.       CPA,  R#Ø; OCT:2ØØ.     SECT. IF
P15-Ø66: 116-144.          Ø4-Ø6Ø.       BRH,  PØ6; 1ØØ.         HIGH-ELSE
P15-Ø7Ø: 15Ø-Ø1Ø.          Ø4-Ø7Ø.       SMS;  S#1.              RESET SECT.
```

·DPL-1 CLASS 1: BRANCH BRANCH ON LOW

BRL

OBJECT	SOURCE
11X-YXY1	; RRR±NNN. BRL , Pnn; LLL. ; *±NNN.

WHERE: 11X is the command, in which X is the page.
AND: YXY is a 7 bit address.

WHERE: RRR is a Symbolic address.
AND: NNN is a decimal byte displacement.
AND: nn is a decimal page.
AND: LLL is a decimal location.
AND: * is the location of the instruction itself.

DESCRIPTION:

The conditional branch instruction, branch on low, is performed when the condition register, set by a previous compare or test instruction, is found to be low. If this condition is not satisfied, the next sequential instruction is executed. The conditional branch is performed by introducing a branch address as a new instruction address.

(Refer to "BRU" for Basic Rules of Branching)

TIMING: 3 Microseconds if the branch is not performed.
4 Microseconds if the branch is performed.

EXAMPLE:

```
PPP-LLL: MP1-MP2-MP3-MP4. E SEQ. NO. LAB: VERB  OPERANDS          COMMENTS

P15-Ø72: 115-Ø75.          Ø4-13Ø.       BRL;  IN4.              WITHIN SECT.
P15-Ø74: 15Ø-ØØØ.          Ø4-14Ø.  IN4: SMS;  S#Ø.              OUT OF A
P15-Ø76: 34Ø-2ØØ.          Ø4-15Ø.       CPA,  R#Ø; OCT:2ØØ.     SECT. IF
P15-1ØØ: 116-145.          Ø4-16Ø.       BRL,  PØ6; 1ØØ.         LOW-ELSE
P15-1Ø2: 15Ø-Ø1Ø.          Ø4-17Ø.       SMS;  S#1.              RESET SECT.
```

.DPL-1 CLASS 1: BRANCH STACK AND BRANCH UNCONDITIONAL

SBU

OBJECT	SOURCE
12X-YXYØ	; RRR±NNN. SBU , Pnn; LLL. ; *±NNN.

WHERE: 12X is the command, in which X is the page,
AND: YXY is a 7 bit address.

WHERE: RRR is a symbolic address.
AND: NNN is a decimal byte displacement.
AND: nn is a decimal page.
AND: LLL is a decimal location.
AND: * is the location of the instruction itself.

The Stack and Branch Unconditional Instruction is performed regardless of the setting of the condition register.

DESCRIPTION:

The stack and branch instructions are in contrast with the branch instructions, in that the stack and branch instructions preserve the current value of the location counter which is present in the current stack; this is performed by incrementing the stack pointer to the next stack level and creating a new location counter value containing the branch address as a new instruction address, within that stack. Thus, the return linkage between sub-routines is established. For the stack and branch function there are sixteen levels of stacks that the stack pointer can address, of which fifteen levels of stacks may temporarily preserve the return linkages for fifteen levels of stack and branching.

TIMING: 3 Microseconds

EXAMPLE:

```
PPP-LLL: MP1-MP2-MP3-MP4. E SEQ. NO. LAB: VERB  OPERANDS        COMMENTS

P15-1Ø4: 125-112.           Ø5-Ø4Ø.       SBU;  IN5.           WITHIN A
P15-1Ø6: 125-112.           Ø5-Ø5Ø.       SBU;  *+Ø4.          SECTION
P15-11Ø: 127-ØØØ.           Ø5-Ø6Ø.       SBU,  P15; ØØØ.
P15-112: 15Ø-ØØØ.           Ø5-Ø7Ø.  IN5: SMS;  S#Ø.           OUT OF A
P15-114: 126-144.           Ø5-Ø8Ø.       SBU,  PØ6; 1ØØ.      SECTION
```

·DPL-1 CLASS 1: BRANCH STACK AND BRANCH EQUAL

SBE

OBJECT	SOURCE
12X-YXY1	; RRR+NNN. SBE , Pnn; LLL. ; *+NNN.

WHERE: 12X is the command, in which X is the page.
AND: YXY is a 7 bit address.

WHERE: RRR is a symbolic address.
AND: NNN is a decimal byte displacement.
AND: nn is a decimal page.
AND: LLL is a decimal location.
AND: * is the location of the instruction itself.

DESCRIPTION:

The conditional stack and branch, stack and branch equal, is performed when the condition register, set by a previous compare or test instruction, is found to be equal. If the condition is not satisfied, the next sequential instruction is executed.

(Refer to "SBU" for Basic Rules of Stack and Branching).

TIMING: 3 Microseconds if the stack and branch is not performed.
4 Microseconds if the stack and branch is performed.

EXAMPLE:

```
PPP-LLL: MP1-MP2-MP3-MP4. E SEQ. NO. LAB: VERB  OPERANDS        COMMENTS

P15-116: 125-121.          Ø5-14Ø.       SBE;  *+Ø2.           WITHIN SECT.
P15-12Ø: 15Ø-ØØØ.          Ø5-15Ø.  IN6: SMS;  S#Ø.            OUT OF A
P15-122: 34Ø-2ØØ.          Ø5-16Ø.       CPA,  R#Ø; OCT:2ØØ.   SECTION IF
P15-124: 126-145.          Ø5-17Ø.       SBE,  PØ6; 1ØØ.       EQUAL-ELSE
P15-126: 15Ø-Ø1Ø.          Ø5-18Ø.       SMS;  S#1.            RESET SECT.
```

.DPL-1 CLASS 1: BRANCH STACK AND BRANCH ON HIGH

SBH

OBJECT	SOURCE
13X-YXYØ	; RRR±NNN. SBH , Pnn; LLL. ; *±NNN.

WHERE: 13X is the command, in which X is the page.
AND: YXY is a 7 bit address.

WHERE: RRR is a symbolic address.
AND: NNN is a decimal byte displacement.
AND: nn is a decimal page.
AND: LLL is a decimal location.
AND: * is the location of the instruction itself.

DESCRIPTION:

The conditional stack and branch, stack and branch on high, is performed when the condition register, set by a previous compare or test instruction, is found to be high. If the condition is not satisfied the next sequential instruction is executed.

(Refer to "SBU" for Basic Rules of Stack and Branching)

TIMING: 3 Microseconds if the stack and branch is not performed.
4 Microseconds if the stack and branch is performed.

EXAMPLE:

```
PPP-LLL: MP1-MP2-MP3-MP4. E SEQ. NO. LAB: VERB  OPERANDS          COMMENTS

P15-13Ø: 135-132.          Ø6-Ø4Ø.       SBH;  IN7.              WITHIN SECT.
P15-132: 15Ø-ØØØ.          Ø6-Ø5Ø.  IN7: SMS;  S#Ø.              OUT OF A
P15-134: 34Ø-2ØØ.          Ø6-Ø6Ø.       CPA,  R#Ø; OCT:2ØØ.     SECTION IF
P15-136: 136-144.          Ø6-Ø7Ø.       SBH,  PØ6; 1ØØ.         HIGH-ELSE
P15-14Ø: 15Ø-Ø1Ø.          Ø6-Ø8Ø.       SMS;  S#1.              RESET-SECT.
```

CLASS 1: BRANCH STACK AND BRANCH ON LOW

SBL

OBJECT	SOURCE
13X-YXY1	; RRR±NNN. SBL , Pnn; LLL. ; *±NNN.

WHERE: 13X is the command, in which X is the page.
AND: YXY is a 7 bit address.

WHERE: RRR is a symbolic address.
AND: NNN is a decimal byte displacement.
AND: nn is a decimal page.
AND: LLL is a decimal location.
AND: * is the location of the instruction itself.

DESCRIPTION:

The conditional stack and branch, stack and branch on low, is performed when the condition register, set by a previous compare or test instruction, is found to be low. If the condition is not satisfied the next sequential instruction is executed.

(Refer to "SBU" for Basic Rules of Stack and Branching)

TIMING: 3 Microseconds if the stack and branch is not performed.
4 Microseconds if the stack and branch is performed.

EXAMPLE:

```
PPP-LLL: MP1-MP2-MP3-MP4. E SEQ. NO. LAB: VERB  OPERANDS          COMMENTS

P15-142: 135-145.          Ø6-14Ø.       SBL;  IN8.              WITHIN SECT.
P15-144: 15Ø-ØØØ.          Ø6-15Ø.  IN8: SMS;  S#Ø.              OUT OF A
P15-146: 34Ø-2ØØ.          Ø6-16Ø.       CPA,  R#Ø; OCT:2ØØ.     SECTION IF
P15-15Ø: 136-145.          Ø6-17Ø.       SBL,  PØ6; 1ØØ.         LOW-ELSE
P15-152: 15Ø-Ø1Ø.          Ø6-18Ø.       SMS;  S#1.              RESET SECT.
```

'DPL-1 CLASS 1: BRANCH EXIT AND BRANCH

EXB

OBJECT	SOURCE
16X-YXYØ	; RRR±NNN. EXB , Pnn; LLL. ; *±NNN.

WHERE: 16X is the command, in which X is the page.
AND: YXY is a 7 bit address.

WHERE: RRR is a symbolic address.
AND: NNN is a decimal byte displacement.
AND: nn is a decimal page.
AND: LLL is a decimal location.

DESCRIPTION:

The exit and branch instruction combines the functions of the exit instruction and the branch unconditional instruction. This form of exit does not perform the return linkage established by the preceding stack and branch instruction. The stack pointer is decremented to the preceding stack level. The address specified in the operand is then used to establish a new location counter value within that stack. This value is the new instruction address within the current section, where processing continues.

TIMING: 4 Microseconds

EXAMPLE:

```
PPP-LLL: MP1-MP2-MP3-MP4. E SEQ. NO. LAB: VERB  OPERANDS       COMMENTS

P15-154: 165-156.          Ø7-Ø5Ø.      EXB;  *+Ø2.          WITHIN SECT.
P15-156: 15Ø-ØØØ.          Ø7-Ø6Ø.      SMS;  S#Ø.           OUT OF A
P15-16Ø: 166-144.          Ø7-Ø7Ø.      EXB,  PØ6; 1ØØ.      SECTION
```

CLASS 1: BRANCH EXIT UNCONDITIONAL

EXU

OBJECT	SOURCE
14Ø-ØØØ	EXU; ØØØ.

.WHERE: 14Ø is the command.
AND: ØØØ is the 8-Bit Operand

DESCRIPTION:

This form of exit, exit unconditional, performs the return linkage established by the last executed stack and branch instruction. The stack pointer is decremented to the preceding stack level, which contains the address of the last stack and branch instruction executed. This address is then incremented by 2 bytes which establishes the address of the instruction following the stack and branch instruction, and a new location counter value. This value is the new instruction address, where processing continues.

The exit function may return within a section or outside a section without any special consideration, since the stack contains the page and location of the return address.

The condition register is not changed by this instruction.

TIMING: 4 Microseconds.

EXAMPLE:

```
PPP-LLL: MP1-MP2-MP3-MP4. E SEQ. NO. LAB: VERB  OPERANDS        COMMENTS

P15-162: 15Ø-ØØØ.          07-130.       SMS;  S#Ø.            LINK OUT OF
P15-164: 126-144.          07-140.       SBU;  OUT.            A SECTION
                          *07-150.
                           07-160.       ORG:  PØ6, 1ØØ.
PØ6-144: 213-ØØ6.          07-170.  OUT: LDA,  I#3; PØ1.
PØ6-146: 14Ø-ØØØ.          07-180.       EXU;  ØØØ.            RETURN
```

·DPL-1 CLASS 1: BRANCH SET MEMORY SECTION

SMS

OBJECT	SOURCE
15Ø-ØXØ	SMS; S#X.

WHERE: X is the section number (Ø-7).

DESCRIPTION:

The set section instruction provides a means of transferring control from the current section to an outside section. A branch function (Branch, Stack & Branch or Exit & Branch) preceded by an SMS command will transfer control to the address defined by the branch address and the section specified in the set section operand.

Note that once the SMS instruction has been executed, transfer to that section will only be made when an unconditional branch function is executed or a conditional branch function that is found to be true.

The condition register is not changed by this instruction.

TIMING: 4 Microseconds.

EXAMPLE:

```
PPP-LLL: MP1-MP2-MP3-MP4. E SEQ. NO. LAB: VERB  OPERANDS        COMMENTS

P15-ØØØ: 15Ø-ØØØ.          Ø8-Ø7Ø.       SMS;  S#Ø.           SET SECTION Ø
P15-ØØ2: 1Ø2-ØØØ.          Ø8-Ø8Ø.       BRU,  PØ2; ØØØ.      BRANCH PAGE 2
```

SMC

OBJECT	SOURCE
151-YØØ	SMC; C#Y.

WHERE: Y = Ø Resets U & V control bits,
Y = 1 Sets V Bit, Resets U Bit,
Y = 2 Sets U Bit, Resets V Bit,
Y = 3 Sets U and V Bits.

DESCRIPTION:

When the U bit is set to Ø, the address of the index registers is memory location 1-7 and direct addressing is only available in page Ø of section Ø. When the U bit is set to 1, the effective index register address is location 1-7 of the section where the indexed instruction is being executed. Likewise the effective direct address is page Ø of the section where the direct address instruction is being executed.

When the V bit is set to 1 any branch, stack & branch or exit & branch instructions given with page Ø specified in the branch address will cause the branch to occur within the currrent section and page of the program. If any page other than Ø is specified in the branch address, the V bit control is inactive and a normal branch will occur. For example, assume that the V bit is set and a branch instruction located in page 5 specifies a branch to page Ø location AAA. The resulting branch will be, to page 5 location AAA. If the branch address specified was page 6 location BBB, then the resulting branch will be to page 6 location BBB. If a program resides in any page other than page Ø and a branch to page Ø is desired, the V bit must be inactive.

An exit or exit and branch operation will restore the control bits to the value associated with that stack level. A stack and branch operation will not affect the control bits.

TIMING: 4 Microseconds.

EXAMPLE:

```
PPP-LLL: MP1-MP2-MP3-MP4. E SEQ. NO. LAB: VERB  OPERANDS        COMMENTS

P15-ØØ4: 151-3ØØ.           Ø8-14Ø.       SMC;  C#3.           SET U,V CONTROL
P15-ØØ6: 23Ø-2ØØ.           Ø8-15Ø.       STA,  R#Ø; 128.      STORE PAGE 1Ø
P15-Ø1Ø: 1ØØ-1ØØ.           Ø8-16Ø.       BRU,  PØØ; Ø64.      BRANCH TO
                           *08-17Ø.                            PAGE 15-1ØØ/OCT
```

*DPL-1 CLASS 1 SET MEMORY SECTION & CONTROL **SSC**

OBJECT	SOURCE
152-YXØ	SSC, S#X; C#Y

WHERE: X is the section designation
Y is the control bit designation

DESCRIPTION:

This instruction performs both the set memory section operation of SMS and set memory control operation of SMC.

TIMING: 4 Microseconds.

EXAMPLE:

```
PPP-LLL: MP1-MP2-MP3-MP4. E SEQ. NO. LAB: VERB  OPERANDS        COMMENTS

P15-Ø12: 152-Ø3Ø.           Ø9-Ø3Ø.       SSC,  S#3; C#Ø.      SET SECT. 3 &
                            Ø9-Ø4Ø.                            RESET U & V-BIT
P15-Ø14: 152-36Ø            Ø9-Ø5Ø.       SSC,  S#6; C#3.      SET SECT. 6 &
                           *Ø9-Ø6Ø.                            SET U & V-BITS
```

SAC

OBJECT	SOURCE
153-ØØØ	SAC; ØØØ.

DESCRIPTION:

This instruction will force the arithmetical condition registers of the processor to an equal, high or low condition, dependent upon the state of bits 4 and 5 of the accumulator at the time of the SAC instruction.

The condition forced by the SAC instruction for a given state of bits 4 and 5 of the ACC is given below.

ACCUMULATOR BITS 7 6 5 4 3 2 1 Ø	CONDITION FORCED
1 Ø	Equal
Ø 1	High
Ø Ø	Low
1 1	Equal

TIMING: 4 Microseconds.

EXAMPLE:

```
PPP-LLL: MP1-MP2-MP3-MP4. E SEQ. NO. LAB: VERB  OPERANDS        COMMENTS

P15-Ø16: 2ØØ-Ø2Ø.          Ø9-12Ø.       LDA,  R#Ø; OCT:Ø2Ø.  LOAD COND. CODE
P15-Ø2Ø: 153-ØØØ.          Ø9-13Ø.       SAC;  ØØØ.           SET COND. HIGH
```

DPL-1 CLASS 1 LOAD SENSE SWITCHES

LSW

OBJECT	SOURCE
154-ØØØ	LSW; ØØØ.

DESCRIPTION:

The load sense switch instruction will load the state of 8 toggle switches (located in the switch well under the CRT screen) to the accumulator. Switch Ø is loaded to ACC Bit Ø, switch 1 to ACC Bit 1, etc.

TIMING: 4 Microseconds.

EXAMPLE:

```
PPP-LLL: MP1-MP2-MP3-MP4. E SEQ. NO. LAB: VERB  OPERANDS        COMMENTS

P15-Ø22: 154-ØØØ.           Ø9-19Ø.       LSW;  ØØØ.            LOAD S.S.
P15-Ø24: Ø44-ØØ4.           Ø9-2ØØ.       TMJ,  +Ø4; OCT:ØØ4.   TEST SW. #2
```

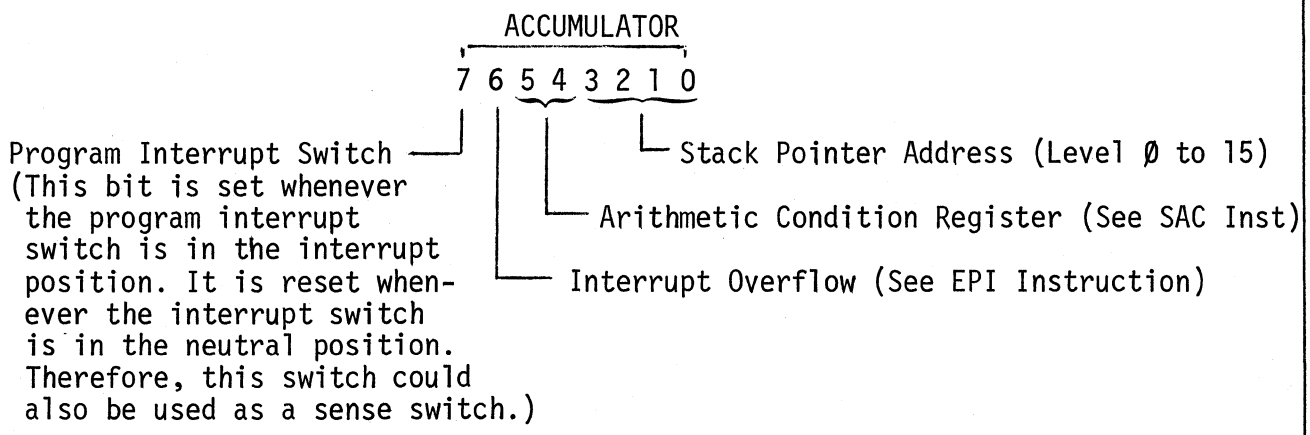
DPL-1 CLASS 1 LOAD PROCESSOR STATUS

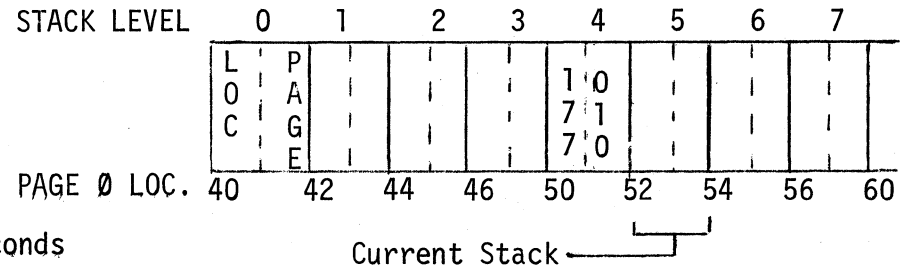
LPS

OBJECT	SOURCE
155-ØØØ	LPS; ØØØ.

DESCRIPTION

This instruction loads a processor status word into the accumulator. The following shows the accumulator bits and their respective meaning.





TIMING; 4 microseconds

EXAMPLE:

```
PPP-LLL: MP1-MP2-MP3-MP4. E SEQ. NO. LAB: VERB  OPERANDS        COMMENTS

                          *1Ø-Ø5Ø.  CALCULATE ADD.OF LAST STACK & BRANCH INST.
P15-Ø26: 155-ØØØ.          1Ø-Ø6Ø.       LPS;  ØØØ.            CALCULATE ADDR.
P15-Ø3Ø: 3Ø7-Ø36.          1Ø-Ø7Ø.       SAN,  S#7; OCT:Ø36.   OF CURRENT
P15-Ø32: 36Ø-Ø4Ø.          1Ø-Ø8Ø.       IRA,  R#Ø; OCT:Ø4Ø.   STACK (LOC.)
P15-Ø34: 23Ø-ØØ1           1Ø-Ø9Ø.       STA,  R#Ø; R#1.       SAVE IN R#1.
P15-Ø36: 261-ØØ2.          1Ø-1ØØ.       SUX,  R#1; OCT:ØØ2.   PREVIOUS STK LV
P15-Ø4Ø: 211-ØØ2.          1Ø-11Ø.       LDA,  I#1; PØØ.       RETRIEVE LOC.
P15-Ø42: 3ØØ-376.          1Ø-12Ø.       ANA,  R#Ø; OCT:376.   FROM PRIOR STK
P15-Ø44: 23Ø-ØØ2.          1Ø-13Ø.       STA,  R#Ø; R#2.       REMOVE LO BIT
P15-Ø46: 211-ØØØ.          1Ø-14Ø.       LDA,  R#1; PØØ.       RETRIEVE PAGE
P15-Ø5Ø: 3Ø6-377.          1Ø-15Ø.       SAN,  S#6; OCT:377.   OCTAL FORM-SHFT
P15-Ø52: 23Ø-ØØ3.          1Ø-16Ø.       STA,  R#Ø; R#3.       FOR DPL PG R-FM
```

DPL-1 CLASS 1 DISABLE PROCESSOR INTERRUPT

DPI

OBJECT	SOURCE
156-ØØØ	DPI; ØØØ.

DESCRIPTION

The disable processor interrupt instruction will inhibit the auto stack and branch that occurs upon receipt of an interrupt. It does not disable the interrupt.

If the Disable Processor Interrupt (DPI) was executed prior to the Enable Processor Interrupt (EPI) Instruction, and the interrupt is activated, the interrupt will occur only immediately following the execution of an EPI instruction.

TIMING: 4 Microseconds.

OBJECT	SOURCE
156-ØØ1	EPI; ØØØ.

DESCRIPTION:

Within the SYSTEM 4 hardware structure, there are two mechanisms which provide for the interrupting of normal instruction processing.

One is designed primarily for use by the programmer. The Interrupt Switch is located in the Switch Well under the CRT screen and can be activated by pushing the PROGRAM LOAD Toggle Switch in the opposite direction of the PROGRAM LOAD. Another interrupt cannot be generated until the switch is moved to the off position and then on again. Also an external interrupt will not be effective unless the interrupt switch is off. Two levels of interrupts may be preserved by the interrupt logic prior to the execution of an Enable Processor Interrupt (EPI) Instruction. Additional interrupts will cause the interrupt overflow indicator to be set (see 'LPS' Instruction).

An external interrupt input is provided for use by external devices. This is activated by a pulse from an external device.

For each interrupt (external or program) an EPI Instruction must be previously executed. After execution, that level of interrupt will be reset. If the interrupt is activated after the execution of the Enable Processor Interrupt (EPI) Instruction, an automatic Stack and Branch operation to Section 0, Page 3, Location 000 will occur. Therefore, the user program to process the interrupt condition must be initialized in Page 03-000. The return to the point of interrupt can be effected by an Exit (EXU) instruction to that Stack level. (Refer to Stack and Branch Unconditional).

NOTE: If the Disable Processor Interrupt (DPI) was not executed prior to the Enable Processor Interrupt (EPI) Instruction, and the interrupt is activated, the interrupt will occur after the execution of the current instruction.

INTERRUPT LOCK OUT

An interrupt will not occur during any of the following conditions. However, they do not inhibit the interrupt, they simply delay the auto Stack and Branch until the condition is completed. These conditions are:

1) Tape movement (Read/Write/Rewind/Search)

INTERRUPT LOCK OUT (cont'd.)

2) If the keyboard is made ready (i.e. by depressing a key and a Read from the keyboard has not been executed). In this case, the interrupt will occur immediately following the keyboard Read Instruction.

3) During a Set Section or Set Section and Control operation.

4) Interrupt Disable has been set (see Disable Processor Interrupt (DPI) Instruction).

TIMING: 4 Microseconds.

CPI

OBJECT	SOURCE
156-ØØ2	CPI; ØØØ.

DESCRIPTION:

An interrupt overflow condition occurs when more than two interrupts have been activated before the execution of an Enable Processor Interrupt (EPI) instruction. This condition may be tested through the use of the Load Processor Status (LPS) Instruction.

By execution of a Clear Interrupt instruction, the interrupt overflow indicator will be cleared.

TIMING: 4 Microseconds.

LDA

OBJECT	SOURCE
2ØØ-LLL	LDA, R#Ø; Literal.
21Ø-YXX	LDA, R#Ø; AAA±NNN.
21I-YXZØØ	LDA, R#X; PPP.
21I-YXZ1Ø	LDA, I#X; PPP.
21I-YXZ11	LDA, D#X; PPP.
21I-YXZØØ	LDA, R#X; AAA±NNN.

WHERE: 2ØØ is an immediate address command.
AND: LLL is any form of literal notation.
AND: 21Ø is a direct address command.
AND: YXX is an 8-bit location address within a level zero page.
AND: 21I is an indexed address command.
AND: I is any index (1-7).
AND: YXZ is a 6-bit base page address.

WHERE: R#Ø is the immediate or direct indicator.
AND: AAA is a direct address page within level #Ø.
AND: NNN is a decimal byte displacement.
AND: X is any register (1-7).
AND: PPP is a decimal page notation.
AND: I-Increment register by ØØ1 after execution.
AND: D-Decrement register by ØØ1 after execution.
AND: R-Register value unchanged after execution.

DESCRIPTION:

Load the Accumulator with the value specified by the immediate, the direct or the indexed address. The immediate form of this instruction provides the means of specifying machine values or bit congigurations as part of the instruction. Literal terms may be used to specify such program elements as immediate data, masks, and addresses. The direct form of this instruction allows the user to directly address any level zero page. By supplying the base page as the operand and by specifying the index register containing the address displacement within that page, the user can address any location within the memory.

The condition register value remains unchanged after execution of this instruction.

TIMING: 4 Microseconds when literal form is used.
6 Microseconds when literal form is not used.

EXAMPLE

```
PPP-LLL: MP1-MP2-MP3-MP4. E SEQ. NO. LAB: VERB  OPERANDS         COMMENTS

P15-Ø54: 2ØØ-2ØØ.           11-Ø3Ø.       LDA,  R#Ø; OCT:2ØØ.   IMMEDIATE
P15-Ø56: 2ØØ-Ø66.           11-Ø4Ø.       LDA,  R#Ø; ADL:AAA.   ADDRESSING
P15-Ø6Ø: 2ØØ-Ø64.           11-Ø5Ø.       LDA,  R#Ø; ADP:AAA.
                           *11-Ø6Ø.
P15-Ø62: 21Ø-2ØØ.           11-Ø7Ø.       LDA,  R#Ø; 128.       DIRECT
P15-Ø64: 21Ø-ØØ7.           11-Ø8Ø.       LDA,  R#Ø; R#7.       ADDRESSING
                           *11-Ø9Ø.
P15-Ø66: 213-Ø4Ø.           11-1ØØ.  AAA: LDA,  R#3; PØ8.       INDEXED
P15-Ø7Ø: 213-Ø42.           11-11Ø.       LDA,  I#3; PØ8.       ADDRESSING
P15-Ø72: 213-Ø43.           11-12Ø.       LDA,  D#3; PØ8.
```

.DPL-1 CLASS 2: DATA TRANSFER LOAD INDEX REGISTER

LDX

OBJECT	SOURCE
2ØI-LLL	LDX, R#X; Literal.

WHERE: I is any register (1-7).
AND: LLL is any form of literal notation.

DESCRIPTION:

Load the specified index register with the value indicated by the immediate address. The primary use of this instruction is to establish the address displacement, within a page, for the indexed addressing instructions. Any form of literal notation may be used.

The condition register value remains unchanged after execution of this instruction.

TIMING: 4 Microseconds

EXAMPLE:

```
PPP-LLL: MP1-MP2-MP3-MP4. E SEQ. NO. LAB: VERB  OPERANDS        COMMENTS

P15-Ø74: 2Ø1-Ø17.          11-18Ø.       LDX,  R#1; OCT:Ø17.
P15-Ø76: 2Ø2-Ø17.          11-19Ø.       LDX,  R#2; DEC:Ø15.
P15-1ØØ: 2Ø3-Ø17.          11-2ØØ.       LDX,  R#3; HEX:ØF.
P15-1Ø2: 2Ø4-Ø17.          12-Ø1Ø.       LDX,  R#4; (C).
P15-1Ø4: 2Ø5-11Ø.          12-Ø2Ø.       LDX,  R#5; ADL:BBB.
P15-1Ø6: 2Ø6-1Ø2.          12-Ø3Ø.       LDX,  R#6; BBB-Ø6.
P15-11Ø: 2Ø7-Ø66.          12-Ø4Ø.  BBB: LDX,  R#7; IDP:BBB.
```

·DPL-1 CLASS 2. LOAD INSTRUCTION ADDRESS **LIA**

OBJECT	SOURCE
22I-ØØØ	LIA, R#X; ØØØ.
22I-LLL	LIA, R#X; Literal.

WHERE: I is any register (1-7)
AND: LLL is any form of Literal Notation.

DESCRIPTION:

This instruction will transfer the 8 least significant bits of the current instruction address to the specified index register. If the instruction literal is ØØØ, then the section and page of the current instruction address is transferred to the accumulator. If the literal is not ØØØ, then the literal is transferred to the accumulator.

TIMING: 4 Microseconds

EXAMPLE:

```
PPP-LLL: MP1-MP2-MP3-MP4. E SEQ. NO. LAB: VERB  OPERANDS        COMMENTS

                          *12-1ØØ.  LOAD INSTRUCTION ADDRESS (MAIN ROUTINE)
                          *12-11Ø.  MONITOR, FIXED, NON-RELOCATABLE.
                          *12-12Ø.
P15-112: 151-2ØØ.          12-13Ø.       SMC;  C#2.            SET CONTROL
P15-114: 2Ø1-123.          12-14Ø.  MTR: LDX,  R#1; MTR+Ø7.    SET POINTER
P15-116: 231-Ø64.          12-15Ø.       STA,  R#1; MTR.       (ACCUM CONTAINS
                          *12-16Ø.                            PAGE NUMBER)
P15-12Ø: 21Ø-ØØ6.          12-17Ø.       LDA,  R#Ø; R#6.       DATA PAGE
P15-122: 237-ØØØ.          12-18Ø.       STA,  R#7; PØØ.       INSERT PAGE
P15-124: 14Ø-ØØØ.          12-19Ø.       EXU;  ØØØ.
                          *12-2ØØ.
                          *13-Ø1Ø.  RELOCATABLE SUBROUTINE.
                           13-Ø2Ø.       ORG:  P14, ØØØ.       ACM.CONTAINS PG
P16-ØØØ: 227-ØØØ.          13-Ø3Ø.  SUB: LIA,  R#7; ØØØ.       R#(-ADDR + 1
P16-ØØ2: 247-ØØ6.          13-Ø4Ø.       ADX,  R#7; OCT:ØØ6.   ADJUST
P16-ØØ4: 125-114.          13-Ø5Ø.       SBU;  MTR.
P16-ØØ6: 354-ØØ2.          13-Ø6Ø.  SB1: CPA,  I#4; PØØ.       PAGE CHANGED
                          *13-Ø7Ø.                            AT OBJECT-TIME
```

·DPL-1 CLASS 2: DATA TRANSFER STORE ACCUMULATOR

STA

OBJECT	SOURCE
23Ø-YXX	STA, R#Ø; AAA+NNN.
23I-YXZØØ	STA, R#X; PPP.
23I-YXZ1Ø	STA, I#X; PPP.
23I-YXZ11	STA, D#X; PPP.
23I-YXZØØ	STA, R#X; AAA±NNN.

WHERE: 23Ø is a direct address command.
AND: YXX is an 8-bit location address within a level zero page.
AND: 23I is an indexed address command.
AND: I is any index (1-7).
AND: YXZ is a 6 bit base page address.

WHERE: AAA is a direct address page within level #Ø.
AND: NNN is a decimal byte displacement.
AND: X is any register (1-7).
AND: PPP is a decimal page notation.
AND: I-Increment register by ØØ1 after execution.
AND: D-Decrement register by ØØ1 after execution.
AND: R-Register value unchanged after execution.

DESCRIPTION:

Store the contents of the Accumulator into the address specified by the direct or indexed address contained in the instruction operand. The direct form of this instruction allows the user to directly address any location within a level zero page. By supplying the base page address as the operand and by specifying the index register containing the address displacement within that page, the user can address any location within the memory, thus performing the function of indexed addressing.

The condition register value remains unchanged after execution of this instruction.

TIMING: 6 Microseconds.

EXAMPLE:

```
PPP-LLL: MP1-MP2-MP3-MP4. E SEQ. NO. LAB: VERB  OPERANDS        COMMENTS

P16-Ø1Ø: 23Ø-144.           13-14Ø.       STA,  R#Ø; 1ØØ.       DIRECT
P16-Ø12: 23Ø-ØØ7.           13-15Ø.       STA,  R#Ø; R#7.        ADDRESSING
                           *13-16Ø.
P16-Ø14: 231-Ø7Ø.           13-17Ø.       STA,  R#1; OCC.        INDEXED
P16-Ø16: 231-Ø37.           13-18Ø.       STA,  D#1; PØ7.        ADDRESSING
P16-Ø2Ø: 231-Ø36.           13-19Ø. CCC:  STA,  I#1; PØ7.
```

DPL-1 CLASS 2: ORDINARY ARITHMETIC ADD TO ACCUMULATOR

ADA

OBJECT	SOURCE
24Ø-LLL	ADA, R#Ø; Literal.
25Ø-YXX	ADA, R#Ø; AAA±NNN.
25I-YXZØØ	ADA, R#X; PPP.
25I-YXZ1Ø	ADA, I#X; PPP.
25I-YXZ11	ADA, D#X; PPP.

WHERE: 24Ø is an immediate address command.
AND; LLL is any form of literal notation.
AND: 25Ø is a direct address command.
AND: YXX is an 8-bit location address within a level zero page.
AND: 25I is an indexed address command.
AND: I is any index (1-7).
AND: YXZ is a 6-bit base page address.

WHERE: R#Ø is the immediate or direct indicator.
AND: AAA is a direct address page within level #Ø.
AND: NNN is a decimal byte displacement.
AND: X is any register (1-7).
AND: PPP is a decimal page notation.
AND: I-Increment register by ØØ1 after execution.
AND: D-Decrement register by ØØ1 after execution.
AND: R-Register value remains unchanged after execution.

DESCRIPTION:

Binary add to the Accumulator the value specified by the immediate, the direct, of the indexed address. The immediate form of this instruction provides the means of specifying machine values or bit configurations as part of the instruction. The direct form of this instruction allows the user to directly address any level zero page. By supplying the base page address as the operand and by specifying the index register containing the address displacement within that page, the user can address any location within the memory, thus performing the function of indexed addressing. In the event of an overflow condition, the overflow character is lost.

The condition register value remains unchanged after execution of this instruction.

TIMING: 4 Microseconds when literal form is used.
6 Microseconds when literal form is not used.

EXAMPLE:

```
PPP-LLL. MP1-MP2-MP3-MP4. E SEQ. NO. LAB: VERB  OPERANDS          COMMENTS

P16-Ø22: 24Ø-ØØ4.           14-Ø6Ø.  DDD: ADA,  R#Ø; (1).        ADD IMMEDIATE
P16-Ø24: 25Ø-ØØ7.           14-Ø7Ø.       ADA,  R#Ø; R#7.        ADD DIRECT
P16-Ø26: 253-ØØ4.           14-Ø8Ø.       ADA,  R#3; PØ1.        ADD INDEXED
P16-Ø3Ø: 253-Ø72.           14-Ø9Ø.       ADA,  I#3; DDD.
P16-Ø32: 253-ØØ7.           14-1ØØ.       ADA,  D#3; PØ1.
```

CLASS 2: ORDINARY ARITHMETIC

ADD TO INDEX REGISTER

ADX

OBJECT	SOURCE
24I-LLL	ADX, R#X; Literal.

WHERE: I is any register (1-7)
AND: LLL is any form of literal notation.

DESCRIPTION:

Binary add to the index register specified, the value indicated by the immediate address. In the event of any overflow condition, the overflow character is lost.

The condition register value remains unchanged after execution of this instruction.

TIMING: 4 Microseconds.

EXAMPLE:

```
PPP-LLL: MP1-MP2-MP3-MP4. E SEQ. NO. LAB: VERB  OPERANDS        COMMENTS

P16-Ø34: 241-ØØ4.           14-16Ø.  EEE: ADX,  R#1; (1).       BINARY ADD
P16-Ø36: 242-ØØ4.           14-17Ø.       ADX,  R#2; OCT:ØØ4.   TO INDEX
P16-Ø4Ø: 243-372.           14-18Ø.       ADX,  R#3; HEX:FA.    REGISTER
```

DPL-1 CLASS 2: ORDINARY ARITHMETIC SUBTRACT FROM ACCUMULATOR

SUA

OBJECT	SOURCE
26Ø-LLL	SUA, R#Ø; Literal.
27Ø-YXX	SUA, R#Ø; AAA+NNN.
27I-YXZØØ	SUA, R#X; PPP.
27I-YXZ1Ø	SUA, I#X; PPP.
27I-YXZ11	SUA, D#X; PPP.

WHERE: 26Ø is an immediate address command.
AND: LLL is any form of literal notation.
AND: 27Ø is a direct address command.
AND: YXX is an 8-bit location address within a level zero page.
AND: 27I is an indexed address command.
AND: I is any index (1-7)
AND: YXZ is a 6-bit base page address.

WHERE: R#Ø is the immediate or direct indicator.
AND: AAA is a direct address page within level #Ø.
AND: NNN is a decimal byte displacement.
AND: X is any register (1-7)
AND: PPP is a decimal page notation.
AND: I-Increment register by ØØ1 after execution.
AND: D-Decrement register by ØØ1 after execution.
AND: R-Register value unchanged after execution.

DESCRIPTION:

Binary subtract from the Accumulator, the value specified by the immediate, the direct or the indexed address. The immediate form of the instruction provides the means of specifying machine values or bit configuration as part of the instruction. The direct form of this instruction allows the user to directly address any location within a level zero page. By supplying the base page address as the operand and by specifying the index register containing the address displacement within that page, the user can address any location within the memory, thus performing the function of indexed addressing. The two's complement results from an underflow condition. The condition register value remains unchanged after execution of this instruction.

TIMING: 4 Microseconds when literal form is used.
6 Microseconds when literal form is not used.

EXAMPLE:

```
PPP-LLL: MP1-MP2-MP3-MP4. E SEQ. NO. LAB: VERB  OPERANDS        COMMENTS

P16-Ø42: 26Ø-ØØ4.           15-Ø4Ø.       SUA,  R#Ø; (1)       SUB. IMMED.
P16-Ø44: 27Ø-ØØ7.           15-Ø5Ø.  FFF: SUA,  R#7.           SUB. DIRECT
P16-Ø46: 273-ØØ4.           15-Ø6Ø.       SUA,  R#3; PØ1       SUB. INDEXED
P16-Ø5Ø: 273-Ø7Ø.           15-Ø7Ø.       SUA,  R#3; FFF.
```

·DPL-1 CLASS 2: ORDINARY ARITHMETIC SUBTRACT FROM INDEX REGISTER

SUX

OBJECT	SOURCE
26I-LLL	SUX, R#X; Literal.

WHERE: I is any register (1-7).
AND: LLL is any form of literal notation.

DESCRIPTION:

Binary subtract from the index register specified, the value indicated by the immediate address. The two's complement results from an underflow condition. The condition register value remains unchanged after execution of this instruction.

TIMING: 4 Microseconds.

EXAMPLE:

```
PPP-LLL: MP1-MP2-MP3-MP4. E SEQ. NO. LAB: VERB  OPERANDS        COMMENTS

P16-Ø52: 261-ØØ4.           15-13Ø.  GGG: SUX,  R#1; (1).       BINARY
P16-Ø54: 262-ØØ4.           15-14Ø.       SUX,  R#2; OCT:ØØ4.   SUBTRACT FROM
P16-Ø56: 262-Ø17.           15-15Ø.       SUX,  R#2; HEX:ØF.    INDEX REG.
P16-Ø6Ø. 263-144.           15-16Ø.       SUX,  R#3; DEC:1ØØ.
```

·DPL-1 CLASS 3: BOOLEAN ARITHMETIC

LOGICAL "AND" TO ACCUMULATOR

ANA

OBJECT	SOURCE
3ØØ-LLL	ANA, R#Ø; Literal.
31Ø-YXX	ANA, R#Ø; AAA+NNN.
31I-YXZØØ	ANA, R#X; PPP.
31I-YXZ1Ø	ANA, I#X; PPP.
31I-YXZ11	ANA, D#X; PPP.

WHERE: 3ØØ is an immediate address command.
AND: LLL is any form of literal notation.
AND: 31Ø is a direct address command.
AND: YXX is an 8-bit location address within a level zero page.
AND: 31I is an indexed address command.
AND: I is an index (1-7).
AND: YXZ is a 6-bit base page address.

WHERE: R#Ø is the immediate or direct indicator.
AND: AAA is a direct address page within level #Ø.
AND: NNN is a decimal byte displacement.
AND: X is any register (1-7).
AND: PPP is a decimal page notation.
AND: I-Increment register by ØØ1 after execution.
AND: D-Decrement register by ØØ1 after execution.
AND: R-Register value remains unchanged after execution.

DESCRIPTION:

Logical "and" to the Accumulator the value specified by the immediate, direct or indexed address. The immediate form of this instruction provides the means of specifying machine values of bit configurations as part of the instruction. The direct form of this instruction allows the user to directly address any level zero page. By supplying the base page address as the operand and by specifying the index register containing the address displacement within that page, the user can address any location within the memory, thus performing the function of indexed addressing. The value of the operand is treated as an unstructured logical quantity, and the value is applied bit by bit to the Accumumulator. The bit position in the result (Accumulator) is set to one if the corresponding bit positions in the Accumulator and the operand both contain a one; otherwise, the result bit is set to zero. (Result is one if both are ones). All operand values and results are valid. The condition register value remains unchanged after execution of this instruction.

TIMING: 4 Microseconds when literal form is used.
6 Microseconds when literal form is not used.

EXAMPLE:

```
PPP-LLL: MP1-MP2-MP3-MP4. E SEQ. NO. LAB: VERB  OPERANDS        COMMENTS

                          *16-Ø2Ø.  LOGICAL "AND" TO ACCUMULATOR.
P16-Ø62: 3ØØ-Ø26.          16-Ø3Ø.       ANA,  R#Ø; OCT:Ø26.   IMMEDIATE
P16-Ø62: 31Ø-ØØ7.          16-Ø4Ø.       ANA,  R#Ø; R#7.       DIRECT
P16-Ø66: 314-Ø3Ø.          16-Ø5Ø.       ANA,  R#4; PØ6.       INDEXED
P16-Ø7Ø: 314-Ø32.          16-Ø6Ø.       ANA,  I#4; PØ6.       ADDRESSING
P16-Ø72: 314-Ø33.          16-Ø7Ø.       ANA,  D#4; PØ6.
```

·DPL-1 CLASS 3: BOOLEAN ARITHMETIC

SHIFT & LOGICAL "AND" TO ACCUMULATOR

SAN

OBJECT	SOURCE
3ØI-LLL	SAN, S#X; Literal.

WHERE: I is the bit shift count (1-7). WHERE: X is the bit shift count (1-7).
AND: LLL is any literal notation.

DESCRIPTION:

This form of the logical "and" instruction performs a right circular shift of the bits in the Accumulator, by the number of bits specified in the shift counter, before the logical "and" of the literal to the Accumulator is performed.

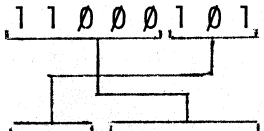
All literal values and results are valid. The condition register value remains unchanged after the execution of this instruction.

TIMING: 4 Microseconds.

EXAMPLE:

```
PPP-LLL: MP1-MP2-MP3-MP4. E SEQ. NO. LAB: VERB  OPERANDS        COMMENTS

                           *16-12Ø.  SHIFT AND LOGICAL "AND" TO ACCUMULATOR.
P16-Ø74: 3Ø3-Ø26.           16-13Ø.       SAN,  S#3; OCT:Ø26.
```



```
1 1 Ø Ø Ø 1 Ø 1     Initial Accumulator Value

1 Ø 1 1 1 Ø Ø Ø     Accumulator Value after a Shift of 3.
Ø Ø Ø 1 Ø 1 1 Ø     Literal Value
Ø Ø Ø 1 Ø Ø Ø Ø     Accumulator Value after the logical "AND" of OCT:Ø26.
```

DPL-1 CLASS 3: BOOLEAN ARITHMETIC — EXCLUSIVE "OR" TO ACCUMULATOR

ERA

OBJECT	SOURCE
32Ø-LLL	ERA, R#Ø; Literal.
33Ø-YXX	ERA, R#Ø; AAA±NNN.
33I-YXZØØ	ERA, R#X; PPP.
33I-YXZ1Ø	ERA, I#X; PPP.
33I-YXZ11	ERA, D#X; PPP.

WHERE: 32Ø is an immediate address command.
AND: LLL is any form of literal notation.
AND: 33Ø is a direct address command.
AND: YXX is an 8-bit location address within a level zero page.
AND: 33I is an indexed address command.
AND: I is an index (1-7).
AND: YXZ is a 6-bit base page address.

WHERE: R#Ø is the immediate or direct indicator.
AND: AAA is a direct address page within level zero.
AND: NNN is a decimal byte displacement
AND: X is any register (1-7).
AND: PPP is a decimal page notation.
AND: I-Increment register by ØØ1 after execution.
AND: D-Decrement register by ØØ1 after execution.
AND: R-Register value remains unchanged after execution.

DESCRIPTION:

Exclusive "or" to the Accumulator the value specified by the immediate, direct or indexed address. The immediate form of this instruction provides the means of specifying machine values or bit configurations as part of the instruction. The direct form of this instruction allows the user to directly address any level zero page. By supplying the base page address as the operand and by specifying the index register containing the address displacement within that page, the user can address any location within the memory. The value of the operand is treated as an unstructured logical quantity, and the value is applied bit by bit to the Accumulator. A bit position in the result (Accumulator) is set to one if the corresponding bit positions in the accumulator and as specified by the operand, are unlike; otherwise, the result bit is set to zero. (Result is one if unlike). All operand values and results are valid. The condition register value remains unchanged after execution of this instruction.

TIMING: 4 Microseconds when literal form is used.
6 Microseconds when literal form is not used.

EXAMPLE:

```
PPP-LLL: MP1-MP2-MP3-MP4. E SEQ. NO. LAB: VERB  OPERANDS        COMMENTS

                          *17-Ø6Ø.  EXCLUSIVE "OR" TO ACCUMULATOR
P16-Ø76: 32Ø-Ø26.          17-Ø7Ø.       ERA,  R#Ø; OCT:Ø26.   IMMEDIATE
P16-1ØØ: 33Ø-ØØ7.          17-Ø8Ø.       ERA,  R#Ø; R#7.       DIRECT
P16-1Ø2: 334-Ø3Ø.          17-Ø9Ø.       ERA,  R#4; PØ6.       INDEXED
P16-1Ø4: 334-Ø32.          17-1ØØ.       ERA,  I#4; PØ6.       ADDRESSING
P16-1Ø6: 334-Ø33.          17-11Ø.       ERA,  D#4: PØ6.
```

·DPL-1 CLASS 3: BOOLEANE ARITHMETIC SHIFT AND "EOR" ACCUMULATOR

SER

OBJECT	SOURCE
32I-LLL	SER, S#X; Literal.

WHERE: I is the bit shift count (1-7). WHERE: X is the bit shift count (1-7)
AND: LLL is any literal notation.

DESCRIPTION:

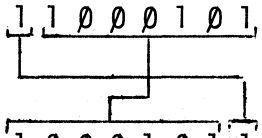
This form of the exclusive "or" instruction performs a right circular shift of the bits in the Accumulator, by the number of bits specified in the shift counter, before the exclusive "or" of the literal to the Accumulator is performed. All literal values and results are valid. The condition register value remains unchanged after execution of this instruction.

TIMING: 4 Microseconds

EXAMPLE:

```
PPP-LLL: MP1-MP2-MP3-MP4. E SEQ. NO. LAB: VERB  OPERANDS        COMMENTS

                          *17-16Ø.  SHIFT AND "EOR" ACCUMULATOR
P16-11Ø: 327-Ø26.          17-17Ø.       SER,  S#7; OCT:Ø26.
```



```
1 1 Ø Ø Ø 1 Ø 1     Initial Accumulator Value

1 Ø Ø Ø 1 Ø 1 1     Accumulator Value after a Shift of 7.
Ø Ø Ø 1 Ø 1 1 Ø     Literal Value
1 Ø Ø 1 1 1 Ø 1     Accumulator Value after the exclusive "OR" of OCT:Ø26.
```

DPL-1 CLASS 3: INCLUSIVE "OR" TO ACCUMULATOR
BOOLEAN ARITHMETIC

IRA

OBJECT	SOURCE
36Ø-LLL	IRA, R#Ø; Literal.
37Ø-YXX	IRA, R#Ø; NNN.
37Ø-YXX	IRA, R#Ø; AAA±NNN.
37I-YXZØØ	IRA, R#X; PPP.
37I-YXZ1Ø	IRA, I#X; PPP.
37I-YXZ11	IRA, D#X; PPP.

WHERE: 36Ø is an immediate address command.
AND: LLL is any form of literal notation.
AND: 37Ø is a direct address command.
AND: YXX is an 8-bit location address within a level zero page.
AND: 37I is an indexed address command.
AND: I is an index (1-7).
AND: YXZ is a 6-bit base page address.

WHERE: R#Ø is the immediate or direct indicator.
AND: AAA is a direct address page within level zero.
AND: NNN is a decimal byte displacement
AND: X is any register (1-7).
AND: PPP is a decimal page notation.
AND: I-Increment register by ØØ1 after execution.
AND: D-Decrement register by ØØ1 after execution.
AND: R-Register value remaining unchanged after execution.

DESCRIPTION:

Inclusive "or" to the accumulator the value specified by the immediate, direct, or indexed address. The immediate form of this instruction provides the means of specifying machine values or bit configurations as part of the instruction. The direct form of this instruction allows the user to directly address any level zero page. By supplying the base page address as the operand and by specifying the index register containing the address displacement within that page, the user can address any location within the memory. The value of the operand is treated as an unstructured logical quantity, and the value is applied bit by bit to the accumulator. A bit position in the result (accumulator) is set to one if the corresponding bit position in the accumulator or as specified by the operand, either contain a one; otherwise, the result bit is set to zero. (Result is one if either are one). All operand values and results are valid. The condition register value remains unchanged after execution of this instruction.

TIMING: 4 Microseconds when literal form is used.
6 Microseconds when literal form is not used.

EXAMPLE:

```
PPP-LLL: MP1-MP2-MP3-MP4. E SEQ. NO. LAB: VERB  OPERANDS        COMMENTS

                          *18-1ØØ.  INCLUSIVE "OR" TO ACCUMULATOR.
P16-112: 36Ø-Ø26.          18-11Ø.       IRA,  R#Ø; OCT:Ø26.   IMMEDIATE
P16-114: 37Ø-ØØ7.          18-12Ø.       IRA,  R#Ø; R#7.       DIRECT
P16-116: 374-Ø3Ø.          18-13Ø.       IRA,  R#4; PØ6.       INDEXED
P16-12Ø: 374-Ø32.          18-14Ø.       IRA,  I#4; PØ6.       ADDRESSING
P16-122: 374-Ø33.          18-15Ø.       IRA,  D#4; PØ6.
```

SIR

OBJECT	SOURCE
36I-LLL	SIR, S#X; Literal.

WHERE: I is the bit shift count.
AND: LLL is any literal notation.

WHERE: X is the bit shift count.

DESCRIPTION:

This form of the inclusive "or" instruction performs a right circular shift of the bits in the Accumulator, by the number of bits specified in the shift counter, before the inclusive "or" of the literal to the Accumulator is performed. All literal values and results are valid. The condition register value remains unchanged after execution of this instruction.

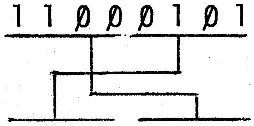
TIMING: 4 Microseconds.

EXAMPLE:

```
PPP-LLL: MP1-MP2-MP3-MP4. E SEQ. NO. LAB: VERB  OPERANDS        COMMENTS

                             *18-2ØØ.  SHIFT AND INCLUSIVE "OR" TO ACCUMULATOR.
P16-124: 364-Ø26.             19-Ø1Ø.       SIR,  S#4; OCT:Ø26.
```

1 1 Ø Ø Ø 1 Ø 1 Initial Accumulator Value



Ø 1 Ø 1 1 1 Ø Ø Accumulator Value after a Shift of 4
Ø Ø Ø 1 Ø 1 1 Ø Literal Value
Ø 1 Ø 1 1 1 1 Ø Accumulator Value after the inclusive "OR" of OCT:Ø26.

DPL-1 CLASS 3: COMPARE COMPARE ACCUMULATOR

CPA

OBJECT	SOURCE
34Ø-LLL	CPA, R#Ø; Literal.
35Ø-YXX	CPA, R#Ø; AAA±NNN.
35I-YXZØØ	CPA, R#X; PPP.
35I-YXZ1Ø	CPA, I#X; PPP.
35I-YXZ11	CPA, D#X; PPP.

WHERE: 34Ø is an immediate address command.
AND: LLL is any form of literal notation.
AND: 35Ø is a direct address command.
AND: YXX is an 8-bit location address within a level zero page.
AND: 35I is an indexed address command.
AND: I is any index (1-7).
AND: YXZ is a 6-bit base page address.

WHERE: R#Ø is the immediate or direct indicator.
AND: AAA is a direct address page within level zero.
AND: NNN is a decimal byte displacement
AND: X is any register (1-7).
AND: PPP is a decimal page notation.
AND: I-Increment register by ØØ1 after execution.
AND: D-Decrement register by ØØ1 after execution.
AND: R-Register value unchanged after execution.

DESCRIPTION:

Compare the contents of the Accumulator to the value specified by the immediate, direct or indexed address. The immediate form of this instruction provides the means of specifying machine values or bit configurations as part of the instruction. The direct form of this instruction allows the user to directly address any level zero page. By supplying the base page address as the operand and by specifying the index register containing the address displacement within that page, the user can address any location within the memory. The character in the Accumulator is not altered. The condition register value is changed to reflect the high, low or equal result of the compare instruction. Once set, the condition register remains unchanged until modified by an instruction that reflects a different condition code.

TIMING: 4 Microseconds when literal form is used.
6 Microseconds when literal form is not used.

EXAMPLE:

```
PPP-LLL: MP1-MP2-MP3-MP4. E SEQ. NO. LAB: VERB  OPERANDS        COMMENTS

P16-126: 34Ø-Ø26.          19-15Ø.       CPA,  R#Ø; (J).       IMMEDIATE
P16-13Ø: 35Ø-ØØ7.          19-16Ø.       CPA,  R#Ø; R#7.       DIRECT
P16-132: 34Ø-136.          19-17Ø.       CPA,  R#Ø; ADL:CPT.
P16-134: 354-Ø3Ø.          19-18Ø.       CPA,  R#4; PØ6.       INDEXED
P16-136: 354-Ø32.          19-19Ø.  CPT: CPA,  I#4; PØ6.       ADDRESSING
P16-14Ø: 354-Ø33.          19-2ØØ.       CPA,  D#4; PØ6.
```

CLASS 3: COMPARE — COMPARE INDEX REGISTER

CPX

OBJECT	SOURCE
34I-LLL	CPX, R#X; Literal.

WHERE: I is any index register (1-7). WHERE: X is any register (1-7).
AND: LLL is any literal notation.

DESCRIPTION:

Compare the contents of the index register specified to the byte of immediate data (literal). Comparison is binary, and all codes are valid. The value of the index register is not altered.

The condition register value is changed to reflect the high, low, or equal result of the compare instruction. Once set, the condition register remains unchanged until modified by an instruction that reflects a different condition code.

TIMING: 4 Microseconds.

EXAMPLE:

```
PPP-LLL: MP1-MP2-MP3-MP4. E SEQ. NO. LAB: VERB  OPERANDS        COMMENTS

P16-142: 343-ØØ6.          2Ø-Ø6Ø. CRT: CPX,  R#3; (3).       IMMEDIATE
P16-144: 343-146.          2Ø-Ø7Ø.      CPX,  R#3; CRT+Ø4.    ADDR. LOC.
```

GET

OBJECT	SOURCE
Ø17-YXX-ØXX-YXX	GET; NNN, T#N, MMM.
Ø17-YXX-2XX-YXX	GET; NNN, M#N, MMM.
Ø17-YXX-Ø12-YXX	GET; NNN, KBD, MMM.

WHERE: NNN is the decimal size.
AND: T#N, M#N, or KBD is the device number of the Mini-Tape, the Maxi-Tape, or the Keyboard, respectively.
AND: MMM is the left-most high order decimal address of the receiving field.

DESCRIPTION:

A Read operation is initiated at the I/O device, and the data is transferred from the device into memory. Page destination is initially set to 01 but may be changed with a SET PAGE Instruction. The Page remains at this setting until a different SET PAGE Instruction is executed. Any data continuing past a Page boundary will be wrapped around to the beginning of the Page.

MAXI-TAPE:

When retrieving records from Maxi-Tape, the same number of bytes as contained in the tape record must be specified by the size operand within the instruction. The size may be up to 256 bytes for Maxi-Tape. Data is placed in memory in ascending order of addresses within the "Into" Page which is currently set, starting with the address specified in the instruction.

MINI-TAPE:

When retrieving records from Mini-Tape, the physical record length must be 136 bytes. The standard Mini-Tape record is comprised of an 8-byte label, generated by the Mini-Write software function, followed by 128 bytes of data. Because the label is generated by the software and not by the user, it is not included in the record size operand. Therefore, when reading the standard Mini-Tape record, specify 128 (number of data bytes) as the size. Although a Mini-Tape record may contain a maximum of 128 bytes of data, it may be desirable to read a lesser number of characters into the input buffer. By specifying a lesser number in the size operand, only the number of characters specified will be stored into the I/O area indicated by the user. The remainder of the record will be read and used to check for tape errors and CRC Check but these characters will not be stored into the I/O Buffer. The 8-byte label is automatically read into Page 00 Locations 030_8 thru 037_8. The data portion of the record is placed in memory in ascending order of addresses within the "Into" Page which is currently set, starting with the address specified in the instruction. An automatic sequence check is made on the first byte of the label. If the record contains the wrong sequence number (i.e. a record was skipped), the error condition will be set.

KEYBOARD

When retrieving data from the keyboard, the same number of bytes as contained in the size operand must be entered. The keyboard Supervisor provides for corrections to be made to data entered. By depressing the "CORR" key on the keyboard, the point of entry will be backspaced one location within the current Page. The size operand may specify up to 256 bytes for the keyboard operation. Data is placed in memory in ascending order of addresses within the "Into" Page which is currently set, starting with the address specified in the instruction.

Device assignment is as follows:

DDD	DEVICE NAME	SYMBOLIC CODE	OBJECT CODE
Standard Pair 1	Mini-Tape 1 Mini-Tape 2	T#1 T#2	ØØ1 ØØ2
Optional Pair 2	Mini-Tape 3 Mini-Tape 4	T#3 T#4	ØØ3 ØØ4
Optional Pair 3	Mini-Tape 5 Mini-Tape 6	T#5 T#6	ØØ5 ØØ6
Optional Pair 4	Mini-Tape 7 Mini-Tape 8	T#7 T#8	ØØ7 Ø1Ø
	Keyboard	KBD	Ø12
	Maxi-Tapes	M#N	2ØX

EXAMPLE:

```
PPP-LLL: MP1-MP2-MP3-MP4. E SEQ. NO. LAB: VERB  OPERANDS          COMMENTS

P15-ØØØ: 15Ø-ØØØ-122-ØØ4.   Ø1-1ØØ.       ENT:  DPL-3.
P15-ØØ4: ØØ5-ØØ6-Ø12-Ø66.   Ø1-11Ø.       SET;  PAG:FØ1,TØ2.      SET PAGE
P15-Ø1Ø: Ø17-2ØØ-ØØ1-ØØØ.   Ø1-12Ø.       GET;  128,T#1,ØØØ.      READ INTO PG2
P15-Ø14: ØØ4-ØØ1-Ø17-ØØØ.   Ø1-13Ø.       SEL;  LOW,P15,ØØØ.      EOC RECORD?
P15-Ø2Ø: ØØ4-ØØ4-Ø17-144.   Ø1-14Ø.       SEL;  HGH,P15,1ØØ.      TAPE ERROR?
P15-Ø24: 15Ø-Ø1Ø-1Ø5-Ø3Ø.   Ø1-15Ø.       ENT:  DPL-1.            RECORD OK
P15-Ø3Ø: 14Ø-ØØØ.           Ø1-16Ø.       EXU;  ØØØ.              RETURN
```

PUT

OBJECT	SOURCE
Ø27-YXX-YXX-ØXX	PUT; NNN, MMM, T#N.
Ø27-YXX-YXX-2XX	PUT; NNN, MMM, M#N.
Ø27-YXX-YXX-Ø13	PUT; NNN, MMM, PRT.

WHERE: NNN is the decimal size.
AND: MMM is the left-most high order decimal address of the sending field.
AND: T#N, M#N, or PRT is the device number of the Mini-tape, the Maxi-tape, or the Printer; respectively.

DESCRIPTION:

A Write operation is initiated at the I/O device, and the data is transferred from memory to the device. The Page source is initially set to 01, but may be changed with a SET PAGE Instruction. The Page remains at this setting until a different SET PAGE Instruction is executed. Data can be written from any Location within a Page. Any data continuing past the Page boundaries, based on the size operand, will be wrapped around to the beginning of the Page.

MAXI-TAPE:

When writing records to Maxi-Tape, the actual number of bytes desired to be written must be specified by the size operand within the Instruction. The size may be up to 256 bytes for Maxi-Tape. Data in memory is fetched in an ascending order of addresses, within the "From" Page which is currently set, starting with the address specified in the Instruction.

MINI-TAPE:

When writing records to Mini-Tape, the physical record length will be 136 bytes. The standard Mini-Tape record is comprised of an 8-byte label, generated by the software function, followed by 128 bytes of data. Because the label is generated by the software and not by the user, it is not included in the record size operand. Therefore, when writing the Standard Mini-Tape record, specify 128 (number of data bytes) as the size. Although a Mini-Tape record may contain a maximum of 128 bytes of data, it may be desirable to write a lesser number of characters from the input buffer. By specifying a lesser number in the size operand, only the number of characters specified will be written from the I/O area indicated by the user. The remainder of the record will contain Octal zeroes. The complete 136 byte record will be used to check for tape write errors.

MINI-TAPE (cont'd.)

The 8-byte label is automatically written from Page 00 locations 030_8 thru 037_8. The record sequence number is automatically generated by the Mini-Write software function. The remainder of the label may be controlled by the user. The data portion of the record is written from memory in ascending order of addresses within the "From" Page which is currently set, starting with the address specified in the Instruction.

PRINTER:

When printing, the actual number of bytes desired to be printed must be specified by the size operand within the Instruction. The size may be up to 256 characters. Data in memory is fetched in an ascending order of addresses, within the "From" Page which is currently set, starting with the address specified in the Instruction.

In order to execute a control command for the Print Function as part of the data, the control byte must have the high-order bit present (Ref. I/O Instruction) (i.e. to execute an "Index Function for the typewriter, either execute PCL; ØØ1, PRT, IDX, or Place OCTAL 212 as a character in the print buffer).

There will be an automatic Carriage Return after each print command.

Device assignment is as follows:

DDD	DEVICE NAME	SYMBOLIC CODE	OBJECT CODE
Standard Pair 1	Mini-Tape 1 Mini-Tape 2	T#1 T#2	ØØ1 ØØ2
Optional Pair 2	Mini-Tape 3 Mini-Tape 4	T#3 T#4	ØØ3 ØØ4
Optional Pair 3	Mini-Tape 5 Mini-Tape 6	T#5 T#6	ØØ5 ØØ6
Optional Pair 4	Mini-Tape 7 Mini-Tape 8	T#7 T#8	ØØ7 Ø1Ø
	Printer	PRT	Ø13
	Maxi-Tapes	M#N	2ØX

EXAMPLE:

```
PPP-LLL: MP1-MP2-MP3-MP4. E SEQ. NO. LAB: VERB  OPERANDS        COMMENTS

P15-Ø32: 15Ø-ØØØ-122-ØØ4.   Ø2-Ø2Ø.       ENT:  DPL-3.
P15-Ø36: ØØ5-Ø56-ØØ6-Ø66.   Ø2-Ø3Ø.       SET;  PAG:F11,TØ1.    SET PAGE
P15-Ø42: Ø27-2ØØ-ØØØ-ØØ1.   Ø2-Ø4Ø.       PUT;  128,ØØØ,T#1.    WRITE
P15-Ø46: ØØ4-ØØ1-Ø17-ØØØ.   Ø2-Ø5Ø.       SEL;  LOW,P15,ØØØ.    EOF(REF.SPOT)
P15-Ø52: ØØ4-ØØ4-Ø17-144.   Ø2-Ø6Ø.       SEL;  HGH,P15,1ØØ.    TAPE ERROR?
P15-Ø56: 15Ø-Ø1Ø-1Ø5-Ø62.   Ø2-Ø7Ø.       ENT:  DPL-1.          RECORD OK
P15-Ø62: 14Ø-ØØØ.           Ø2-Ø8Ø.       EXU;  ØØØ.            RETURN
```

GET & PUT

LINKAGE:

The GET or PUT Functions may be used in a DPL-1 context in conjunction with the Pseudo (ENT:IOS) or it may be used in a DPL-3 context (ENT:DPL-3).

RETURN STATUS (DPL-3)

If the software condition value (P00-377) is "=", function good.

If the software condition value (P00-377) is ">", tape 8-retry error (or Read Sequence error).

If the software condition value (P00-377) is "<", file mark read.

The above conditions may be tested by the software by using the "SEL;" commands.

Location $P00\text{-}017_8$ contains the Status Byte - Refer to the Instruction Reference Cards for Error Conditions.

Return Status (IGS)

The return from ENT:IOS will set the status in the hardware condition register and can be tested using DPL-1 corresponding to the above conditions.

MOV

MOVE
STORAGE TO STORAGE

OBJECT	SOURCE
037-YXX-YXX-YXX	MOV; NNN, AAA, BBB.

WHERE: NNN is the decimal size.
AND: AAA is a decimal or symbolic "from" address.
AND: BBB is the decimal or symbolic "into" address.

DESCRIPTION:

The DPL-2 move instruction is used for a storage-to-storage move where the data specified by the A-operand is moved to the address specified by the B-operand address. In storage-to-storage movement the fields may overlap in any desired way. Movement is left to right through each field a byte at a time.

The "from" and "into" page are initialized as page one. To move "from" a page, or "into" a page other than page one, a SET PAGE instruction must have been previously executed. The page remains at this setting until a different SET PAGE instruction is executed.

The A-operand and the B-operand may be within the same page or in different pages. Any data continuing past page boundaries will be wrapped around to the beginning of the page.

The software condition value remains unchanged.

The hardware condition register is inpredictable after execution of any instruction executed in DPL-3 or IOS Mode.

EXAMPLE:

```
PPP-LLL: MP1-MP2-MP3-MP4. E SEQ. NO. LAB: VERB  OPERANDS        COMMENTS

P15-Ø64: 15Ø-ØØØ-122-ØØ4.   Ø2-15Ø.       ENT:  DPL-3.          MOVE 10 CHAR.
P15-Ø7Ø: ØØ5-Ø12-ØØ6-Ø66.   Ø2-16Ø.       SET;  PAG:FØ2,TØ1.    FROM PAGE 2
P15-Ø74: Ø37-Ø12-Ø4Ø-Ø4Ø.   Ø2-17Ø.       MOV;  Ø1Ø,Ø32,Ø32.    TO PAGE 1
```

ADD

OBJECT	SOURCE
Ø47-YXX-YXX-YXX	ADD; NNN, AAA, BBB.

WHERE: NNN is the decimal size.
AND: AAA is a decimal or symbolic addend 1 address.
AND: BBB is a decimal or symbolic addend 2 address.
(AAA+BBB)=BBB

DESCRIPTION:

The ADD command adds a decimal value specified by the A-operand to a decimal value specified by the B-operand for the number of bytes indicated by the size operand. The A-operand value and the B-operand value must be the same size. The A-operand value and the B-operand value may contain a sign although it is not included in the size count. Addition is algebraic. The results of the addition displaces the previous contents of the B-operand field and any overflow character is lost. The octal value of ØØ1 (-) is the minus sign. Any other value is assumed to be positive. If the A-operand field contains a minus sign, a sign position must be reserved in the result field.

The A-operand field and the B-operand field may be within the same page or different pages as specified by a Set Page instruction.

The software condition value is unchanged.

The hardware condition register is unpredictable after execution of any instruction executed in DPL-3 or IOS Mode.

EXAMPLE:

```
PPP-LLL: MP1-MP2-MP3-MP4. E SEQ. NO. LAB: VERB  OPERANDS        COMMENTS

P15-1ØØ: Ø47-ØØ3-ØØØ-1ØØ.   Ø3-Ø4Ø.       ADD;  ØØ3,ØØØ,Ø64.
                           *Ø3-Ø5Ø.
                           *Ø3-Ø6Ø.  6ØØ + (B-OPERAND)
                           *Ø3-Ø7Ø.  2ØØ - (A-OPERAND)
                           *Ø3-Ø8Ø.
                           *Ø3-Ø9Ø.  4ØØ + (B-OPERAND RESULT)
```

·DPL-2 GROUP 2: DATA MODIFY SUBTRACT STORAGE-TO-STORAGE

SUB

OBJECT	SOURCE
Ø57-YXX-YXX-YXX	SUB; NNN, AAA, BBB.

WHERE: NNN is the decimal size.
AND: AAA is a decimal or symbolic subtrahend address.
AND: BBB is a decimal or symbolic minuend address.
(BBB-AAA) = BBB

DESCRIPTION:

The SUB command subtracts a decimal value specified by the A-operand from a decimal value specified by the B-operand. The A-operand value and the B-operand value must be the same size.

The A-operand value and the B-operand value may contain a sign, although it is not included in the size count. Subtraction is algebraic. The result of the subtraction displaces the previous contents of the B-operand field and any overflow character is lost.

The octal value of ØØ1 (-) is the minus sign. Any other value is assumed to be positive. A sign position must be reserved in the result field.

The A-operand and the B-operand fields may be within the same page or different pages as specified by a set page instruction.

The software condition value is unchanged.

The hardware condition register is unpredictable after execution of any instruction executed in DPL-3 or IOS Mode.

EXAMPLE:

```
PPP-LLL: MP1-MP2-MP3-MP4. E SEQ. NO. LAB: VERB  OPERANDS        COMMENTS

P15-1Ø4: Ø57-ØØ3-ØØØ-1ØØ.   Ø3-15Ø.       SUB;  ØØ3,ØØØ,Ø64.
                           *Ø3-16Ø.
                           *Ø3-17Ø.  2ØØ - (B-OPERAND)
                           *Ø3-18Ø.  1ØØ + (A-OPERAND)
                           *Ø3-19Ø.
                           *Ø3-2ØØ.  3ØØ - (B-OPERAND RESULT)
```

MUL

OBJECT	SOURCE
067-YXX-YXX-YXX	MUL; NNN, AAA, BBB.

WHERE: NNN is the decimal size.
AND: AAA is a high order decimal or symbolic multiplier address.
AND: BBB is a high order decimal or symbolic multiplicand address.

DESCRIPTION:

The MUL command multiplies a decimal value specified by the B-operand by a decimal value specified by the A-operand (A x B) for the number of bytes indicated by the size operand. The multiplier may contain a sign, although it is not included in the size count. An unsigned multiplier is assumed to be positive. The extended product area, the size of the multiplicand field plus one, filled with decimal zeros, must be reserved immediately following the multiplicand field. If the multiplicand field is to contain a sign, it must appear immediately following the product area. An unsigned multiplicand field is assumed to be positive. The octal value of ØØ1 (-) is the minus sign. Any other value is assumed to be positive. If the Multiplier contains a minus sign, a sign position must be reserved in the product field. The sign result in the product field is algebraic.

The product result of the multiplication displaces the previous contents of the multiplicand field and is right justified with left zeros in the product field.

Unit Position of the Multiplier Field is: AAA+NNN-ØØ1.
Unit Position of the Multiplicand Field is: BBB+NNN-ØØ1.
Unit Position of the Product Field is: BBB+2xNNN.

The multiplier and the multiplicand fields may be within the same page or different pages as specified by a set page instruction.

The software condition value is unchanged.

The hardware condition register is unpredictable after execution of any instruction executed in DPL-3 or IOS Mode.

EXAMPLE:

```
PPP-LLL: MP1-MP2-MP3-MP4. E SEQ. NO. LAB: VERB  OPERANDS        COMMENTS

P15-11Ø: ØØ5-ØØ6-ØØ6-Ø66.   Ø4-Ø6Ø.       SET;  PAG:FØ1,TØ1.
P15-114: Ø67-ØØ3-ØØØ-1ØØ.   Ø4-Ø7Ø.       MUL;  ØØ3,ØØØ,Ø64.
```

DIV

OBJECT	SOURCE
Ø77-YXX-YXX-YXX	DIV; NNN, AAA, BBB.

WHERE: NNN is the decimal size.
AND: AAA is a decimal or symbolic divisor address.
AND: BBB is a decimal or symbolic dividend address.

DESCRIPTION:

The DIV command divides a decimal value specified by the B-operand by a decimal value specified by the A-operand for the number of bytes indicated by the size operand. The size of the dividend field must be twice the size of the divisor field plus one. The dividend field must be right justified and have at least one leading zero. The maximum value of the dividend is the result of the maximum value of a multiply of the same size. $(999....)^2$. The divisor and the dividend fields may contain a sign although it is not included in the size count. The size count is the size of the divisor. An unsigned field is assumed to be positive. Division is algebraic. The octal value ØØ1 (-) is the minus sign. Any other value is assumed to be positive. If the divisor contains a minus sign, a sign position must be reserved in the quotient field.

The quotient result of the division desplaces the previous contents of the dividend field and is left justified. The size of the quotient is the size of the divisor. The remainder is placed immediately following the quotient.

Unit Position of the Divisor Field is: AAA+NNN-ØØ1.
Unit Position of the Dividend Field is: BBB+2xNNN.
Unit Position of the Quotient Field is: BBB+NNN-ØØ1.

The divisor and the dividend fields may be within the same page or different pages as specified by a Set Page instruction.

The software condition value is unchanged.

The hardware condition register is unpredictable after execution of any instruction executed in DPL-3 or IOS Mode.

EXAMPLE:

```
PPP-LLL: MP1-MP2-MP3-MP4. E SEQ. NO. LAB: VERB  OPERANDS        COMMENTS

P15-12Ø: ØØ5-ØØ6-ØØ6-Ø66.   Ø4-13Ø.       SET;  PAG:FØ1,TØ1.
P15-124: Ø77-ØØ3-ØØØ-1ØØ.   Ø4-14Ø.       DIV;  ØØ3,ØØØ,Ø64.
```

COM

OBJECT	SOURCE
ØØ3-YXX-YXX-YXX	COM; NNN, AAA, BBB.

WHERE: NNN is the decimal size.
AND: AAA is the high order decimal or symbolic address of the compare field.
AND: BBB is the high order decimal or symbolic address of the field compared to.

DESCRIPTION:

Within the current page setting established by a set page instruction, compare the data specified by the A-operand address to the data specified by the B-operand address for the number of bytes indicated by the size operand. The comparison operation proceeds left to right through each field a byte at a time and ends when an inequality is found or end of field is reached. Comparison is binary, with a collating sequence based on ascending binary values. All codes are valid. Memory is not altered as a result of this operation. A field that overflows a page boundary will wrap around to the beginning of the page.

The software condition register is memory location PØØ 377_8. This will contain an octal Ø6Ø for > , Ø57 for < or Ø66 for =. The "SEL" instructions will test these conditions.

The result of the compare operation is indicated by the software condition value.

HIGH	AAA	>	BBB
LOW	AAA	<	BBB
EQUAL	AAA	=	BBB

EXAMPLE:

```
PPP-LLL: MP1-MP2-MP3-MP4. E SEQ. NO. LAB: VERB  OPERANDS          COMMENTS

P15-13Ø: ØØ5-ØØ2-Ø76-Ø66.   Ø5-Ø1Ø.       SET;  PAG:FØØ,T15.
P15-134: ØØ3-Ø12-15Ø-31Ø.   Ø5-Ø2Ø.       COM;  Ø1Ø,ØØ2,2ØØ.
P15-14Ø: ØØ4-ØØ2-ØØ6-ØØØ.   Ø5-Ø3Ø.       SEL;  EQL,P06,ØØØ.
P15-144: ØØ4-ØØØ-Ø14-ØØØ.   Ø5-Ø4Ø.       SEL;  UNC,P12,ØØØ.
P15-15Ø:                    Ø5-Ø5Ø.  CØ2: A/N:  (COMPARE XX).
```

SEL;UNC

OBJECT	SOURCE
004-000-YXX-YXX	SEL; UNC, RRR. PNN, LLL.

WHERE: RRR is a symbolic address.
AND: NN is a decimal page.
AND: LLL is a decimal address.

DESCRIPTION:

The Select (Branch) Uncondition command is used in a DPL-3 context to transfer control to a new instruction location regardless of the setting of the software condition value.

The DPL-2 commands are executed in an interpretive mode and therefore are not limited to section boundaries. In the interpretive mode the SEL command may be used to transfer control to any DPL-2 command or to any DPL-1 command except the DPL-1 branch functions and the jump functions. DPL-1 branch functions and jump functions can only be used in the DPL-1 mode of operation (ENT:DPL-1). The branch address may be represented as a symbolic address or as an absolute address. The software condition value remains unchanged.

EXAMPLE:

```
PPP-LLL: MP1-MP2-MP3-MP4. E SEQ. NO. LAB: VERB  OPERANDS        COMMENTS

P15-162: ØØ4-ØØØ-Ø17-ØØØ.    Ø5-11Ø.       SEL;  UNC,P15,ØØØ.
P15-166: ØØ4-ØØØ-Ø15-166.    Ø5-12Ø.  SØ1: SEL;  UNC,SØ1.
```

SEL;LOW

OBJECT	SOURCE
ØØ4-ØØ1-YXX-YXX	SEL; LOW, RRR. PNN, LLL.

WHERE: RRR is a symbolic address
AND: NN is a decimal page.
AND: LLL is a decimal address.

DESCRIPTION:

The conditional branch command, select LOW, is used in a DPL-3 context to transfer control to a new instruction location if the software condition register previously set by a DPL-2 compare or a DPL-2 I/O instruction is found to be LOW. If the condition is not satisfied, the next sequential instruction is executed.

(Refer to "SEL;UNC" for Basic Rules of Select Branching)

EXAMPLE:

```
PPP-LLL: MP1-MP2-MP3-MP4. E SEQ. NO. LAB: VERB  OPERANDS          COMMENTS

P15-172: ØØ4-ØØ1-Ø17-ØØØ.   Ø5-18Ø.       SEL;  LOW,P15,ØØØ.
P15-176: ØØ4-ØØ1-Ø15-176.   Ø5-19Ø.  SØ2: SEL;  LOW,SØ2.
```

SEL;EQL

OBJECT	SOURCE
ØØ4-ØØ2-YXX-YXX	SEL; EQL, RRR. PNN, LLL.

WHERE: RRR is a symbolic address
AND: NN is a decimal page.
AND: LLL is a decimal address.

DESCRIPTION:

The conditional branch command, select EQUAL, is used in a DPL-3 context to transfer control to a new instruction location if the software condition value previously set by a DPL-2 compare or DPL-2 I/O instruction is found to be EQUAL. If the condition is not satisfied, the next sequential instruction is executed.

(Refer to "SEL;UNC" for Basic Rules of Select Branching)

EXAMPLE:

```
PPP-LLL: MP1-MP2-MP3-MP4. E SEQ. NO. LAB: VERB  OPERANDS          COMMENTS

P15-2Ø2: ØØ4-ØØØ-Ø17-ØØØ.   Ø6-Ø5Ø.       SEL;  EQU,P15,ØØØ.
P15-2Ø6: ØØ4-ØØ2-Ø15-2Ø6.   Ø6-Ø6Ø.  SØ3: SEL;  EQL,SØ3.
```

SEL;HGH

OBJECT	SOURCE
ØØ4-ØØ4-YXX-YXX	SEL; HGH, RRR. PNN, LLL.

WHERE: RRR is a symbolic address.
AND: NN is a decimal page.
AND: LLL is a decimal address.

DESCRIPTION:

The conditional branch command, select HIGH, is used in a DPL-3 context to transfer control to a new instruction location if the software condition value previously set by a DPL-2 compare or DPL-2 I/O instruction is found to be HIGH. If the condition is not satisfied, the next sequential instruction is executed.

(Refer to "SEL;UNC" for Basic Rules of Select Branching)

EXAMPLE:

```
PPP-LLL: MP1-MP2-MP3-MP4. E SEQ. NO. LAB: VERB  OPERANDS          COMMENTS

P15-212: ØØ4-ØØ4-Ø17-ØØØ.    Ø6-12Ø.      SEL;  HGH,P15,ØØØ.
P15-216: ØØ4-ØØ4-Ø15-216.    Ø6-13Ø. SØ4: SEL;  HGH,SØ4.
```

GROUP 3: SELECT SELECT NOT HIGH

SEL;NHG

OBJECT	SOURCE
ØØ4-Ø14-YXX-YXX	SEL; NHG, RRR. PNN, LLL.

WHERE: RRR is a symbolic address.
AND: NN is a decimal page.
AND: LLL is a decimal address.

DESCRIPTION:

The conditional branch command, select NOT HIGH, is used in a DPL-3 context to transfer control to a new instruction location if the software condition value previously set by a DPL-2 compare or DPL-2 I/O instruction is found to be LOW or EQUAL. If the condition is not satisfied, the next sequential instruction is executed.

(Refer to "SEL;UNC" for Basic Rules of Select Branching)

EXAMPLE:

```
PPP-LLL: MP1-MP2-MP3-MP4. E SEQ. NO. LAB: VERB  OPERANDS          COMMENTS

P15-222: ØØ4-Ø14-Ø17-ØØØ.   Ø6-2ØØ.       SEL;  NHG,P15,ØØØ.
P15-226: ØØ4-Ø14-Ø15-226.   Ø7-Ø1Ø   SØ5: SEL;  NHG,SØ5.
```

SEL;NEQ

OBJECT	SOURCE
ØØ4-Ø12-YXX-YXX	SEL;NEQ, RRR. PNN, LLL.

WHERE: RRR is a symbolic address.
AND: NN is a decimal page.
AND: LLL is a decimal address.

DESCRIPTION:

The conditional branch command, select NOT EQUAL, is used in a DPL-3 context to transfer control to a new instruction location if the software condition value previously set by a DPL-2 compare or DPL-2 I/O instruction is found to be LOW or HIGH. If the condition is not satisfied, the next sequential instruction is executed.

(Refer to "SEL;UNC" for Basic Rules of Select Branching)

EXAMPLE:

```
PPP-LLL: MP1-MP2-MP3-MP4. E SEQ. NO. LAB: VERB  OPERANDS          COMMENTS

P15-232: ØØ4-Ø12-Ø17-ØØØ.   Ø7-Ø7Ø.       SEL;  NEQ,P15,ØØØ.
P15-236: ØØ4-Ø12-Ø15-236.   Ø7-Ø8Ø.  SØ6: SEL;  NEQ,SØ6.
```

· DPL-2 GROUP 3: SELECT SELECT NOT LOW

SEL;NLW

OBJECT	SOURCE
ØØ4-Ø11-YXX-YXX	SEL;NLW, RRR. PNN, LLL.

WHERE: RRR is a symbolic address.
AND: NN is a decimal page.
AND: LLL is a decimal address.

DESCRIPTION:

The conditional branch command, select NOT LOW, is used in a DPL-3 context to transfer control to a new instruction location if the condition value previously set by a DPL-2 compare or DPL-2 I/O instruction is found to be HIGH or EQUAL. If the condition is not satisfied, the next sequential instruction is executed.

(Refer to "SEL;UNC" for Basic Rules of Select Branching)

EXAMPLE:

```
PPP-LLL: MP1-MP2-MP3-MP4. E SEQ. NO. LAB: VERB  OPERANDS         COMMENTS

P15-242: ØØ4-Ø11-Ø17-ØØØ.   Ø7-14Ø.       SEL;  NLW,P15,ØØØ.
P15-246: ØØ4-Ø11-Ø15-246.   Ø7-15Ø.  SØ7: SEL;  NLW,SØ7.
```

SET

OBJECT	SOURCE
ØØ5-FFF-TTT-III	SET; PAG: FNN, TNN.

WHERE: FFF is the DPL A-Operand (from) page.
AND: TTT is the DPL B-Operand (to) page.
AND: III is the DPL Instruction page.

WHERE: FNN is the decimal page setting for the A-Operand data instructions.
AND: TNN is the decimal page setting for the B-Operand data instructions.

DESCRIPTION:

The A-Operand and the B-Operand, in the DPL-2 data functions, specify an address within a page where the data resides. The set page instruction provides a means of controlling the setting of that page as a base address. The page setting for the A-Operand and the B-Operand may be the same page or they may be different pages regardless of the section. Only operands that specify data use the page setting. The page setting is unchanged until another set page instruction is executed reflection different pages.

The software condition value is unchanged.

The hardware condition register is unpredictable after execution on any instruction executed in DPL-3 or IOS Mode.

EXAMPLE:

```
PPP-LLL: MP1-MP2-MP3-MP4. E SEQ. NO. LAB: VERB  OPERANDS          COMMENTS

P15-252: ØØ5-ØØ6-Ø76-Ø66.   Ø8-Ø1Ø.  SØ8: SET;  PAG:FØ1,T15.     MOVE CHAR.
P15-256: Ø37-Ø12-Ø24-112.   Ø8-Ø2Ø.       MOV;  Ø1Ø,Ø2Ø,Ø74.     FROM PAGE 1
P15-262: Ø37-ØØ4-Ø12-Ø24.   Ø8-Ø3Ø.       MOV;  ØØ4,Ø1Ø,Ø2Ø.     TO PAGE 15
```

TCL

OBJECT	SOURCE
ØØ7-ØØ1-ØXX-ØØ1	TCL; ØØ1, T#N, BSP.
ØØ7-ØØ1-2XX-ØØ1	TCL; ØØ1, M#N, BSP.
ØØ7-ØØ1-2XX-ØØ2	TCL; ØØ1, M#N, RWD.
ØØ7-ØØ1-2XX-ØØ3	TCL; ØØ1, M#N, RWI.
ØØ7-ØØ1-2XX-ØØ4	TCL; ØØ1, M#N, WFM.

WHERE: N is a decimal tape device number,
AND: BSP is a backspace record function,
AND: RWI is a maxi-tape rewind with interlock function,
AND: RWD is a maxi-tape rewind without interlock function,
AND: WFM is a maxi-tape write file mark.

DESCRIPTION:

These commands control the basic tape operations for the device specified in the instruction. The backspace function (BSP) applies to all tape I/O devices. The backspace function backspaces the device specified by one record.

The two rewind functions and the write file mark function apply only to maxi-tape devices. The rewind with interlock function (RWI) rewinds the maxi-tape specified and takes the device off-line. After the device has been set off-line, manual intervention is required to return the device to on-line status. The rewind without interlock function (RWD) rewinds the maxi-tape specified, but does not take the device off-line. The write file mark function (WFM) writes a special hardware 3-byte file mark for the maxi-tape specified.

EXAMPLE:

```
PPP-LLL: MP1-MP2-MP3-MP4. E SEQ. NO. LAB: VERB  OPERANDS        COMMENTS

P15-266: ØØ7-ØØ1-ØØ1-ØØ1.   Ø8-1ØØ.       TCL;  ØØ1,T#1,BSP.    BKSP MINI-1
P15-272: ØØ7-ØØ1-2Ø1-ØØ1.   Ø8-11Ø.       TCL;  ØØ1,M#1,BSP.    BKSP MAXI
P15-276: ØØ7-ØØ1-2Ø2-ØØ2.   Ø8-12Ø.       TCL;  ØØ1,M#2,RWD.    RWD MAXI W/INTL
P15-3Ø2: ØØ7-ØØ1-2Ø3-ØØ3.   Ø8-13Ø.       TCL;  ØØ1,M#3,RWI.    RWD MAXI-NO INT
P15-3Ø6: ØØ7-ØØ1-2Ø4-ØØ4.   Ø8-14Ø.       TCL;  ØØ1,M#4,WFM.    WR.FILE MK-MAXI
```

GROUP 4. TYPEWRITER CONTROL (TYPES I & II)

PCL

OBJECT	SOURCE
ØØ7-ØØ1-Ø13-ØØ2	PCL; ØØ1, PRT, BSC.
ØØ7-ØØ1-Ø13-ØØ3	PCL; ØØ1, PRT, RRS.
ØØ7-ØØ1-Ø13-ØØ4	PCL; ØØ1, PRT, CRT.
ØØ7-ØØ1-Ø13-ØØ6	PCL; ØØ1, PRT, BRS.
ØØ7-ØØ1-Ø13-Ø11	PCL; ØØ1, PRT, TAB.
ØØ7-ØØ1-Ø13-Ø12	PCL; ØØ1, PRT, IDX.

DESCRIPTION:

These commands will control the paper and carriage positioning on the IBM 73Ø and 735 typewriters.

ØØ2	(BSC)	Backspace carriage one character position.
ØØ3	(RRS)	Red ribbon shift (optional on 735).
ØØ4	(CRT)	Carriage return.
ØØ6	(BRS)	Black ribbon shift (optional on 735).
Ø11	(TAB)	Position carriage to the first tab stop.
Ø12	(IDX)	Index paper one line.

EXAMPLE:

```
PPP-LLL: MP1-MP2-MP3-MP4. E SEQ. NO. LAB: VERB  OPERANDS        COMMENTS

P15-312: ØØ7-ØØ1-Ø13-Ø11.   Ø8-2ØØ.  CPT: PCL;  ØØ1,PRT,TAB.    TAB ONE FIELD
P15-316: Ø27-Ø12-ØØØ-Ø13.   Ø9-Ø1Ø.       PUT;  Ø1Ø,ØØØ,PRT.   PRINT 1Ø CHAR.
P15-322: ØØ7-ØØ1-Ø13-ØØ4.   Ø9-Ø2Ø.       PCL;  ØØ1,PRT,CRT.   CARRIAGE RETURN
P15-326: ØØ7-ØØ1-Ø13-Ø12.   Ø9-Ø3Ø.       PCL;  ØØ1,PRT,IDX.   INDEX ONE LINE
P15-332: ØØ4-ØØØ-Ø15-312.   Ø9-Ø4Ø.       SEL;  UNC,CPT.
```

PCL

OBJECT	SOURCE
ØØ7-ØØ1-Ø13-Ø6Ø	PCL; ØØ1, PRT, TOF.
ØØ7-ØØ1-Ø13-Ø7Ø	PCL; ØØ1, PRT, LFD.

DESCRIPTION:

These commands control the paper positioning on the medium speed line printers.

Ø6Ø (TOF) When this command is given the paper will slew to "top of form".

Ø7Ø (LFD) This command will feed one line of paper.

EXAMPLE:

```
PPP-LLL: MP1-MP2-MP3-MP4. E SEQ. NO. LAB: VERB  OPERANDS         COMMENTS

P15-336: Ø27-1ØØ-ØØØ-Ø13.   Ø9-1ØØ.  WTT: PUT;  Ø64,ØØØ,PRT.    PRINT 64 CHAR.
P15-342: ØØ4-ØØ4-Ø17-ØØØ.   Ø9-11Ø.       SEL;  HGH,P15,ØØØ.    PRINT ERROR
P15-346: ØØ7-ØØ1-Ø13-Ø7Ø.   Ø9-12Ø.       PCL;  ØØ1,PRT,LFD.    ADVANCE 1 LINE
P15-352: ØØ4-ØØØ-Ø15-336.   Ø9-13Ø.       SEL;  UNC,WTT.
```

NOTATIONS FOR DPL-3B CONSTANTS

These statements are used to enter data constants into memory, to define and reserve areas of memory, and to specify the address of relocatable symbols. The statements may be named by symbols so that other program elements can refer to the fields they generate.

The forms OCT, DEC, HEX and A/N may specify one constant or a string of constants.

The form DSA provides a method of reserving specified areas of memory for future reference. The contents of the reserved area is not disturbed.

The form ADC provides a means of storing the address components of relocatable symbols. ADC generates a two-byte constant, containing the DPL code of the page and the octal code of the location of the symbol.

CONSTANT NOTATIONS

OCT: (NNN-NNN-NNN-etc.)

A byte-string constant in octal notation where the maximum number of terms is six.

HEX: (HH-HH-HH-etc.)

A byte-string constant in Hex notation where the maximum number of terms is eight.

DEC: (NNN-NNN-NNN-etc.)

A byte-string constant in decimal notation where the maximum number of terms is six.

A/N: (XXXXXXX... etc.)

A string of keyboard characters where the maximum number of characters is 24.

DSA: (NNN)

Define Symbol area where NNN is decimal number of bytes required up to 256.

ADC: (AAA±NNN)

Address constant for labels in symbolic notation. This instruction generates two-bytes. The first byte contains the DPL page of the address specified in increment form. The second byte contains the location.

ORG

OBJECT	SOURCE
	ORG: PNN, LLL. ORG: PNN.

WHERE: NN is the decimal page of the program origin.
AND: LLL is the decimal location within the page.
AND: Where LLL is not specified location ØØØ is assumed.

DESCRIPTION

The assembler uses the decimal term specified by the operand to alter the setting of the location counter for the current segment. This value should be on a half-word boundary if instruction statements are to follow. The ORG instruction must appear following each SEG (segment) or OVL (overlay) statement, and may appear elsewhere within the segment. If the ORG instruction is omitted following SEG or OVL, the assembler sets the initial instruction address to zero. The ORG operand specifies a page and location as either an absolute address, or as an implied address of location ØØØ within the page specified, if the location is not included in the operand. Each ORG statement is considered one label of the 128 possible labels.

EXAMPLE:

```
PPP-LLL: MP1-MP2-MP3-MP4. E SEQ. NO. LAB: VERB  OPERANDS        COMMENTS

                           1Ø-Ø1Ø.      OVL:  PID.           OVERLAY ID
                           1Ø-Ø2Ø.      ORG:  P1Ø.           ORIGIN P1Ø,ØØØ
P12-ØØØ:                   1Ø-Ø3Ø.      A/N:  (Ø12).
                           1Ø-Ø4Ø.      ORG:  P1Ø, 128.      ORIGIN P1Ø,128
P12-2ØØ:                   1Ø-Ø5Ø.      A/N:  (XXX).
```

·DPL-1 CLASS: PSEUDO IDENTIFY SEGMENT

SEG

OBJECT	SOURCE
	SEG: PID.

WHERE: PID is any 3 character segment identification.

DESCRIPTION:

A segment is a block of program coding that can be relocated independently of other coding if linkage addresses are changed where necessary. The concept of program segmenting is a consideration at coding time, assembly time, and at object generation time. By using the form of the Branch functions specifying the absolute address to which control is to be passed at execution time, external segments may be referenced. In assembled multi-segment programs, segments may symbolically address locations in other segments. A program is composed of at least one segment, and the SEG or OVL pseudo must be the first instruction encountered during assembly which is immediately followed by an ORG pseudo. Any three characters may be used for segment identification. The SEG identification is contained in all subsequent source instructions up to the end of the segment. The SEG identification assigned by the SEG pseudo is used in conjunction with the USE pseudo to retrieve external segments at object generation time.

EXAMPLE:

```
PPP-LLL: MP1-MP2-MP3-MP4. E SEQ. NO. LAB: VERB  OPERANDS        COMMENTS

                           1Ø-11Ø.      SEG:  PID.             SEGMENT ID
                           1Ø-12Ø.      ORG:  PØ8, ØØØ.        START LOCATION
P1Ø-ØØØ: 2ØØ-ØØØ.          1Ø-13Ø.      LDA,  R#Ø; OCT:ØØØ.
```

· DPL-1 PSEUDO ENTER CONTROL FUNCTION

ENT

OBJECT	SOURCE
15Ø-YXX-1ØX-YXX	ENT: DPL-1.
15Ø-ØØØ-122-ØØ4	ENT: DPL-3.
15Ø-ØØØ-123-ØØ4	ENT: IOS.

WHERE: 15Ø-YXX is the section.

WHERE: DPL-1 is a machine executable mode.
AND: DPL-3 is an interpretive mode.
AND: IOS is the I/O supervisor.

DESCRIPTION:

ENT:

The ENT pseudo instructions change the operating context for the program instructions. There are three forms of the ENT pseudo and each generates Branch linkage code to the appropriate control point.

ENT:DPL-1 switches the instruction environment from interpretive DPL-3 mode into direct execution DPL-1 mode. DPL-1 mode is the normal hardware context and executes instructions at machine speed. Only DPL-1 and Pseudo instructions may be executed in DPL-1 mode. This pseudo will be an SMS and a BRU to the next instruction in sequence.

ENT:DPL-3 switches the instruction environment from direct DPL-1 mode into interpretive DPL-3 mode. In DPL-3 mode any DPL-1 instruction except Class Ø, Class 1 and any DPL-2 statement may be executed under control of a resident software monitor. Exit from DPL-3 mode is accomplished only with an ENT:DPL-1 pseudo instruction.

ENT:IOS switches control temporarily from a DPL-1 context into the Input/Output Supervisor for the execution of one I/O function. The DPL-2 I/O commands (GET, PUT, SET, TCL, PCL) are used to specify the I/O operation. Following execution, control is automatically returned to DPL-1 mode and the succeeding instructions.

IOS:

The Input Output Supervisor is a resident monitor program used to provide complete I/O functions for DPL-1 programs. The ENT:IOS pseudo instruction is used to turn program control over to the Supervisor. After one complete I/O function has been performed, program control is automatically returned to the using DPL-1 program. The I/O function to be performed is specified using a GET, PUT, PCL or TCL command from the DPL-2 instruction set. A SET;PAG command may precede the I/O function command where required.

(continued)

The I/O buffer page for the Supervisor is set initially to page Ø1. The SET;PAG command changes the page context for the GET and PUT commands where desired.

All index registers in section Ø are used by the IOS during its operation and their contents lost. Any valuable data contained in these index registers should be saved by the user progrom before calling the Supervisor and restored by the user program after return from the IOS.

The Supervisor uses the software status byte as a status indicator upon its return to the user program. This condition may be tested by using the DPL-2 "SEL" command. Equal condition means that the I/O function has been successfully completed. High condition indicates that an error has occurred. Low condition means that a file mark or end of file record has been detected during a tape read operation. When a high condition is encountered the user should branch to an error routine. The software status byte ($PØØ-Ø17_8$) can help diagnose an error that occurred during a mini-tape read or write. If the status byte contains the value eight, a retry failure is indicated.

Refer to the Cogar System 4 Instruction Reference Card for explanation of the status byte.

EXAMPLE:

```
PPP-LLL: MP1-MP2-MP3-MP4. E SEQ. NO. LAB: VERB  OPERANDS        COMMENTS

P1Ø-ØØ2: 2Ø1-Ø12.           1Ø-19Ø.       LDX,  R#1; HEX:ØA.
P1Ø-ØØ4: 15Ø-ØØØ-122-ØØ4.   1Ø-2ØØ.       ENT:  DPL-3.
P1Ø-Ø1Ø: Ø17-2ØØ-ØØ1-2ØØ.   11-Ø1Ø.       GET;  128,T#1,128.
P1Ø-Ø14: Ø37-Ø12-2ØØ-226.   11-Ø2Ø.       MOV;  Ø1Ø,128,15Ø.
P1Ø-Ø2Ø: Ø47-Ø12-226-31Ø.   11-Ø3Ø.       ADD;  Ø1Ø,15Ø,2ØØ.
P1Ø-Ø24: 15Ø-Ø1Ø-1ØØ-Ø3Ø.   11-Ø4Ø.       ENT:  DPL-1.
                           *11-Ø5Ø.
P1Ø-Ø3Ø: 15Ø-ØØØ-123-ØØ4.   11-Ø6Ø.       ENT:  IOS.
P1Ø-Ø34: ØØ5-ØØ6-Ø76-Ø42.   11-Ø7Ø.       SET;  PAG:FØ1,T15.   (OPTIONAL INST)
P1Ø-Ø4Ø: Ø17-2ØØ-ØØ1-ØØØ.   11-Ø8Ø.       GET;  128,T#1,ØØØ.   AUTOMATICALLY
P1Ø-Ø44: 2Ø1-Ø3Ø.           11-Ø9Ø.       LDX,  R#1, DEC:Ø24.  ENTERS DPL-1
```

EQU

OBJECT	SOURCE
	LAB: EQU: PNN, LLL. LAB: EQU: RRR.

WHERE: LAB is a symbolic label.
AND: NN is the decimal page.
AND: LLL is the decimal location within the page.
AND: RRR is a symbolic reference.

DESCRIPTION

The EQU pseudo instruction defines a symbol by assigning it to either an absolute location or another symbol. The EQU instruction is the means of equating symbols to registers, relocatable expressions, and other arbitrary values.

The EQU operand may be represented as an absolute expression, or as a symbolic label present in the context of the program unit.

EXAMPLE:

```
PPP-LLL: MP1-MP2-MP3-MP4. E SEQ. NO. LAB: VERB  OPERANDS        COMMENTS

                           11-16Ø.       SEG:  ABC.             EQUATE EØ1 TO
                           11-17Ø.       ORG:  P1Ø.             AN EXTERNAL
                           11-18Ø.  EØ1: EQU:  P12, ØØØ.        SEGMENT
                           11-19Ø.  EO2: EQU:  XYZ.             EØ2-INTERNAL
                          *11-2ØØ.
P12-ØØØ: 231-Ø52.          12-Ø1Ø.  XYZ: STA,  I#1; P1Ø.
P12-ØØ2: 1Ø4-ØØØ.          12-Ø2Ø.       BRU;  EØ1.
```

·DPL-1 CLASS: PSEUDO OVERLAY

OVL

OBJECT	SOURCE
	OVL: PID.

WHERE: PID is any 3 non-blank character overlay identification.

DESCRIPTION:

The OVL pseudo names a section of program coding in the same way that SEG pseudo does, and restrictions are identical. Program overlays must be considered at coding time.

In contrast to the SEG segment, which generates object coding into the main body of the program, the OVL segment generates overlay records outside the main body.

At object generation time, each OVL segment is inserted into the object string tape, in the order that they occur, following the records used in loading the full memory. The overlay records are retrieved into memory, under user program control, using the normal I/O procedures.

EXAMPLE:

```
PPP-LLL: MP1-MP2-MP3-MP4. E SEQ. NO. LAB: VERB  OPERANDS        COMMENTS

                          12-ØØØ.       SEG:  PID.            MAIN BODY OF
                          12-Ø1Ø.       ORG:  P1Ø, ØØØ.       PROGRAM
                         *12-1ØØ.
                         *12-11Ø.
                          12-12Ø.       OVL:  XYZ.            1ST OVERLAY
                          12-13Ø.       ORG:  PØ2, ØØØ.       RECORD
                         *12-14Ø.
                          12-15Ø.       END:  *+Ø.            (MUST HAVE END)
```

DPL-1 CLASS: PSEUDO USE EXTERNAL SOURCE SEGMENT

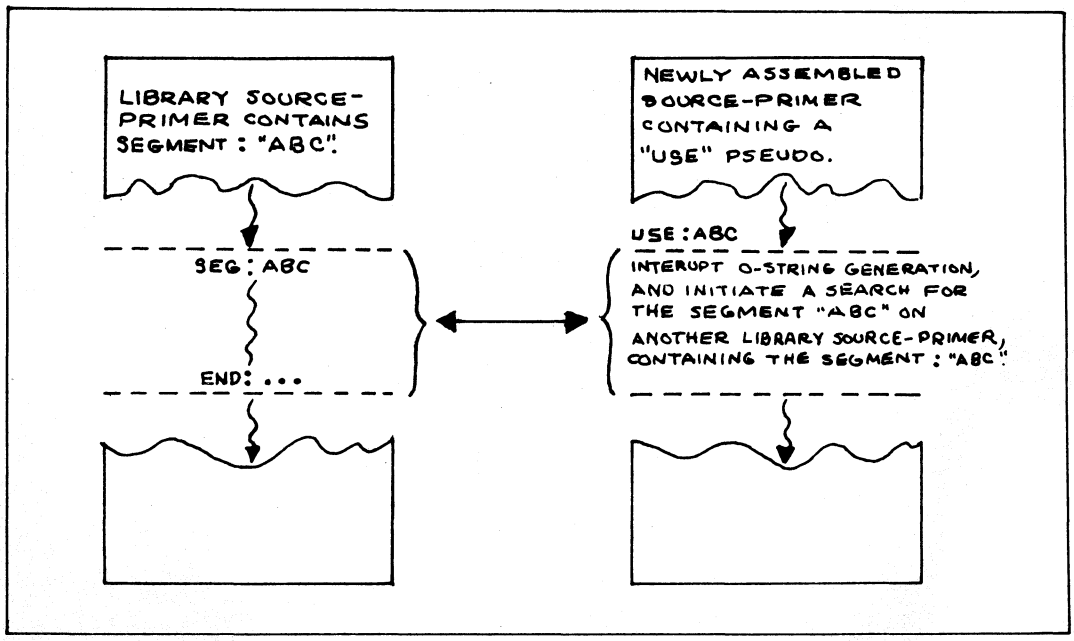
USE

OBJECT	SOURCE
	USE: PID.

WHERE: PID is an external OVL or SEG identifier.

DESCRIPTION

The USE pseudo instruction identifies an independently defined segment or overlay that is to form part of the current program. The retrieval of these segments, at object generation time, can be effected only if the assembler is able to identify the 3 position name in the USE operand with a segment name established on the source primer tape. At object generation time, when a USE pseudo is encountered, the object string generation of the current source primer is interrupted. The user is then instructed to continue processing, using the source primer tape containing the identification specified by the USE operand. The first set of identifiers that satisfy the USE operand will continue the object generation process until an END pseudo instruction is reached. If, within the segment specified by the USE pseudo, another use pseudo is encountered (nested USE) the same interrupt procedure takes place. When a segment is completed, the trail back must be initiated by using the most currently interrupted source primer.



'DPL-1 CLASS: PSEUDO END SEGMENT

END

OBJECT	SOURCE
1ØX-YXX	END: PNN, LLL.
1ØX-YXX	END: RRR.

WHERE: 1ØX is a Branch command.
AND: X is the page within Section I.
AND: YXX is the location.

WHERE: NN is the decimal page address.
AND: LLL is the decimal location.
AND: RRR is a symbolic reference.

DESCRIPTION:

The END pseudo instruction is used to define the end of a program segment or overlay and to identify the starting address for program execution. This starting address may be different than the origin address and is specified as an absolute or symbolic address within Section I. For multiple segment programs, the starting address from the last encountered END instruction is used.

The END pseudo generates a Branch Unconditional instruction to the address specified by the operand.

At object generation time, this generated Branch instruction is inserted into the background in the corresponding location of Page 2, locations 2 and 3, and is used as the entry point within the section assigned (PØ2-ØØØ) at execution time. If the format (END:*+Ø.) is used, this branch address will not be inserted into the background. This format is commonally used to terminate and overlay.

EXAMPLE:

```
PPP-LLL: MP1-MP2-MP3-MP4. E SEQ. NO. LAB: VERB  OPERANDS        COMMENTS

                           13-14Ø.       SEG:  XYZ.
                           13-15Ø.       ORG:  PØ8, ØØØ.
                          *13-16Ø.
                          *13-17Ø.
P1Ø-ØØØ: 174-Ø11.          13-18Ø.  BGN: IOC,  C#4; Ø11.       DISPLAY
                          *13-19Ø.
                          *13-2ØØ.
                           14-Ø1Ø.       END:  BGN.            LINK TO BEGIN
```

EJT

OBJECT	SOURCE
	EJT: ØØØ.

DESCRIPTION:

The EJT pseudo instruction causes the printer to go to Top of Form before printing out the next instruction, during a Source Listing operation. It thus allows the programmer to set up the listings in easily read formats. The Sequence Number for this instruction, but not the instruction itself, is printed on the Source Listing.

IOC

OBJECT	SOURCE
17Ø 171 172 } FFF	IOC, C#N; FFF.

WHERE: FFF is Function Code

WHERE: N=Ø Current tape channel selected.
N=1 Tape cartridge #1 selected.
N=2 Tape cartridge #2 selected.

TIMING: 4 - 6 Microseconds

FFF (Function Codes) DESCRIPTION

ØØØ FORWARD, SLOW, ERASE

Start Tape Forward with Erase. This command sets the run direction and the erase control for the selected tape channel. This command initiates a Tape Write Routine. Since the Erase is active, the tape will be erased until subsequent write data commands. After this command is given, a time delay of approximately 30 M sec should be given to allow the tape to reach a stable speed of 10 in/sec. before the writing of data is begun.

ØØ1 FORWARD, SLOW

This Instruction starts the selected tape forward with Erase condition off. This instruction generally initiates a tape read sequence.

ØØ2 FORWARD, FAST

This command sets the run, forward, and high speed control in the tape electronics starting the tape forward at 40 inches/sec.

ØØ3 REVERSE, SLOW

This command sets the run control, resets the high speed and forward control, starting the tape reverse normal speed. When the Forward Control is in reset state, the erase function is inhibited preventing the erasing of data on any tape reverse condition. When in reverse if the clip comes home, (tape rewound) the run control is reset stopping tape motion.

ØØ4 REVERSE, FAST

This command is identical to the reverse normal command except the tape is driven at high speed 40 inches/sec.

FFF (Function Codes) DESCRIPTION CONTINUED

ØØ5 STOP

This command resets the run control stopping tape. Again, the channel may or may not be specified.

ØØ7 TAPE TRANSFER BYTE AND
2Ø7 TAPE TRANSFER BYTE, SKIP

This command controls the transfer of characters to and from the tape interface. The transfer is controlled by a Busy or Not Ready condition within the tape controls and can be executed in two modes, Stall on Busy, and Skip on Busy. When executed in the Stall if Busy mode, the program stalls at the transfer byte instruction until a tape sprocket is generated indicating that a byte has been written or read. In the Skip on Busy mode, the program automatically skips the next sequential instruction if a character has not been received.*

When a program is in a Read or Write subroutine, a Transfer Byte instruction must be given every 512 us. I.E. - the loop from transfer byte to transfer byte must not exceed 512 us.

Ø1Ø WRITE MODE

This command sets the write operation and begins the timing sequence that controls the writing frequency, loads and shifts the tape buffer and generates the sprockets to drop the stall condition.

This command will follow the start forward normal with erase command, the time delay and any set-up commands. Included in the set-up instruction should be a loading of the ACC with the first to be written. Immediately following the write set should be a transfer byte command. The write set must be given only once in a write sequence.

Ø11 READ MODE

This command activates the tape read circuitry within the tape systems. It has to be executed only once in a normal tape read sequence. The setting of the Read condition/resets the Write condition.

Ø12 REWIND

This command will set a rewind F/F for the specified tape or the current tape which will be reset only by the clip-in signal. This permits overlapped rewinds or rewinding one tape while performing an operation on the other tape.

*NOTE: The Accumulator value is destroyed after execution of a "Skip/Busy" Instruction.

FFF (Function Codes) DESCRIPTION: Continued

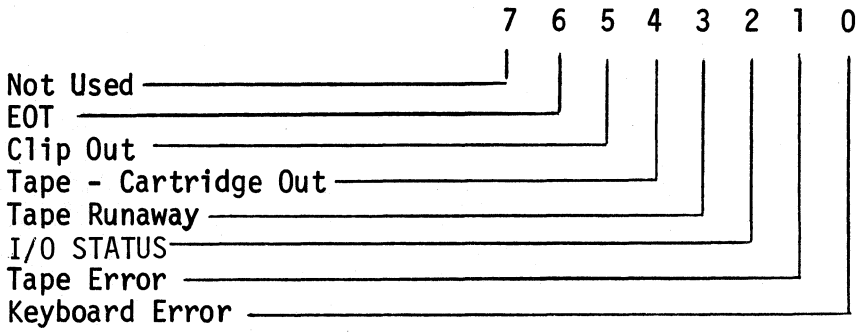
Ø16 SELECT DECK 1 IF N=1, DECK 2 IF N=2 & LOAD ITS STATUS (PAIR 1)

Ø26 SELECT DECK 3 IF N=1, DECK 4 IF N=2 & LOAD ITS STATUS (PAIR 2)

Ø36 SELECT DECK 5 IF N=1, DECK 6 IF N=2 & LOAD ITS STATUS (PAIR 3)

Ø46 SELECT DECK 7 IF N=1, DECK 8 IF N=2 & LOAD ITS STATUS (PAIR 4)

These commands will Select a Tape Deck and load its status in the Accumulator. If N=Ø, the status of the current deck will be loaded.



This instruction will end a read sequence. After the status has been loaded to the Accumulator, the Read, Runaway and Tape Error will be reset. It is important that this command be given before stopping tape and before the end of the block. Either of these conditions gives an energy dropout and a resulting tape error.

This instruction may ask for either channel of one of the 4 PAIRS or the current channel. A specific channel command is useful in a rewind test to determine end of rewind. It is not possible to check the error status on both decks of a selected pair since the first command will reset the error. A Tape error occurs during the reading of a block of data if a significant crossing falls outside of the data window or any energy dropout of 2 ms occurs during a write check. A runaway condition occurs if the read F/F is set and no energy is detected for approximately 5 sec at normal speed or 50 M sec at high speed. These conditions set the tape error F/F and the runaway F/F respectively.

When either F/F is set the tape logic forces the generation of sprockets from the internal timing rather than data, to allow the program loop to finish.

The runaway condition will also reset the run F/F, stopping tape.

IOC

OBJECT	SOURCE
173-FFF	IOC, C#3; FFF.

TIMING: 4 - 6 Microseconds. WHERE: FFF is the Function Code

FFF (Function Codes) Description

ØØ7 KEYBOARD TRANSFER BYTE AND
2Ø7 KEYBOARD TRANSFER BYTE, SKIP

This command controls the transfer of characters from the keyboard interface to the accumulator. The transfer is controlled by a Busy or Not Ready condition within the keyboard controls and can be executed in two modes, Stall on Busy, and Skip on Busy. When executed in the Stall if Busy mode, the program stalls at the transfer byte instruction until a keyboard sprocket is generated indicating that a byte is ready to be transferred. In the Skip on Busy mode, the program automatically skips the next sequential instruction if a character has not been received.

NOTE: The Accumulator value is destroyed after execution of a "Skip if Busy" Instruction.

Ø13 BEEP

This command will produce an electronic beep. This may be used for feedback to the operator after a keystroke, an error tone, etc.

Ø16 LOAD STATUS

This command will load a status word to the ACC. The Ø bit signals a keyboard error. The other bits reflect the I/O and current tape status.

DPL-1 CLASS 1. DISPLAY CONTROL

IOC

OBJECT	SOURCE
174-FFF	IOC, C#4; FFF.

WHERE: FFF is the function code.

DESCRIPTION:

The function code has the following structure:

FFF = SS-LIU-DLM

WHERE: S is the section bits of the page to be displayed.
L is the level bits of the page number to be displayed.
I is the interleave bit in 8 line display mode and the half page (zone) bit in 4 line display mode. See note for def. of interleave.
U is the underscore bit. When U = 1, any display character with a bit 6 will be underscored.
D is the disable CRT bit.
M is the 4/8 line mode select bit. If M = Ø, the 8 line display mode is selected.

TIMING: 4 Microseconds.

EXAMPLE:

174-ØØØ 8 line normal mode - page Ø
174-Ø2Ø 8 line interlace mode - page Ø
174-ØØ1 4 line odd zone (zone 1) - (PØØ-2ØØ thru PØØ-377).
174-Ø21 4 line even zone (zone Ø) -(PØØ-ØØØ thru PØØ-177).

Note on Interleave: (8-line option only)

In the normal display mode (not interleaved), a page will be displayed in a continuous fashion, location ØØØ through 377, octal notation.

Line 1	Loc. ØØØ Octal	through	Loc. Ø37 Octal
Line 2	Loc. Ø4Ø Octal	through	Loc. Ø77 Octal
Line 3	Loc. 1ØØ Octal	through	Loc. 137 Octal
Line 4	Loc. 14Ø Octal	through	Loc. 177 Octal
Line 5	Loc. 2ØØ Octal	through	Loc. 237 Octal
Line 6	Loc. 24Ø Octal	through	Loc. 277 Octal
Line 7	Loc. 3ØØ Octal	through	Loc. 337 Octal
Line 8	Loc. 34Ø Octal	through	Loc. 377 Octal

The interleave mode will display this information in the following sequence:

Line 1	Loc. ØØØ Octal	through	Loc. Ø37 Octal
Line 5	Loc. 2ØØ Octal	through	Loc. 237 Octal
Line 2	Loc. Ø4Ø Octal	through	Loc. Ø77 Octal
Line 6	Loc. 24Ø Octal	through	Loc. 277 Octal
Line 3	Loc. 1ØØ Octal	through	Loc. 137 Octal
Line 7	Loc. 3ØØ Octal	through	Loc. 337 Octal
Line 4	Loc. 14Ø Octal	through	Loc. 177 Octal
Line 8	Loc. 34Ø Octal	through	Loc. 377 Octal

EXAMPLE: Display Page Ø9 (Octal Page 11)

	Comments
IOC, C#4;123.	4 lines from loc. ØØØ to 177, No Underscore.
IOC, C#4;113.	4 lines from loc. 2ØØ to 377, With Underscore.
IOC, C#4;102.	8 lines from loc. ØØØ to 377.
IOC, C#4;112.	8 lines from loc. ØØØ to 377, With Underscore.
IOC, C#4;122.	8 lines Interleaved
IOC, C#4;132.	8 lines With Underscore and Interleaved.

APPENDIX

DPL-1 INSTRUCTION SET

Class 0 **Jump Instructions**

Mnemonic
TLJ+
TLJ–

Timing
4 us Jump, 3* us NO Jump

Description
Test Literal and Jump
Compare the instruction Literal to the Accumulator. On comparison equal jump +NNNN. On comparison not equal execute next instruction.

Binary Format

	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
TLJ+	0	0	0	N	N	N	N	0	L	L	L	L	L	L	L	L
TLJ–	0	0	0	N	N	N	N	1	L	L	L	L	L	L	L	L

Mnemonic
TMJ+
TMJ–

Timing
4 us Jump, 3* us NO Jump

Description
Test Mask and Jump
Compare the instruction Mask to the Accumulator. On comparison equal jump +NNNN. On comparison not equal execute next instruction. Mask logical ones are only bits compared.

Binary Format

	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
TMJ+	0	0	1	N	N	N	N	0	M	M	M	M	M	M	M	M
TMJ–	0	0	1	N	N	N	N	1	M	M	M	M	M	M	M	M

Notes
1. Condition register set for +, –, = compare.
2. Jump past section boundary allowed.
3. N = Jump Count
   L = Literal
   M = Mask

Mnemonic
TLX
TMX

Timing
4 us

Description
Test literal and exit, test mask and exit.
Compare the instruction literal/instruction mask to the accumulator. On comparison equal, exit. On comparison not equal, execute next instruction. Mask logical ones are only bits compared.

Binary Format

	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
TLX	0	0	0	0	0	0	0	0	L	L	L	L	L	L	L	L
TMX	0	0	1	0	0	0	0	0	M	M	M	M	M	M	M	M

Notes:
1. Condition register set for +, –, = compare.
2. L = literal
   M = mask

*If the instruction is located at the low order address of any page, 1 uSec is added to the instruction time to propagate the carry of the +2 add to the high order portion of the address.

Class 1 **Branch and I/O Instructions**

Mnemonic
BRU
BRE
BRH
BRL

Timing
4 us Branch, 3* us NO Branch

Description
Branch Unconditional
Branch on Equal
Branch on High
Branch on Low
On condition, branch directly to the 11 bit address carried in the instruction. Condition register previously set by a Jump or Compare instruction. The 11 bits of the direct address replace the least significant 11 bits of the current IAW.

Binary Format

	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
BRU	0	1	0	0	0	A	A	A	A	A	A	A	A	A	A	0
BRE	0	1	0	0	0	A	A	A	A	A	A	A	A	A	A	1
BRH	0	1	0	0	1	A	A	A	A	A	A	A	A	A	A	0
BRL	0	1	0	0	1	A	A	A	A	A	A	A	A	A	A	1

Notes 1. The least significant bit of the direct address is assumed to be zero and that bit in the instruction is used as part of the operation code.

Class 1 **Branch and I/O Instructions**

Mnemonic
SBU
SBE
SBH
SBL

Timing
4 us Branch, 3* us NO Branch

Description
Stack and Branch Unconditional
Stack and Branch on Equal
Stack and Branch on High
Stack and Branch on Low
On condition, increment the stack pointer, store the 11 bit direct address carried by the instruction into the least significant 11 bits of the new IAS member and branch to the resulting IAW. The condition register is set by a previous Jump or Compare instruction.

Binary Format

	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
SBU	0	1	0	1	0	A	A	A	A	A	A	A	A	A	A	0
SBE	0	1	0	1	0	A	A	A	A	A	A	A	A	A	A	1
SBH	0	1	0	1	1	A	A	A	A	A	A	A	A	A	A	0
SBL	0	1	0	1	1	A	A	A	A	A	A	A	A	A	A	1

Notes 1. The least significant bit of the direct address is assumed to be zero and that bit in the instruction is used as part of the operation code.

2. A = Address

Class 1 — **Branch and I/O Instructions**

Mnemonic: EXU, EXB

Timing: 4 us

Description:
Exit Unconditional
Exit and Branch
The exit instructions decrement the Stack Pointer and return program control to the previous IAS position. For EXU the IAW in that position is used. For EXB the 11 least significant bits of the IAW in that position are replaced by the 11 bit direct address carried in the instruction.

Binary Format: 7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0

EXU	0 1 1 0 0 0 0 0	0 0 0 0 0 0 0 0
EXB	0 1 1 1 0 A A A	A A A A A A A 0

Notes 1. The least significant bit of the direct address is assumed to be zero and that bit in the instruction is used as part of the operation code.

2. A = Address

Class 1 — **Branch and I/O Instructions**

Mnemonic: SMS

Timing: 4 us

Description:
Set memory section
Provides a means of transfering control from the current section to an outside section.

Binary Format: 7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0

SMS	0 1 1 0 1 0 0 0	0 0 S S S 0 0 0

Notes 1. S is the section bits defining the section that control will be transfered to.

*If the instruction is located at the low order address of any page, 1 uSec is added to the instruction time to propagate the carry of the +2 add to the high order portion of the address.

Class 1 — **Branch and I/O Instructions**

Mnemonic: SMC, SSC

Timing: 4 us

Description:
Set memory control.
Set memory section and control.
When the U bit is set to 0, the address of the index registers is memory location 1-7 and direct addressing is only available in page 0 of section 0. When the U bit is set to 1, the effective index register address is location 1-7 of the section where the indexed instruction is being executed. Likewise the effective direct address is page 0 of the section where the direct address instruction is being executed.
When the V bit is set to 1 any branch, stack & branch, or exit & branch instructions given with page 0 specified in the branch address will cause the branch to occur with the current section and page of the program. If any page other than 0 is specified in the branch address, the V bit control is inactive and a normal branch will occur.

Set memory section and control is a combination of set memory section and set memory control instructions.

Binary Format: 7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0

SMC	0 1 1 0 1 0 0 1	U V 0 0 0 0 0 0
SSC	0 1 1 0 1 0 1 0	U V S S S 0 0 0

Class 1 — **Branch and I/O Instructions**

Mnemonic: SAC

Timing: 4 us

Description:
Set arithmetic condition.
Arithmetical conditions of the processor will be forced to a +, −, = condition dependent upon the state of Acc. bits 4 & 5. 00 sets −, 01 sets +, 10 sets =, and 11 is invalid.

Binary Format: 7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0

SAC	0 1 1 0 1 0 1 1	0 0 0 0 0 0 0 0
Acc. (force +)	0 0 0 1 0 0 0 0	
Acc. (force −)	0 0 0 0 0 0 0 0	
Acc. (force =)	0 0 1 0 0 0 0 0	

Class 1 **Branch and I/O Instructions**

Mnemonic LSW

Timing 4 us

Description Load sense switches.
The state of 8 toggle switches (located in the switch well under the CRT screen) to the accumulator.

Binary Format	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
LSW	0	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0

Class 1 **Branch and I/O Instructions**

Mnemonic DPI
EPI
CPI

Timing 4 us

Description Disable processor interrupt.
Enable processor interrupt.
Clear processor interrupt.
The automatic stack and branch that results from an interrupt is program enabled or disabled. The interrupt overflow indicator can be reset by the clear instruction.

Binary Format	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
DPI	0	1	1	0	1	1	1	0	0	0	0	0	0	0	0	0
EPI	0	1	1	0	1	1	1	0	0	0	0	0	0	0	0	1
CPI	0	1	1	0	1	1	1	0	0	0	0	0	0	0	1	0

Class 1 **Branch and I/O Instructions**

Mnemonic IOC

Timing 3* us

Description Input/Output Control
This instruction is used for all input and output operations. The IWL is used to designate the I/O sub-class and to pick the I/O device. The IWR designates the function to be performed.

Binary Format	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
IOC	0	1	1	1	1	n	n	n	y	y	x	x	x	x	x	x

Notes 1. n = Device designation
y, x = Command micro-code

2. Appendix C gives detailed listing of all IOC commands.

Class 1 **Branch and I/O Instructions**

Mnemonic L P S Load Processor Status

Timing 3* us

Description Execution of this command transfers a hardware status word to the accumulators.

Binary Format	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
L P S	0	1	1	0	1	1	0	1	0	0	0	0	0	0	0	0

	ACC BIT	
	0	Stack Pointer Address Bit 2^1
	1	Stack Pointer Address Bit 2^2
2	2	Stack Pointer Address Bit 2^3
	3	Stack Pointer Address Bit 2^4
	4	Plus Condition
	5	Equal Condition
	6	Interrupt Overflow
	7	Program Interrupt Switch

*If the instruction is located at the low order address of any page, 1 uSec is added to the instruction time to propagate the carry of the +2 add to the high order portion of the address.

Class 2 **Transfer and Arithmetic Instructions**

Mnemonic
L D X
L D A
S T A

Timing
4* us for Immediate Add.
5 us for Direct Addressing
6 us for Indexed Addressing

Description
Load Index register
Load Accumulator
Store Accumulator
Specified index register is loaded with a literal carried in the instruction. The accumulator is loaded using immediate, direct or indexed addressing modes. The accumulator is stored in a direct or indexed address. In indexed addressing modes the specified index register may be automatically incremented or decremented.

Binary Format

		7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
	L D X	1 0 0 0 0 X X X	L L L L L L L L
(L A)	L D A	1 0 0 0 0 0 0 0	L L L L L L L L
(D A)	L D A	1 0 0 0 1 0 0 0	A A A A A A A A
(I A)	L D A	1 0 0 0 1 X X X	A A A A A A Y Y
(D A)	S T A	1 0 0 1 1 0 0 0	A A A A A A A A
(I A)	S T A	1 0 0 1 1 X X X	A A A A A A Y Y

Notes: 1. X = index register number
L = literal
A = address
Y = index modifier

2. LA = Immediate Addressing
DA = Direct Addressing
IA = Indexed Addressing

3. Direct address 00000000 is invalid.

Mnemonic
LIA

Timing
4* us

Description
Load instruction address.
This instruction will transfer the 8 least significant bits of the current instruction address to the specified index register. If the instruction literal is 000, then the section and page of the current instruction address is transfered to the accumulator. If the literal is not 000, then the literal is transfered to the accumulator.

Binary Format

	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
LIA	1 0 0 1 0 X X X	L L L L L L L L

Notes: 1. X = Index Register number
2. L = Literal
3. A = Address

*If the instruction is located at the low order address of any page, 1 uSec is added to the instruction time to propagate the carry of the +2 add to the high order portion of the address.

Class 2 **Transfer and Arithmetic Instructions**

Mnemonic
A D X
A D A
S U X
S U A

Timing
4* us for Immediate Addressing
5 us for Direct Addressing
6 us for Indexed Addressing

Description
Add to Index register
Add to Accumulator
Subtract from Accumulator
Specified index register is operated on with the literal carried in the instruction. The accumulator operations specify the operand by immediate, direct or indexed addressing. In indexed addressing the specified index register may be automatically incremented or decremented.

Binary Format

		7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
	A D X	1 0 1 0 0 X X X	L L L L L L L L
(L A)	A D A	1 0 1 0 0 0 0 0	L L L L L L L L
(D A)	A D A	1 0 1 0 1 0 0 0	A A A A A A A A
(I A)	A D A	1 0 1 0 1 X X X	A A A A A A Y Y
	S U X	1 0 1 1 0 X X X	L L L L L L L L
(L A)	S U A	1 0 1 1 0 0 0 0	L L L L L L L L
(D A)	S U A	1 0 1 1 1 0 0 0	A A A A A A A A
(I A)	S U A	1 0 1 1 1 X X X	A A A A A A Y Y

Notes 1. L = literal
A = address
X = index register
Y = index modifier

Class 3 **Boolean and Compare Instructions**

Mnemonic

A N A
S A N
E R A
S E R
I R A
S I R

Timing

4* us

Description

AND to Accumulator
Shift and AND to Accumulator
EXCLUSIVE OR to Accumulator
Shift and EXCLUSIVE OR to Accumulator
INCLUSIVE OR to Accumulator
Shift and INCLUSIVE OR to Accumulator

OPERAND	*Accumulator*	*AND*	*Result EOR*	*IOR*
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	0	1

All shift instructions are right circular and literal addressing only. Remaining instructions use literal, direct or effective addressing. In indexed addressing mode, the specified index register may be incremented or decremented. Shifts take place prior to logical operation.

Binary Format

		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
(L A)	A N A	1	1	0	0	0	0	0	0	L	L	L	L	L	L	L	L
(D A)	A N A	1	1	0	0	1	0	0	0	A	A	A	A	A	A	A	A
(I A)	A N A	1	1	0	0	1	X	X	X	A	A	A	A	A	A	Y	Y
	S A N	1	1	0	0	0	S	S	S	L	L	L	L	L	L	L	L
(L A)	E R A	1	1	0	1	0	0	0	0	L	L	L	L	L	L	L	L
(D A)	E R A	1	1	0	1	1	0	0	0	A	A	A	A	A	A	A	A
(I A)	E R A	1	1	0	1	1	X	X	X	A	A	A	A	A	A	Y	Y
	S E R	1	1	0	1	0	S	S	S	L	L	L	L	L	L	L	L
(L A)	I R A	1	1	1	1	0	0	0	0	L	L	L	L	L	L	L	L
(D A)	I R A	1	1	1	1	1	0	0	0	A	A	A	A	A	A	A	A
(I A)	I R A	1	1	1	1	1	X	X	X	A	A	A	A	A	A	Y	Y
	S I R	1	1	1	1	0	S	S	S	L	L	L	L	L	L	L	L

Notes 1. L = literal
A = address
X = index register
Y = index modifier
S = shift count

2. Direct address of 0 0 0 0 0 0 0 0 is invalid.

*If the instruction is located at the low order address of any page, 1 uSec is added to the instruction time to propagate the carry of the +2 add to the high order portion of the address.

Class 3 **Boolean and Compare Instructions**

Mnemonic

C P A
C P X

Timing

4* us Direct Address
6 us Indexed Address

Description

Compare Accumulator
Compare Index Register
The CPX instruction compares the contents of the specified index register to the literal carried in the instruction. The CPA instructions compare the contents of the Accumulator to a literal or to the contents of a direct or indexed address. In the indexed addressing mode the index register may be incremented or decremented. All comparison results are stored in the Condition Register as high, low or equal.

Binary Format

		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
	C P X	1	1	1	0	0	X	X	X	L	L	L	L	L	L	L	L
(L A)	C P A	1	1	1	0	0	0	0	0	L	L	L	L	L	L	L	L
(D A)	C P A	1	1	1	0	1	0	0	0	A	A	A	A	A	A	A	A
(I A)	C P A	1	1	1	0	1	X	X	X	A	A	A	A	A	A	Y	Y

Notes 1. L = literal
X = index register
A = address
Y = index modifier

10C COMMANDS

Class 1	**Branch and I/O Instructions**
Mnemonic	IOC
Timing	3* us
Description	Input/Output Control This instruction is used for all input and output operations. The IWL is used to designate the I/O sub-class and to pick the I/O device. The IWR designates the function to be performed.
Binary Format	7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0
IOC	0 1 1 1 1 n n n y y x x x x x x

Definition of nnn:

nnn	I/O sub-class
0	current tape channel
1	tape channel 1
2	tape channel 2
3	keyboard
4	CRT
5	coaxial interface
6	communications interface

Definition of yxx for the tape channel:

yxx	Function
000	start tape fwd, slow, erase
001	start tape fwd, slow
002	start tape fwd, fast
003	start tape rev, slow
004	start tape rev, fast
005	stop tape
007	transfer byte
207	transfer byte, skip next instruction if busy
010	write byte
011	read byte
012	rewind
016	read status

The Read Status instruction will transfer a status word to the accumulator. This is structured as follows:

Acc bit	Meaning
0	keyboard error
1	tape error
2	I/O status
3	runaway
4	cartridge out
5	clip out
6	end of tape
7	spare

Definition of yxx for the keyboard channel:

yxx	Function
007	transfer byte
207	transfer byte, skip next instruction if busy
013	keyboard beep

Definition of yxx for the CRT channel:

yxx for the CRT has the following structure:

S S P I U D P M

WHERE:
- S is the section bits of the page to be displayed
- P is the page bits of the page to be displayed
- I is the interleave bit in 8 line display mode and the half page (zone) bit in 4 line display mode.
- U is the underscore bit. When U = 1, any display character with a bit 6 will be underscored.
- D is the disable CRT bit
- M is the 4/8 line mode select bit. If M = 0, the 8 line mode is selected.

If the character to be displayed has a 2_7 bit, this character position will be blanked.

Definition of yxx for the coaxial interface:

yxx	Function
000	start transmit
001	receive byte
201	receive byte, skip next instruction if busy
002	transmit data byte
003	transmit control byte
004	stop transmit
006	inhibit line
007	set device address
010	set master mode
011	set slave mode

Definition of yxx for the communications interface:

yxx	Function
000	transfer acc. to queue reg.
001	select comm. interface mode
002	transfer queue to reg. to acc.
004	present status

December 1972

Manual No. S-100-2

COGAR INFORMATION SYSTEMS, INC.

COSBY MANOR ROAD UTICA, NEW YORK 13502 (315) 797-5750