IMLAC CORP.
4130 LINDEN AVE.
DAYTON, OHIO 45432

IMLAC® PDS-1D

PROGRAMMING GUIDE

CONTENTS

Preliminary Notice

This is a temporary document, to serve until the new PDS-1D User Reference Manual is completed.

Although this document has some deficiencies, it will serve to answer most questions.

PROCESSOR DESCRIPTION

The PDS-1 operates as a dual processor machine, in which both processors share the same core memory. The display processor, normally refreshing the CRT at 40 frames per second, has priority over the main processor, which is a conventional 16 bit mini-computer. The display processor has its own instruction set and instruction decoding capability and "steals" memory cycles from the mini-computer when necessary.

The flexibility of the IMLAC system derives principally from its mini-computer. A standard PDS-1 incorporates 4096 words of 16 bit core storage, some of which is used for display generating and display information storage purposes. Core expansion to 32,768 words is optionally available in 4096 word increments.

Control Description

The control logic serially directs the machine into fetch, defer, and/or execute cycles, as determined by the code of an instruction word. The following is a generalized description of the machine operations which occur during each of these cycles.

> Notation of the form $C(X)_{j-m} \Rightarrow C(Y)_{n-r}$ is used extensively throughout the instruction descriptions and is interpreted as follows: The content of bits j through m of register X becomes the content of bits n through r of register Y.

FETCH CYCLE

During a Fetch Cycle a 16 bit instruction word is fetched from core memory and decoded. This is accomplished by loading the current contents of the Program Counter (PC) register into the Memory Address (MA) register. The entire 16 bit content of the addressed location is read from core memory into the Memory Buffer (MB). Bits 1-4 of the MB are then inclusive OR'd into the Instruction Decode (IR) register. Bit 0 is inclusive OR'd into the Indirect Address (ID) register. At the end of the fetch cycle, one of the following events occurs, depending on IR and ID register content:

1. If the Instruction fetched does not reference memory, i.e. if it is an Operate or IOT instruction, then the instruction is carried out and the Fetch cycle is reset.
2. If the Instruction does reference memory and if the ID register is set, the machine enters a Defer Cycle.
3. If the Instruction does reference memory and if the ID register is not set, the machine enters an Execute Cycle.

Indirect Addressing

The Defer Cycle may only be entered following the Fetch Cycle and only when Indirect Addressing has been specified. The purpose of the Defer Cycle is to indirectly obtain an absolute 12 bit address (4K system, 13 bits on 8K) from core memory using an effective 12 (4K) bit address synthesized from the 11 bit address field of the Processor Order and the highest order P.C. register bit(s). This additional necessary address bit(s) is bit 4 (3 and 4 with 8K) of the P.C. register which specifies the upper or lower 2K block of memory.

The 12 bit effective address is generated by copying bit 4 of the P.C. register and bits 5-15 of the Buffer register (the 11 bit address field) into bits 4-15 of the M.A. register.

> NOTE: The term "effective address" is used to denote the limited addressing capability of a word so formed, because only 11 of its component bits are uniquely specified. An effective address can directly address only those memory locations within the 2K block of core from which the original Processor Order itself was retrieved. Indirect addressing must be used when a memory location outside the 2K block specified by P.C. register bit 4, is to be referenced.

To use indirect addressing, the unique 12 bit address of the data word to be processed is stored in one of the 2K directly addressable locations which may be referenced by an "effective address". Bit 0 of the Processor Order is set to a 1, enabling the Defer Cycle, which generates the "effective address", references core, and interprets the word thus retrieved not as the operand (data word) itself but as the absolute address of the actual operand. With this mechanism, it is possible to indirectly address up to 32K of core memory.

However, because indirect addressing requires that core be referenced for this additional cycle between instruction decoding (Fetch Cycle) and data word processing (Execute Cycle), an additional 1.8μsec is consumed per instruction.

This discussion is clearer if the word "pointer" is used in place of "effective", and the word "effective" is used in place of "absolute". The resulting memory address is a 15-bit value, not 12 bit.

Auto-Index Registers

The PDS-1 has 8 auto-index registers in locations 10_8-17_8 of each 2K block of core memory. For example, in a 4K system these registers would occupy locations 0010_8-0017_8 and 4010_8-4017_8.

Whenever these registers are indirectly addressed, their address word content is incremented by 1 in the memory buffer and then rewritten in core. For example, if memory location 15_8 contains the number 466_8 and is indirectly addressed via the instruction LAC * 15 , the contents of 15_8 is incremented to 467_8 written back in 15_8 and the content of location 467_8 is loaded into the accumulator.

EXECUTE CYCLE

The Execute Cycle may be entered following either a Fetch or Defer Cycle. If entered following a Fetch Cycle, (no Indirect Addressing specified), then an effective address must be generated in the manner detailed under the Defer Cycle description. Again an "effective address" may be used only to reference the current block of core for the data word to be operated on.

If the Execute Cycle was set after a Defer Cycle, however, the absolute address obtained during the Defer Cycle is now transferred to the M.A. register and is capable of referencing any core location for the data word to be processed.

In either case, the word so referenced is processed according to the instruction decoded during the Fetch Cycle. The Fetch Cycle is then reset.

INSTRUCTION TIMING:

All references in this manual to 1.8 usec refer to the instruction cycle time for the newer-model PDS-1D. For the earlier-model PDS-1, the instruction cycle time is 2.0 usec, and "2.0" should therefore be substituted for "1.8", where ever it appears.

Note that in the following sections, where instructions are defined which relate to specific peripheral devices, such as a paper tape reader, a paper tape punch, or a light pen, the System must be configured with the particular device, in order for the instructions relating to it to be meaningful.

MINI-PROCESSOR HIGHLIGHTS

FEATURE	DESCRIPTION
MEMORY	
Word Size (bits)	16
Memory Size (words)	4K standard; 32K optional, by 4K sections
Cycle Time (μ seconds)	2.0 for standard PDS-1; 1.8 for PDS-1D
Direct Addressing (words)	2K
Indirect Addressing (words) (set bit zero)	up to 32K Single Level
CPU	
General Purpose Registers	1 + Link bit for Multiple precision
Index Registers	8 auto-index in locations 10_8 - 17_8 in each 2K block of memory
I/O (Parallel)	
I/O Word Size (bits)	16
Priority Interrupt Levels	1; 2 with DMA-1 option
I/O Maximum Word Rate	100K words without checking (without DMA-1)
OTHER FEATURES	
Power Fail/Restart	standard
Core Protect	standard feature; special circuitry prevents any modification in core memory due to power interruption. Thus if display is on the screen when power is lost, display will reappear with restoration of power.
Read Only Memory Bootstrap (ROM)	optional, up to 40_8 instructions
Real Time Clock	optional

INSTRUCTION REPERTOIRE

CATEGORY	MNEMONIC	OCTAL CODE	MEMORY CYCLES*	DESCRIPTION
Processor Orders	LAW N	004 nnn	1 F	N ⇒C(AC)
	LWC N	104 nnn	1 F	-N ⇒C(AC)
One additional memory cycle (defer cycle) is required when indirect addressing is specified.	JMP Q	010 qqq	1 F	Q ⇒C(PC)
	DAC Q	020 qqq	2 F&E	C(AC) ⇒C(Q)
	XAM Q	024 qqq	2 F&E	C(AC) ⇒C(Q); C(Q) ⇒C(AC)
	ISZ Q	030 qqq	2 F&E	C(Q)+1⇒C(Q); if C(Q)=0, C(PC)+1⇒C(PC)
	JMS Q	034 qqq	2 F&E	C(PC) ⇒C(Q); Q+1⇒C(PC)
	AND Q	044 qqq	2 F&E	Logical AND AC with C(Q)
	IOR Q	050 qqq	2 F&E	Inclusive OR AC with C(Q)
	XOR Q	054 qqq	2 F&E	Exclusive OR AC with C(Q)
	LAC Q	060 qqq	2 F&E	C(Q) ⇒C(AC)
	ADD Q	064 qqq	2 F&E	C(AC)+C(Q)⇒C(AC) } 2's Complement
	SUB Q	070 qqq	2 F&E	C(AC)-C(Q)⇒C(AC) } 2's Complement
	SAM Q	074 qqq	2 F&E	If C(AC)=C(Q), then C(PC)+1⇒C(PC)
Operate Instructions				
Class 1 (Manipulative)	HLT	000 xxx	1 F	HALT
	NOP	100 000	1 F	NO OPERATION
	CLA	100 001	1 F	CLEAR AC
	CMA	100 002	1 F	1'S COMP AC
	STA	100 003	1 F	CLA & COMP
	IAC	100 004	1 F	INC. AC
	COA	100 005	1 F	+1 ⇒C(AC)
	CIA	100 006	1 F	2'S COMP AC
	CLL	100 010	1 F	CLEAR LINK
	CML	100 020	1 F	COMPLEMENT LINK
	STL	100 030	1 F	SET LINK
	ODA	100 040	1 F	IOR AC with DS
	LDA	100 041	1 F	C(DS) ⇒C(AC)
	CAL	100 011	1 F	CLA & CLL
Class 2 (Shift and Rotate)	RAL N	003 00N	1 F	Rotate AC left, N positions } N=1, 2 or 3
	RAR N	003 02N	1 F	Rotate AC right, N positions } N=1, 2 or 3
	SAL N	003 04N	1 F	Shift AC left, N positions } N=1, 2 or 3
	SAR N	003 06N	1 F	Shift AC right, N positions } N=1, 2 or 3
	DON	003 100	1 F	Turn Display Processor ON

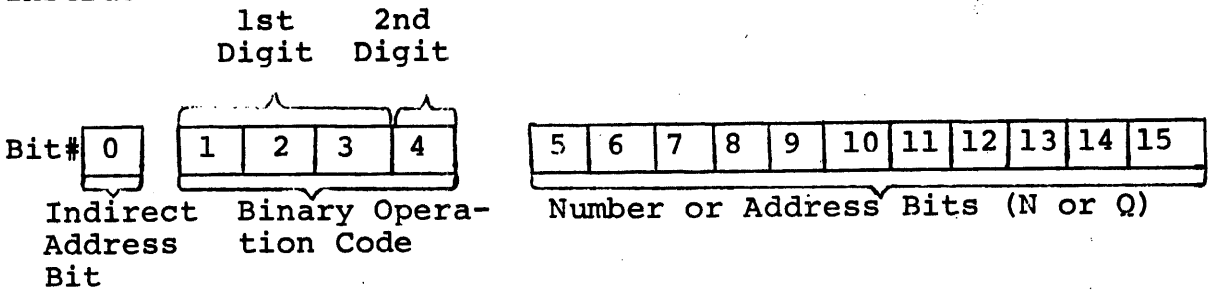
* 1 Memory Cycle requires 1.8 μsec
F - Fetch E - Execute

CATEGORY	MNEMONIC	CODE	CYCLES	DESCRIPTION
OPERATES				
Class 3				Skip if:
(Conditional	ASZ	002 001	1 F	C(AC)=0
Skip)	ASN	102 001	1 F	C(AC)≠0
	ASP	002 002	1 F	C(AC)+
	ASM	102 002	1 F	C(AC)-
	LSZ	002 004	1 F	C(L)=0
	LSN	102 004	1 F	C(L)=1
	DSF	002 010	1 F	Display is on
	DSN	102 010	1 F	Display is off
	KSF	002 020	1 F	Keyboard has input
	KSN	102 020	1 F	Keyboard has no input
	RSF	002 040	1 F	TTY has input data.
	RSN	102 040	1 F	TTY has no input.
	TSF	002 100	1 F	TTY done sending.
	TSN	102 100	1 F	TTY not done sending.
	SSF	002 200	1 F	40 ∿ sync. on.
	SSN	102 200	1 F	40 ∿ sync off.
	HSF	002 400	1 F	PTR has data
	HSN	102 400	1 F	PTR has no data.
IOT				
(partial				
listing)	DLA	001 003	1 F	C(AC) => C(DPC)
	CTB	001 011	1 F	Clear TTY Break
	DOF	001 012	1 F	Turn Display Processor Off
	KRB	001 021	1 F	Keyboard Read
	KCF	001 022	1 F	Keyboard clear flag
	KRC	001 023	1 F	KRB & KCF
	RRB	001 031	1 F	TTY read
	RCF	001 032	1 F	Clear input TTY
	RRC	001 033	1 F	RRB & RCF
	TPR	001 041	1 F	TTY transmit
	TCF	001 042	1 F	Clear TTY Output Status
	TPC	001 043	1 F	TTY print and clear flag
	HRB	001 051	1 F	Read PTR
	HOF	001 052	1 F	Stop PTR
	HON	001 061	1 F	Start PTR
	STB	001 062	1 F	Set TTY Break
	SCF	001 071	1 F	Clear 40 sync.
	IOS	001 072	1 F	IOT Sync.

CATEGORY	MNEMONIC	OCTAL CODE	MEMORY CYCLES	DESCRIPTION
IOT	IOT 101	001 101	1 F	Read Interrupt Status into Accumulator
	IOT 111	001 111	1 F	Protect Core As Per Accumulator
	IOT 131	001 131	1 F	Read Light Pen Register
	IOT 132	001 132	1 F	Clear Light Pen Status
	IOT 134	001 134	1 F	Skip if Light Pen Status = 1
	IOT 141	001 141	1 F	Arm and Disarm Devices
	IOF	001 161	1 F	Disable All Interrupts
	ION	001 162	1 F	Enable All Interrupts
	PUN	001 271	1 F	Punch AC and Clear punch flag (optional)
	PSF	001 274	1 F	Skip if Punch Ready (optional)
Display Processor Orders (partial listing)	DLXA N	01 nnnn	1 F	N => C(XAC)
	DLYA N	02 nnnn	1 F	N => C(YAC)
	DEIM N	03 nnnn	1 F	Increment Mode; N is first byte
	DJMS Q	05 qqqq	1 F	C(DPC)+1 => C(DT), C(Q) => C(DPC)
	DJMP Q	06 qqqq	1 F	Q => C(DPC)
Display Processor Operate (partial listing)	DOPR 15	004 015	1 F	Sensitize Light Pen
	DOPR 14	004 014	1 F	Desensitize Light Pen
	DHLT	000 000	1 F	Halt Display Processor
	DSTS 0	004 004	1 F	Set Scale 1/2
	DSTS 1	004 005	1 F	Set Scale 1
	DSTS 2	004 006	1 F	Set Scale 2
	DSTS 3	004 007	1 F	Set Scale 3
	DSTB 0	004 010	1 F	Set Block 0
	DSTB 1	004 011	1 F	Set Block 1
	DRJM	004 040	1 F	Return Jump, C(DT) => C(DPC)
	DIXM	005 000	1 F	Increment XAC_{msb}
	DIYM	004 400	1 F	Increment YAC_{msb}
	DDXM	004 200	1 F	Decrement XAC_{msb}
	DDYM	004 100	1 F	Decrement YAC_{msb}
	DHVC	006 000	1 F	High Voltage Sync
	DDSP	004 020	1 F	1.8 μsec intensification
	DNOP	004 000	1 F	No Operation

PROCESSOR ORDERS

Instruction Code:



Order Word Decoding

For all Processor Orders except LAW N and LWC N, bit 0, (denoted by an X) specifies indirect addressing when set to a 1. Bits 1 - 4 code the operation to be performed on or with the data word in the memory location specified by the 11 bit address Q. This 11 bit address field can directly address 2K of memory.

NOTE: The following abbreviations are used throughout the instruction descriptions:

Abbreviation	Meaning
D. A.	Direct Addressing
I. A.	Indirect Addressing
N or nnn	11 bit binary number
Q or qqq	11 bit binary address
X	Indirect Addressing Bit

Load Accumulator with N (LAW N)

Load Accumulator with Complement of N (LWC N)

Octal Operation Codes: 004 nnn, 104 nnn

Cycles: Fetch

Operation: LAW N - The accumulator is cleared; The 11 bit binary number N becomes the contents of the accumulator.

LWC N - The accumulator is loaded with the two's complement of N (-N) if bit 0 is a 1.

If Bit 0 = 0 N => C(AC)
If Bit 0 = 1 -N => C(AC)

Jump to Address Q (JMP Q)

Octal Operation Code: X10 qqq

Cycles: D.A. - Fetch
I.A. - Fetch and Defer

Operation: Q becomes the content of the program counter; i.e., the next instruction will be taken from memory location Q:

Q => C(PC)

Deposit Accumulator in Q (DAC Q)
(Does not clear accumulator)

Octal Operation Code: X20 qqq

Cycles: D. A.- Fetch & Execute
I. A.- Fetch, Defer, & Execute

Operation: Content of Accumulator is deposited in memory location Q; Accumulator is not cleared; Original content of Q is lost.

C(AC) => C(Q)

Exchange Accumulator with Memory Location Q (XAM Q)

Octal Operation Code: X24 qqq

Cycles: D. A.- Fetch & Execute
I. A.- Fetch, Defer, & Execute

Operation: Content of Accumulator becomes content of location Q; content of location Q becomes the content of the accumulator. This instruction is useful for inserting data in a display list and rapidly pushing the list down. This is accomplished by letting memory location Q be auto-index register and indirectly addressing it via the XAM instruction.

For Example:

Whenever a command is inserted in an address in the midst of a display list, it is necessary to shift the previous contents of that address and all succeeding addresses "down" by one memory location. For instance, if the following display list were executed by the Display Processor, HLLO would be displayed on the screen (Subroutines H, L, and O are omitted.)

```
100  DJMS H
101  DJMS L
102  DJMS L
103  DJMS O
```

To display HELLO, a DJMS E would have to be inserted in 101 and the succeeding DJMS commands moved down. The following program accomplishes this:

```
        LAW 100;Load ACC with 100.
        DAC 10;Deposit ACC in auto-index register 10.
        LAC INSERT;Load ACC with DJMS E.
        XAM * 10;increment address found in 10
         ;and exchange content of this new
         ;address with content of ACC.
        ISZ CNTR;increment counter and skip if zero.
        JMP .- 2;Jump back and continue pushing
         ;down list.
CNTR:   -4
INSERT: DJMS E
```

Increment Q and SKIP if 0 (ISZ Q)

Octal Operation Code: X30 qqq

Cycles: D.A. - Fetch & Execute
I.A. - Fetch, Defer & Execute

Operation: The data word of location Q is incremented by 1, and the result is rewritten in location Q. Whenever this operation results in a sum of 0,
the Program Counter will be incremented by 1, causing the following instruction to be skipped.

Jump to Subroutine Q (JMS Q)

Octal Operation Code: X34 qqq

Cycles: D.A. - Fetch & Execute
I.A. - Fetch, Defer & Execute

Operation: Store Program Counter Content (present address plus 1) in location Q; continue to execute the program at location Q +1.

$$C(PC) \Rightarrow C(Q)$$
$$Q+1 \Rightarrow C(PC)$$

Logical And (AND Q)

Octal Operation Code: X44 qqq

Cycles: D.A. - Fetch & Execute
I.A. - Fetch, Defer & Execute

Operation: Logically AND the content of location Q, bit by bit, with the corresponding bits of the accumulator. The original accumulator content is lost and the content of Q is undisturbed.

$$AC_j \wedge Y_j \Rightarrow AC_j$$

Inclusive OR (IOR Q)

Octal Operation Code: X50 qqq

Cycles: D.A. - Fetch & Execute
I.A. - Fetch, Defer & Execute

Operation: Logically OR, bit by bit, the contents of the accumulator, with the corresponding bits in memory location Q. The result is left in the accumulator and the original content of Q is undisturbed.

$$AC_j \vee Y_j \Rightarrow AC_j$$

Exclusive OR (XOR Q)

Octal Operation Code: X54 qqq

Cycles: D.A. - Fetch & Execute
I.A. - Fetch, Defer & Execute

Operation: Exclusively OR, bit by bit, the content of the accumulator with the corresponding bits in location Q. The result is left in the accumulator and the original content of Q is undisturbed.

Example:

	Original AC	Content of Q	Final AC
Bit # 1	0	0	0
2	0	1	1
3	1	1	0
4	1	0	1

Load Accumulator with Contents of Q (LAC Q)

Octal Operation Code: X60 qqq

Cycles: D.A. - Fetch & Execute
I.A. - Fetch, Defer & Execute

Operation: Load the accumulator with the content of memory location Q. The original content of the accumulator is lost; the content of Q is undisturbed.

Add to Accumulator, Q (ADD Q)

Octal Operation Code: X64 qqq

Cycles: D.A. - Fetch & Execute
I.A. - Fetch, Defer & Execute

Operation: Perform a binary addition between the contents of the accumulator and location Q and leave the result in the accumulator. If a carry out of bit 0 occurs, the link will be complemented. Q is undisturbed.

C(AC) + C(Q) => C(AC)

Subtract from Accumulator the Contents of Q (SUB Q)

Octal Operation Code: X70 qqq

Cycles: D.A. - Fetch & Execute
I.A. - Fetch, Defer, & Execute

Operation: The two's complement of the content of memory location Q is added to the Accumulator; the result is stored in the accumulator and the content of Q is undisturbed. Link bit is complemented if carry occurs.

C(AC) - C(Q) => C(AC)

Skip if Accumulator Same as Address Q (SAM Q)

Octal Operation Code: X74 qqq

Cycles: D.A. - Fetch & Execute
I.A. - Fetch, Defer & Execute

Operation: Skip the next instruction if content of location Q is same as content of accumulator.

if C(AC) ⇒ C(Q) C(Q) is undisturbed.
then C(PC) +1 ⇒ C(PC);

PROCESSOR MICROINSTRUCTIONS

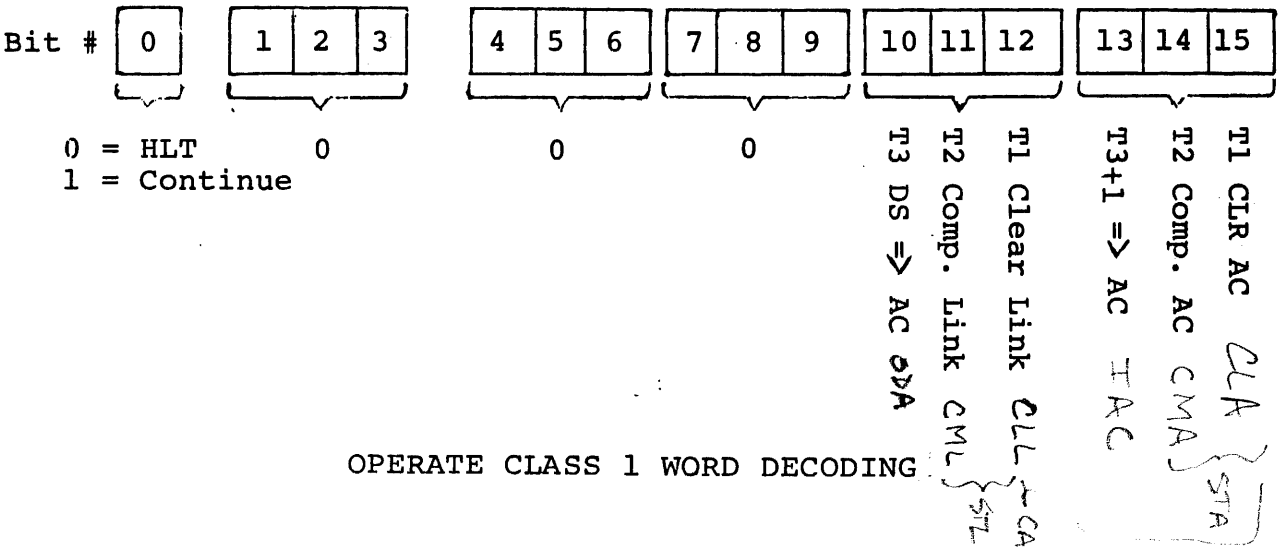
The remaining Mini-computer instructions fall into two categories - Microinstructions and I/O Commands. I/O Commands, which do not reference memory, give the user software control of data transfers between the PDS-1 and a peripheral device. The operate microinstructions (which also do not reference memory) manipulate and/or test data in the Link bit and the Accumulator and are further divided into three classes.

Operate Class 1 (Manipulative)

All Class 1 instructions contain a 1 in bit 0 (except the halt instruction) and 0's in bits 1-9.

The last six bits (10-15) of the operate word independently define 1 instruction apiece, but may be combined, as shown below, when they are not contradictory.

Example: 100005/ COA
+ 100020/CML
100025/ COA, CML +1⇒ C(AC) and Complement link



OPERATE CLASS 1 WORD DECODING

T1,T2 and T3 in "OPERATE CLASS 1 WORD DECODING", refer to the three timing cycles that sequentially occur whenever an operate microinstruction is performed. All operations that are requested during cycle T1 are performed before any T2 operation is started.

For example, an instruction with a bit setting of 100017 will first clear both the accumulator and link simultaneously during cycle T1. Then during cycle T2 the accumulator is complemented and, finally, at time T3 the accumulator is incremented by one, resulting in the link but being set to 1 and the accumulator being cleared.

Cycle by cycle breakdown of instruction 100017:

Cycle	Link Bit	Accumulator
T1	0	000000
T2	0	177777
T3	1	000000

This same result can be obtained by combining the CLA instruction with the STL instruction (100031).

Halt after this Command (HLT)

Octal Operation Code: 000xxx

Cycles: Fetch

Operation: Clears the "run" flip-flop and so stops execution of the program immediately. This instruction may be combined with other operates as the halt function is executed last.

No Operation (NOP)

Octal Operation Code: 100000

Cycles: Fetch

Operation: None; this command is used to accumulate execution time in a program for such purposes as waiting for data from an input device, or as a dummy instruction in a dynamically changing subroutine.

Clear Accumulator (CLA)

Octal Operation Code: 100001

Cycles: Fetch

Operation: Set all accumulator flip-flops (bits 0-15) to binary zeros.

$0 \Rightarrow C(AC)$

Complement Accumulator - 1's Complement (CMA)

Octal Operation Code: 100002

Cycles: Fetch

Operation: The content of the accumulator is set to its ones complement,i.e., every bit which is currently a one is made a zero and vice versa.

$\overline{C(AC)} \Rightarrow C(AC)$

Clear and Complement Accumulator (STA)

Octal Operation Code: 100003

Cycles: Fetch

Operation: The accumulator is first cleared and then all bits (0-15) are set to a binary 1 value. The accumulator then contains a 177777_8 which is equivalent to a -1 (minus one), decimal.

$-1 \Rightarrow C(AC)$

Increment Accumulator (IAC)

Octal Operation Code: 100004

Cycles: Fetch

Operation: Increase the content of the accumulator by 1. If a carry should occur, the link will be complemented.

Clear and Increment Accumulator (COA)

Octal Operation Code: 100005

Cycles: Fetch

Operation: The accumulator is first cleared and then a 1 is added.

$$1 \Rightarrow C(AC)$$

Complement and Increment Accumulator (CIA)
(2's Complement)

Octal Operation Code: 100006

Cycles: Fetch

Operation: First form the one's complement of present accumulator contents and then increment this result. This operation forms the two's complement (the negative) of the original content.

$$\overline{C(AC)} + 1 \Rightarrow C(AC)$$

Clear the Link (CLL)

Octal Operation Code: 100010

Cycles: Fetch

Operation: Set the content of the link to a binary zero.

Complement Link (CML)

Octal Operation Code: 100020

Cycles: Fetch

Operation: Complement the content of the link.

$$\overline{C(L)} \Rightarrow C(L)$$

Clear AC and Clear Link (CAL)

Octal Operation Code: 100011

Cycles: Fetch

Operation: Set all accumulator bits to binary 0's; set link to a binary 0.

Set Link (STL)

Octal Operation Code: 100030

Cycles: Fetch

Operation: Set the content of the link to a binary 1.

Inclusive OR AC with Data Switches (DS) (ODA)

Octal Operation Code: 100040

Cycles: Fetch

Operation: Bit by bit logically OR the content of the accumulator with that of the data switches and leave the result in the accumulator.

$$C(AC_j) \vee C(DS_j) \Rightarrow C(AC_j)$$

Load Content of Data Switches into Accumulator (LDA)

Octal Operation Code: 100041

Cycles: Fetch

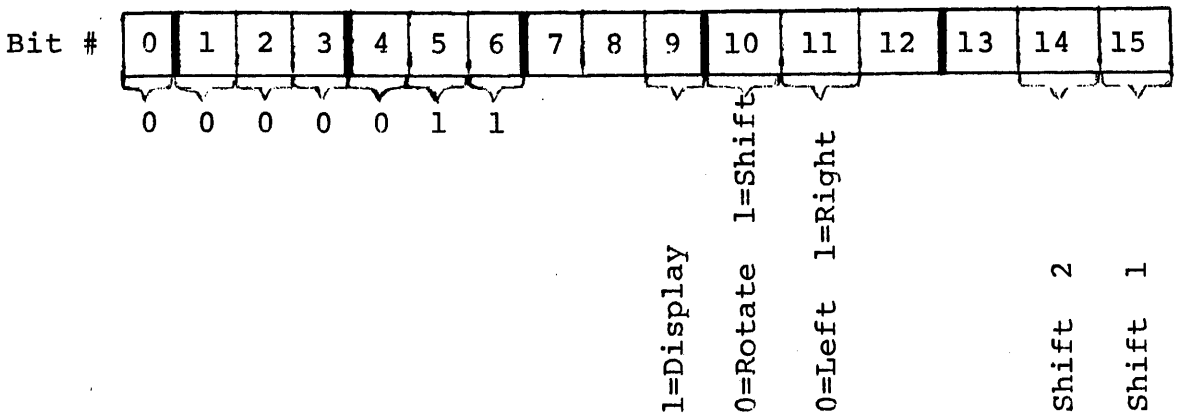
Operation: The content of the data switches becomes the content of the accumulator; original accumulator content is lost.

$$C(DS) \Rightarrow C(AC)$$

Operate Class 2 (Shift and Rotate)

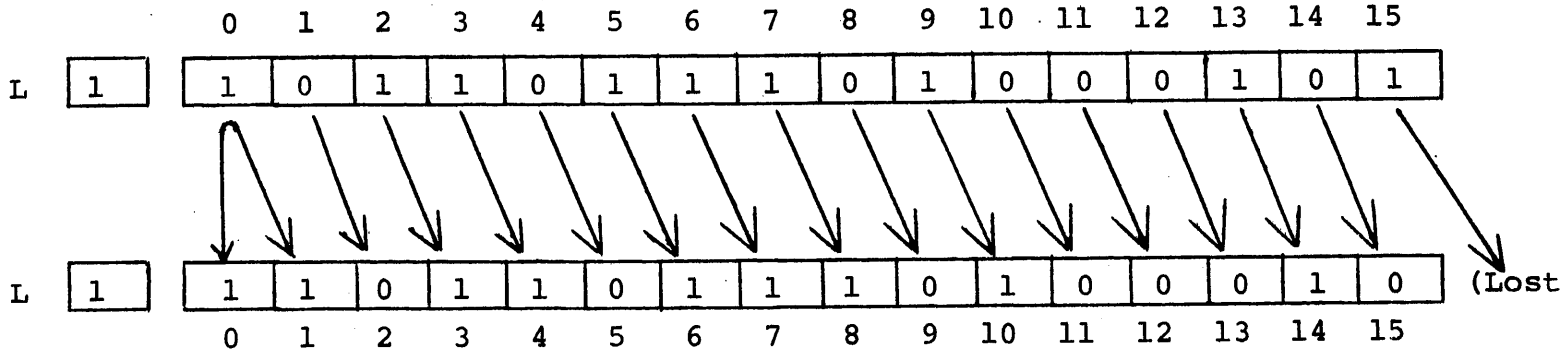
These instructions are actually a specialized supplement to class 1 operators, as they also manipulate the AC and L. Bit 9 turns on the display, 10 determines a rotate or a shift, 11 determines the direction of the latter and 14 and 15 specify the number (N) of shift positions.

The shift operation does not change the link bit or bit zero of the accumulator. Shift right copies the sign bit into bit 1. Shift left puts zero into bit 15 of the accumulator:

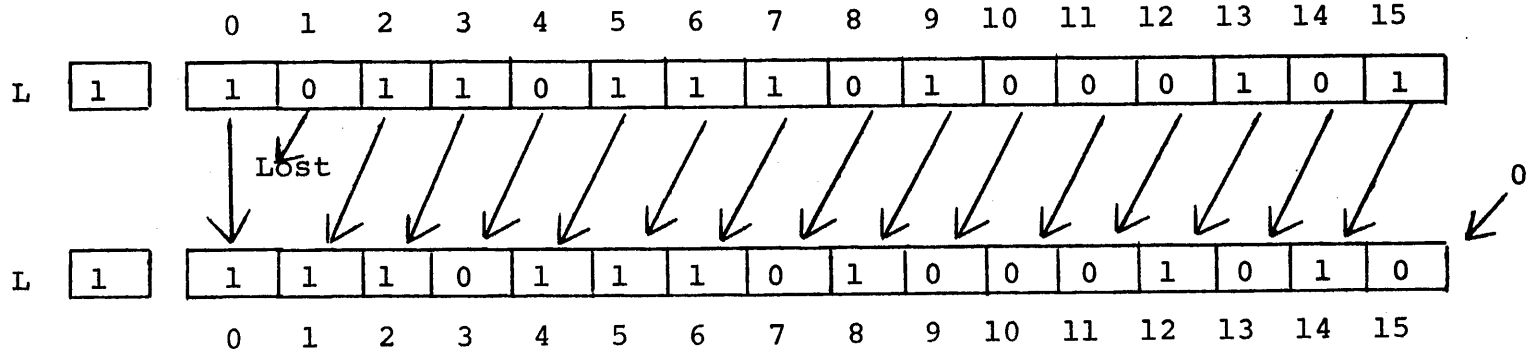


Shift & Rotate Decoding

SHIFT RIGHT



SHIFT LEFT



Shift Left N Times (SAL N)

Octal Code: 00304N where N is 1, 2, or 3

Cycles: Fetch

Operation for N=1: Accumulator bits 2-15 are moved 1 binary position to the left. Bit 1 is lost; bit 0 and the link are unchanged, and a 0 moves into bit 15.

Execute These Operations N Times:

$(AC)j \Rightarrow (AC)j-1$

$C(AC)_0 \Rightarrow C(AC)_0$

$L \Rightarrow L$

Shift Right N Times (SAR N)

Octal Code: 00306N

Cycles: Fetch

Operation for N=1: Accumulator bits 1-14 are moved 1 position to the right. Bit 0 is copied into bit 1. Bit 0 and link are preserved. Bit 15 is lost.

Execute N Times:

$AC_j \Rightarrow AC_{j+1}$

$AC_0 \Rightarrow AC_1$

$L \Rightarrow L$

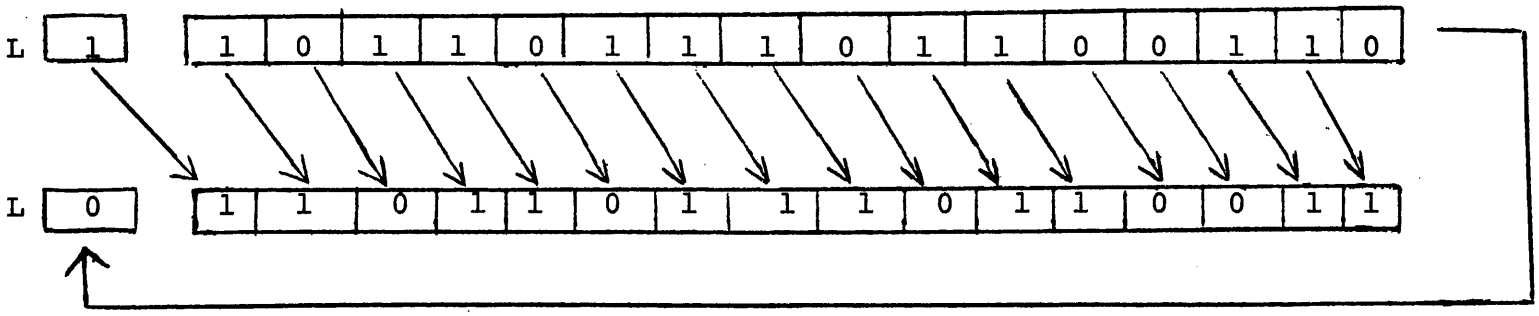
Display ON (DON)

Octal Code: 003100

Cycles: Fetch

Operation: This instruction starts the display processor; the display processor will take the next memory cycle using the address in the display program counter.

ROTATE RIGHT



Rotate Left N Times (RAL N)

Octal Code: 00300N

Cycles: Fetch

Operation: For N=1 The content of bit j becomes the content of j-1. AC_0 goes to the link. The link becomes the content of bit fifteen.

Execute these Operations N Times:

$$\begin{cases} C(AC_j) \Rightarrow C(A_{j-1}) \\ C(AC_0) \Rightarrow C(L) \\ C(L) \Rightarrow C(AC_{15}) \end{cases}$$

Rotate Right N Times (RAR N)

Octal Code: 00302N

Cycles: Fetch

Operation: For N=1 The content of the accumulator and link is moved to the right by 1 binary position. The content of the link becomes the content of AC_0, and the content of bit 15 becomes the content of the link.

Execute these Operations N Times:

$$\{C(AC_j) \Rightarrow C(AC_{j+1}), C(L) \Rightarrow C(AC_0), C(AC_{15}) \Rightarrow C(L)$$

Operate Class 3 (Conditional Skip)

A 1 in bit 0 specifies the complement meaning of the same instruction with a zero in bit zero.

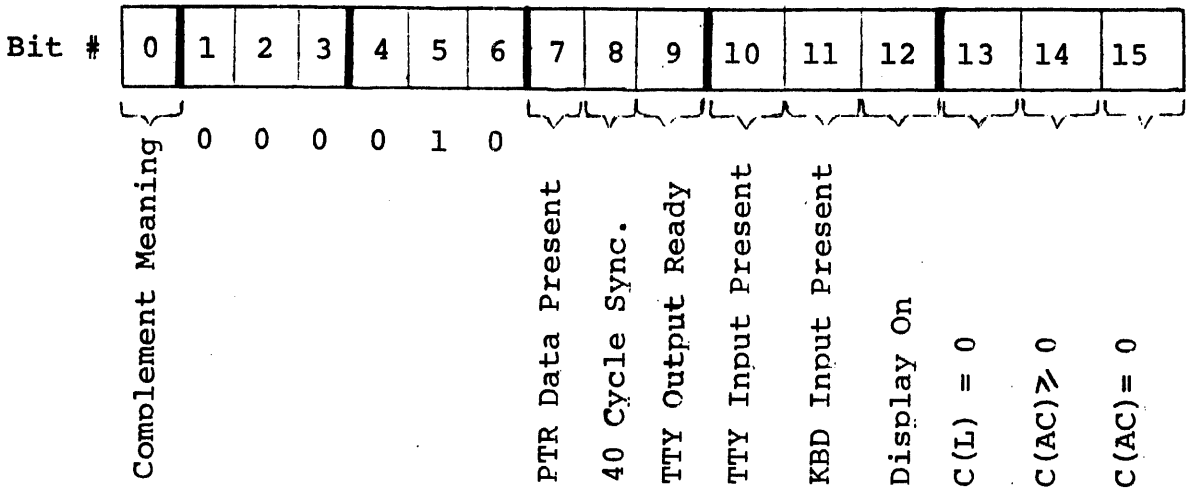
The last 9 bits independently define particular skip instructions. The instructions of this class may be combined in inclusive OR fashion if they are compatible, and if bit 0 has the same value.

For Example:

```
002002/ASP
+ 002004/LSZ
  002006/ASP.LSZ
```

Skip if contents of accumulator are +, or if the link is 0.

C (AC) is undisturbed with the execution of these instructions.



Skip Instruction Decoding

Skip if Accumulator Content is Zero (ASZ)

Skip if Accumulator Content is not Zero (ASN)

Octal Code: 002001, 102001

Cycles: Fetch

Operation for ASZ: If the content of the accumulator is zero then skip the next instruction.

If C(AC) = 0
Then C(PC) +1 => C(PC)

For ASN: Complement of 002001.

Skip if Accumulator Content is + (positive) (ASP)

Skip if Accumulator Content is - (minus) (ASM)

Octal Code: 002002, 102002

Cycles: Fetch

Operation for ASP: If the accumulator contains a positive number, skip the next instruction,

If C(AC) = positive number
Then C(PC)+1 => C(PC)

Operation for ASM: Complement of above.

Skip if Content of Link is Zero (LSZ)

Skip if Content of Link is not Zero (LSN)

Octal Code: 002004, 102004

Cycles: Fetch

Operation for LSZ: If the link is a 0, then skip the next instruction, If C(L)=0, Then C(PC)+1=> C(PC)

Operation for LSN: Complement of above operation.

Skip if Display ON flag is Set (DSF)
Skip if Display ON flag is Not Set (DSN)

Octal Code: 002010, 102010

Cycles: Fetch

Operation for DSN: If the display processor is on, then skip the next instruction.

If Display on then C(PC)+1 => C(PC)

Operation for DSF: Complement of above operation.

Skip if Keyboard Flag is Set (KSF)
Skip if Keyboard Flag Not Set (KSN)

Octal Code: 002020, 102020

Cycles: Fetch

Operation for KSF: Skip the next instruction if the keyboard flag is set, i.e., if there is a character in the keyboard buffer. For an example of use, see description of IOT instruction KRC. (001 023)
If flag = 1
Then C(PC)+1 => C(PC)

Operation for KSN: Complement of above operation.

Skip if TTY has Input Data (RSF)
Skip if TTY has no Input Data (RSN)

Octal Code: 002040, 102040

Cycles: Fetch

Operation for RSF: Skip the next instruction if the TTY buffer has a character. For example of use, see IOT instruction description of RRC. (001 033)

If TTY Receive flag set,
Then C(PC)+1 => C(PC)

Operation for RSN: Complement of above operation.

Skip if TTY has completed Sending (TSF)
Skip if TTY has not completed Sending (TSN)

Octal Code: 002100, 102100

Cycles: Fetch

Operation for TSF: Skip the next instruction if the TTY has finished sending data.
If TTY Transmit flag is set,
Then C(PC)+1 => C(PC)

Operation for TSN: Complement of above operation.

Skip if 40 CPS Sync is ON (SSF)
Skip if 40 Cycle Sync is OFF (SSN)

Octal Code: 002200, 102200

Cycles: Fetch

Operation for SSF: Skip the next instruction if the 40 cycle sync flip flop is set. This command and its complement (102 200) are used in conjunction with IOT instruction SCF (clear 40 cycle sync) as a means of synchronizing the display and central processors. In the operating program, the instruction SCF is given before the display is turned on. Thereafter the main processor checks the status of the 40 CPS sync flip flop with instruction SSF to determine whether or not the display is on.

Operation for SSN: Complement of above operation.

Skip if PTR has Data (HSF)
Skip if PTR has No Data (HSN)

Octal Code: 002400, 102400

Cycles: Fetch

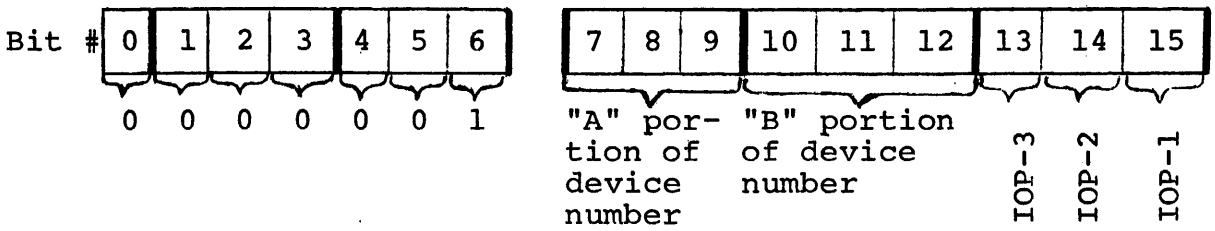
Operation for HSF: Skip the next instruction if the photoelectric tape reader has a character to be sent. An example of the use of 002 400 and 102 400 is given under the description of the operation of IOT instruction HRB (001 051).

Operation for HSN: Complement of above operation.

IOT Instructions

These commands, when used in conjunction with the appropriate processor and skip instructions, allow complete program control of I/O data transfers.

IMPORTANT: PDS-1 users who intend to generate their own I/O commands should first utilize device number 77_8 and work down, as IMLAC is assigning commands from No. 00 upward.



IOT Word Decoding

Bits 0-6 denote an (IOT) instruction. Bits 7-12 specify the device being interacted with, while the values of bits 13, 14 and 15 code the IOP (input-output pulse) to be sent to that device.

Load Display Program Counter with Accumulator Content (DLA)

Octal Code: 001003

Cycles: Fetch

Operation: Load the accumulator content (bits 1-15) into the display program counter.

Clear TTY Break (CTB) Refer to Set Break Instruction 001062

Octal Code: 001 011

Cycles: Fetch

Operation: This command restores the TTY line to "high" or "mark" status. (TTY refers to serial interface which may or may not be connected to a teletype).

Display Off (DOF)

Octal Code: 001012

Cycles: Fetch

Operation: The display processor is halted.

NOTE: Prior to giving any of the following read commands, the accumulator must be cleared because the character being read in is actually inclusive OR'd into the accumulator.

Keyboard Read (KRB)

Octal Code: 001021

Cycles: Fetch

Operation: Read into accumulator bits 5-15 the information found in the 11 bit keyboard input.

Keyboard Clear Flag (KCF)

Octal Code: 001022

Cycles: Fetch

Operation: Clear the keyboard flag in preparation for the next character input.

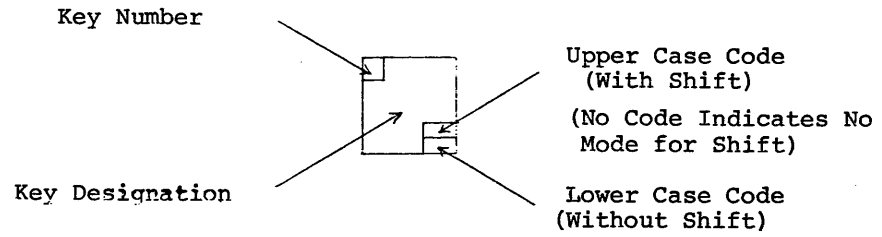
Keyboard Read and Clear Flag (KRC)

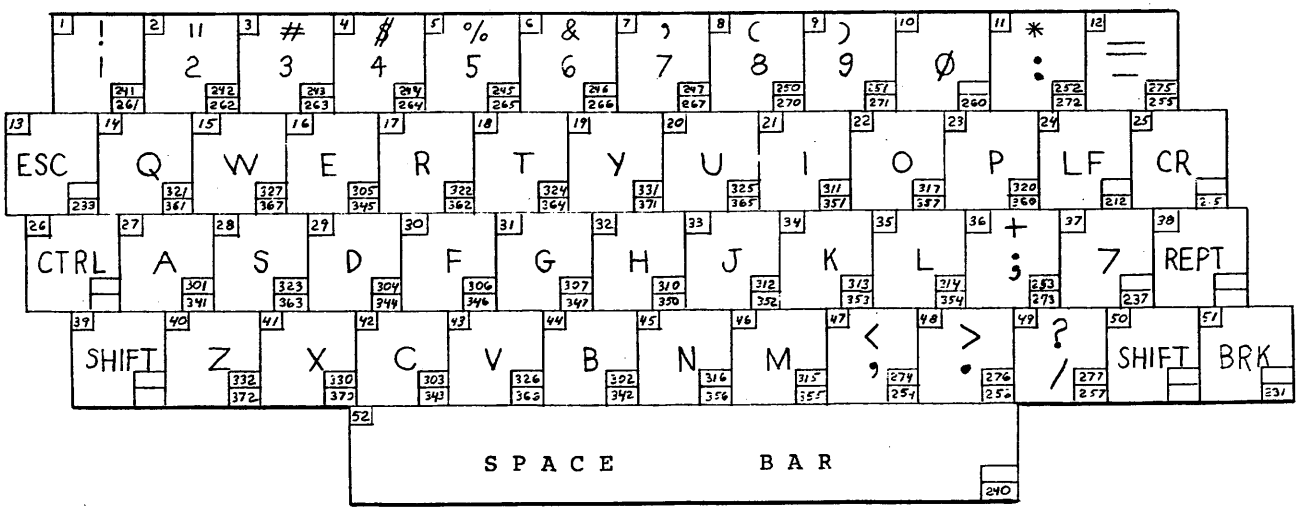
Octal Code: 001023

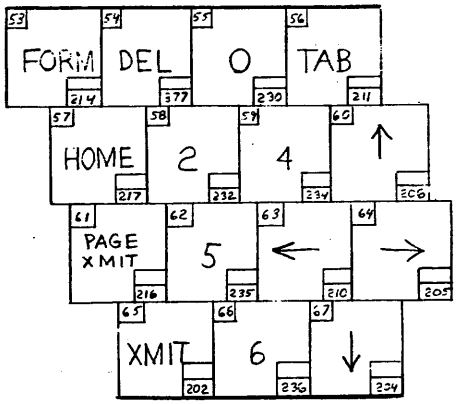
Cycles: Fetch

Operation: Combines the operation of KCF and KRB. After KCF is executed, a new character may be keyed into the keyboard logic, which will set the keyboard flag.

PDS-1D Keyboard







Example of use:

```
                    ; Keyboard Monitor Program

        KSF ; Skip on keyboard flag
        JMP .-1 ; Await character
        CLA ; Clear accumulator
        KRC ; Keyboard read and clear flag
        JMP .-4 ; Prepare to read next character, leave
            ; result in accumulator.
```

TTY Read (RRB)

Octal Code: 001031

Cycles: Fetch

Operation: IOR into the accumulator, the character in the TTY input buffer.

Clear Input TTY Status (RCF)

Octal Code: 001032

Cycles: Fetch

Operation: This clears the TTY buffer flag.

TTY Read and Clear Flag (RRC)

Octal Code: 001033

Cycles: Fetch

Operation: This instruction combines the operations of RCF and RRB.

Example of Use:

```
                    ; Teletype Monitor Program

        RSF ; Skip on TTY flag
        JMP .-1 ; Await character
        CLA ; Clear accumulator
        RRC ; TTY read and clear flag
        JMP .-4 ; Prepare to read next character, leave result
            ; in accumulator.
```

TTY Transmit (TPR)

Octal Code: 001041

Cycles: Fetch

Operation: Transfer C(AC) $_{8-15}$ to TTY output buffer and start transmission.

NOTE: The TPR instruction should never be used without being followed by TCF, as otherwise the output flag would never be cleared.

Clear Output TTY Status (TCF)

Octal Code: 001042

Cycles: Fetch

Operation: Clear the "ready for output" flag in preparation for transmitting a character to the output buffer. This flag is reset by the interface hardware after the character has been transmitted.

TTY Print and Clear Flag (TPC)

Octal Code: 001043

Cycles: Fetch

Operation: This is a combination of TCF and TPR, which clears the TTY output flag and transmits the character presently in the PDS-1's accumulator.

Read PTR (HRB)

Octal Code: 001051

Cycles: Fetch

Operation: IOR into the accumulator the character currently being read from paper tape by the high speed reader. (up to 8 bits of parallel information).

Stop PTR (HOF)

Octal Code: 001052

Cycles: Fetch

Operation: This command causes the PTR transport to halt.

Start PTR (HON)

Octal Code: 001061

Cycles: Fetch

Operation: Start the drive motor of the photoelectric tape reader.

Example of Use: ; Photoelectric Tape Reader Monitor Program

```
HSN ; Skip if PTR has no data
JMP .-1 ; Jump back until PTR has no data
HSF ; Skip if PTR has data
JMP .-1 ; Jump back until PTR has data
CLA ; Clear accumulator
HRB ; Read PTR into accumulator
```

Set TTY Break (STB)

Octal Code: 001062

Cycles: Fetch

Operation: Ground the TTY line (spacing); i.e., the TTY line, normally at high level is shifted to low level. This command allows the user to duplicate, via programming, the hardware "break" function found on many teletypes. Instruction CTB (001 011) clears this break.

Clear 40 Cycle Sync (SCF)

Octal Code: 001071

Cycles: Fetch

Operation: Clear to zero the 40 cycle sync flip-flop. This flip-flop will be automatically set again in 1/40 of one second. For an example of use, see description of skip instruction SSF (002 200).

IOT Sync (IOS)

Octal Code: 001072

Cycles: Fetch

Operation: Not used for regular I/O programs; generates an internal timing pulse. This instruction (which is used for maintenance purposes) is executed in a test program to trigger an oscilloscope.

Skip if Punch Flag is Set (PSF)

Octal Code: 001274

Cycles: Fetch

Operation: If the punch status flag is set to "ready", skip the next instruction.

Punch Accumulator (PPC)

Octal Code: 001271

Cycles: Fetch

Operation: The execution of this command enables the transfer of accumulator bits 8-15 to the punch buffer through the expanded I/O interface.

Read Interrupt Status into Accumulator (IOT 101)

Octal Code: 001101
Cycles: Fetch
Operation: Read (inclusive OR) the interrupt status into the Accumulator. If Bit 15 of the Accumulator is set, then the Light Pen caused the interrupt; if Bit 14 is set, then the 40 cycle sync caused the interrupt, etc.

INTERRUPT STATUS BITS

(IOT 001 101)

AC Bit #	Device
15	Light Pen (LPA-1)
14	40 Cycle SYNC & End of Display Frame
13	Memory Protect (PMP-1)
12	TTY receive
11	Keyboard
10	TTY send
9	Joystick, Mouse, or Trackball (JST-1, GMI-1, or TBL-1)
8	Tablet (TBI-1)
7	Punch (PTP-1)
6	Keyboard #2 (KYB-1)
5	TKA IN } (TKA-1)
4	TKA OUT }
3	16 Bit input/PTR (GSI-1 or PTR-1)
2	Addressable clock with interrupt (ACI-1)
1	
0	Printer (PRT-1)

Arm and Disarm Devices (IOT 141)

Octal Code: 001141
Cycles: Fetch
Operation: The Light Pen is armed if Accumulator bit 15=1; the 40 ~ sync is armed if bit 14=1; etc.

This instruction should be performed upon program startup, in programs which use the interrupt feature.

INTERRUPT ARM / DISARM

(IOT 001 141)

These are the same bit assignments as for Interrupt Status (see IOT 101).

AC Bit #	Device
15	Light Pen
14	40 Cycle SYNC & End of Display Frame
13	Memory Protect
12	TTY Receive
11	Keyboard
10	TTY send
9	Joystick, Mouse, or Trackball
8	Tablet
7	Punch
6	Keyboard #2
5	TKA IN
4	TKA OUT
3	16 Bit input/PTR
2	Addressable Clock with Interrupt
1	
0	Printer

SECOND LEVEL INTERRUPT STATUS BITS

(IOT 001 212)

AC Bit #	Device
15	
14	
13	
12	
11	
10	
9	
8	
7	
6	
5	
4	
3	
2	
1	Magnetic Tape (MTC-1)
0	Disc System (DSC-1)

SECOND LEVEL INTERRUPT ARM/DISARM

(IOT 001 211)

AC BIT #	Device
15	
14	
13	
12	
11	
10	
9	
8	
7	
6	
5	
4	
3	
2	
1	Magnetic Tape
0	Disc System

Disable All Interrupts (IOF)

Enable All Interrupts (ION)

Octal Code: 001161, 001162
Cycles: Fetch
Operation: IOF - disables all interrupts both internal and external.

ION - only enables those interrupts that have previously been armed.

Read Light Pen Register (IOT 131)

Octal Code: 001131
Cycles: Fetch
Operation: Read (inclusive OR) the content of the Light Pen register into the Accumulator. (Bits 1 to 15)

Clear Light Pen Status (IOT 132)

Octal Code: 001132
Cycles: Fetch
Operation: Clear the Light Pen status (ready or not ready) register.

Skip if Light Pen Status = 1 (IOT 134)

Octal Code: 001134
Cycles: Fetch
Operation: Skip the next instruction, if Light Pen status is high or ready.

DESCRIPTION ON USING CASSETTE (CBS) HARDWARE

IOT's 014, 024, 034, 144, and 154 are used with the Serial Bit Cassette (REL-1 for read only, or CBS-1 for read and write). IOT's 144 and 154 are used to start and stop the cassette player/recorder. Following is a description of how IOT's 014, 024, and 034 are used:

The cassette bulk store (CBS) option involves the reading and writing of serial data bit by bit on a cassette recorder which moves the tape at constant speed. This serial bit process differs from other IMLAC input/output devices in two important ways: The data is bit by bit and due to constant tape speed, the density of data is timing dependent.

In general a standard timing loop is used to control the writing of individual bits on the cassette. This timing loop varies depending on the internal clock of the PDS-1D and the cassette interface hardware.

On the tape itself data is written bit by bit left to right from an IMLAC word. For each bit position a timing bit is written on the tape automatically. In addition,for every zero bit one write IOT is issued and for every one bit, two write IOTs are issued. When reading this data, a read IOT will sense each bit position by locating the next timing bit on the cassette tape. At the same instant a data bit is set or reset depending on whether or not the tape had a bit next to the timing bit. Another IOT is used to test this hardware data bit and to set or reset the AC_{15}, the link, a memory location or wherever desired. Thus, the read routine is reconstructing the original word from tape bit by bit from left to right.

CBS DATA FORMAT

A 1 bit is written using 2 write IOT's (001034). The first is followed by a delay of about 350 microseconds and the second is followed by a delay of about 750 microseconds.

A 0 bit is written using 1 write IOT. It is followed by a delay of about 750 microseconds.

A two second startup time works quite well for the Norelco recorders.

Gaps in the data of longer than one second should be followed by a resynchronization pattern.

Following is a listing of a routine that writes an 8 bit character onto a cassette:

```
SEND8: Ø          ; SEND 8 BITS
 RAL 3            ; ROTATE 8 BIT CHAR TO LEFT SIDE
 RAL 3            ;
 RAL 2            ;
 DAC WORD#        ; SAVE WORD
 LWC 1Ø           ; COUNT = -8 DEC
 DAC BITCNT#      ; BIT COUNTER
 IOT 34           ; WRITE
 LAC WORD         ; GET NEXT BIT INTO LINK
 RAL 1            ;
 DAC WORD         ; SAVE FOR NEXT BIT
 LSZ              ; SKIP IF LINK ZERO
 JMP .+1Ø         ; LINK =1 WRITE A 1 BIT
 LWC 2Ø7          ; LINK =Ø WRITE A Ø BIT
 DAC TEMP#        ;
 ISZ TEMP         ;
 JMP .-1          ;
 ISZ BITCNT       ; DONE ALL 8 BITS ?
 JMP .-13         ; NO
 JMP * SEND8      ; YES EXIT
 LWC 76           ; BIT =1
 DAC TEMP         ;
 ISZ TEMP         ;
 JMP .-1          ;
 IOT 34           ;
 JMP .-14         ;
```

Reading of the cassette tape is done by using IOT 014 to detect the next occurrence of a sync (clock) bit, and then using IOT 024 to check if the data bit is set (skip) or clear (no skip).

COMPREHENSIVE IOT LIST

(many of these are for special optional features or devices)

001 003	c(AC) → c(DPC)
001 004	Tablet clr status (TBI-1)
001 011	Clear TTY Break
001 012	Display off
001 014	Skip if one Bit Loader Clock #1 (CBS-1, REL-1)
001 021	Read KBD
001 022	Clear KBD Flag
001 024	Skip if one Bit Loader Data #1 (CBS-1, REL-1)
001 031	TTY Read
001 032	Clear TTY Input Status
001 034	One Bit Loader Write #1 (CBS-1)
001 041	TTY Xmit
001 042	Clear TTY Output Status
001 044	Tablet Skip (TBI-1)
001 051	Read PTR } (PTR-1)
001 052	Stop PTR }
001 054	Tablet Read X (TBI-1)
001 061	Start PTR (PTR-1)
001 062	Set TTY Break
001 064	Tablet Read Y (TBI-1)
001 071	Clear 40 cycle SYNC
001 072	IOT Sync
001 074	Tablet Read Z (TBI-1)

IOT LIST

(continued)

001 101	Read Level 1 Interrupt Status Bits to AC	
001 104	Start print (hard copy) (HCY-1)	
001 111	Set Memory Protect per AC	(PMP-1)
001 112	Clear Memory Protect Status	
001 114	Skip if hard copy busy (HCY-1)	
001 121	Load timer from AC	(ACI-1)
001 122	Clear timer status	
001 124	Skip if timer status =1	
001 131	Read Light Pen Buffer to AC	(LPA-1)
001 132	Clear Light Pen Status	
001 134	Skip if Light Pen Status = 1	
001 141	Arm or Disarm Level 1 Devices per AC Bits	
001 144	Start cassette #1 (CBS-1, REL-1)	
001 151	Read timer to AC (ACI-1)	
001 152	Half duplex turnaround (HDC-1)	
001 154	Stop Cassette #1 (CBS-1, REL-1)	
001 161	Disable Interrupt	Level 1
001 162	Enable Interrupt	
001 164	Load send/receive format (PSA-1)	
001 171	Load X Display Acc. Buffer	(XYR-1)
001 172	Load Y Display Acc. Buffer	
001 174	Load Plotter Register (PLO-1)	

IOT LIST

(continued)

001 201	Clear TTY-2 Break	(TKA-1)
001 202	Set TTY -2 Break	
001 204	Skip if TTY-2 Input Status Set	
001 211	Arm second level interrupt	
001 212	Read second level status	
001 214	Skip if TTY-2 Input Status Clear	(TKA-1)
001 221	Disable second level interrupt	(see note)
001 222 or 223	Enable second level interrupt	
001 224	Skip if TTY-2 Output Status Set	(TKA-1)
001 231	TTY-2 Read	(TKA-1)
001 232	Clear TTY-2 Input Status	
001 234	Skip if TTY-2 Output Status Clear	
001 241	TTY-2 Xmit	(TKA-1)
001 242	Clear TTY-2 Output Status	
001 244	Skip if "Clear to Send" is True	(HDC-1)
001 251	Read keyboard # 2	(KYB-1)
001 252	Clear keyboard # 2	
001 254	Skip on keyboard # 2	
001 261	Load send speed	(PSA-1)
001 262	Load receive speed	
001 271	Punch & Clear Flag	(PTP-1)
001 272	Start/Stop punch if AC_{15} is 1/0	
001 274	Skip if Punch Flag On	

IOT 272 required for BRPE punch only

NOTE: At the end of a second level interrupt servicing subroutine, the last instruction (prior to the jump out of the subroutine) should be IOT 221 (disable further second level interrupts) or IOT 223 (re-enable second level interrupts). Either of these IOT's will remove the lock-out condition which prevented 1st level interrupts from being detected during processing of the second level interrupt.

IOT LIST

(continued)

001 303	Read Mouse X & Y Coordinates and Initiate Next Conversion	(GMI-1)
001 304	Clear display halt status	
001 311	Read Joystick X Coordinate and Initiate Next Conversion	(JST-1)
001 312	Read Joystick Y Coordinate and Initiate Next Conversion	(JST-1)
001 314	Skip if one bit loader clock # 2	(CBS-2)
001 321	Control Audible Tone per AC Bit 15	(BEL-1)
001 322		
001 324	Skip if one bit loader data # 2	(CBS-2)
001 331	Read Mouse Switches & Keyset	(GMI-1)
001 332		
001 334	One bit loader write # 2	(CBS-2)
001 341		
001 342		
001 344	Start cassette # 2	(CBS-2)
001 351		
001 352		
001 354	Stop cassette # 2	(CBS-2)
001 361		
001 362		
001 364		
001 371		
001 372		
001 374		

IOT LIST

(continued)

001 401	Disc - Load Memory Address	(DSC-1)
001 402	Disc - Load Partial Read Control	
001 404	Disc - Skip if drive ready	
001 411	Disc - Seek cylinder	
001 412	Disc - Control command	
001 414	Disc - Skip if attention DRIVE 0	
001 423	Disc - Write sector command	
001 424	Disc - Skip if attention DRIVE 1	
001 433	Disc - Read sector command	
001 434	Disc - Skip if attention DRIVE 2	
001 441	Disc - Read status command	
001 442		
001 444	Disc - Skip if attention DRIVE 3	
001 451		
001 452		
001 454	Skip if high speed input has data (PTR-1 or GSI-1)	
001 461		
001 462		
001 464		
001 471	Print and clear line print flag	(PRT-1)
001 472		
001 474	Skip if line printer ready	

IOT LIST

(continued)

001 501	Tape - Load memory address	(MTC-1)
001 502	Tape - Load record size	
001 504	Tape - Skip if transport ready	
001 511	Tape - Read memory address	
001 512	Tape -	
001 514	Tape - Skip if tape system busy	
001 523	Tape - Transport command	
001 524	Tape - Skip if tape data busy	
001 533	Tape - Control command	
001 534	Tape - Skip if no check flag	
001 541		
001 542		
001 544		
001 551		
001 552		
001 554	Skip on Joystick, Mouse, or Trackball (JST-1, GMI-1, TBL-1)	
001 561		
001 562		
001 564		
001 571		
001 572		
001 574		

Functional Description

The Display Processor functions in one of several mutually-exclusive modes, including Processor Mode, Increment Mode, and Suppressed Grid.

In Processor Mode the Display Processor executes a set of instructions which are similar to but not as extensive as those of the Mini-computer. All processor mode display instructions require only 1.8 μsec execution time, as they are a class of instructions which only read from memory and do not affect memory contents.

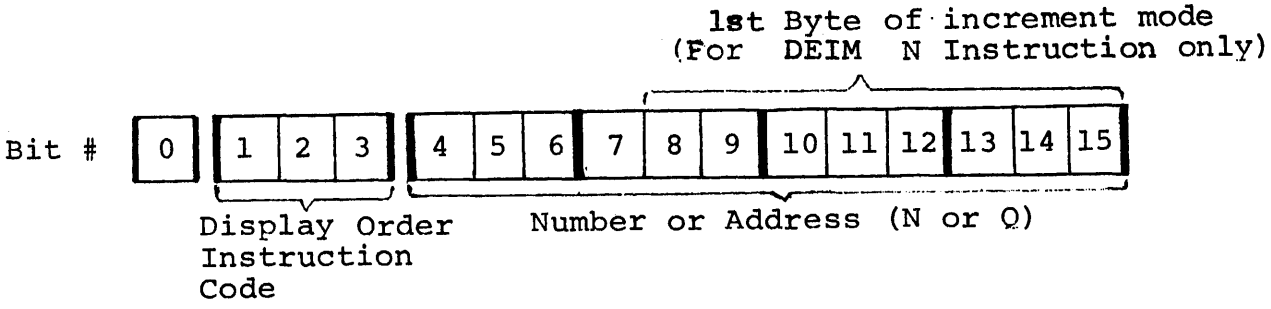
In Increment Mode, where most screen display is generated, the Display Processor decodes memory words into beam movements and intensifications. A drawing vector is coded in 8 bits, and there are two vectors specified by each 16 bit word which is referenced by the display while in Increment Mode. 1.8 μsec are required to execute each 8 bit byte of a 16 bit word; thus by simultaneously generating a display and continually referencing memory, 1.8 μsec are allowed for the Mini-computer per memory word executed in Increment Mode.

This system, however, necessitates the use of three 8 bit storage registers, two for the new memory word and one for the last 8 bit byte of the current word.

The magnitude and directions of the X and Y increments are decoded from the vector bytes and transferred to the X and Y accumulators and then to the D/A convertors which drive the X and Y Deflection Amplifiers.

Because this processor is dedicated to display generation the following instructions are designed to implement the functions of a display program.

Display Orders



DISPLAY ORDER WORD DECODING

(see drawing on page 58, which illustrates DLXA and DLYA)

Display Load X Accumulator with N (DLXA N)

Octal Code: 01nnnn

Cycles: Fetch

Operation: This instruction is used for absolute positioning of the beam along the X axis. The most significant bit and the least significant bit accumulators are specified by N, so any X position on the screen may be addressed to a resolution of 10 bits.

nnnn = N = any number from 0000 to 1777_8

N => C(XAC)

Display Load Y Accumulator (DLYA N)

Octal Code: 02nnnn

Cycles: Fetch

Operation: This instruction performs the operation described above but on the Y accumulator. Together these instructions will address any position on the screen.

nnnn = N = C(YAC)

Display Enter Increment Mode;
N is First Increment Byte (DEIM N)

Octal Code: 030 nnn

Cycles: Fetch

Operation: This operator, usually the first word of a graphic symbol or character subroutine, instructs the processor to decode the ensuing 8 bit bytes (2 per memory word) as vector movements whose sum define a symbol. The processor remains in increment mode, generating a display (during which time ordinary display instructions cannot be executed) until receiving a control byte which exits this mode. The DEIM instruction occupies less than 1/2 of a memory word, so the last 8 bits (N) may be used to code the first vector movement.

Display Jump to Subroutine (DJMS Q)

Octal Code: 05 nnnn

Cycles: Fetch

Operation: The present contents of the Display Program Counter (DPC) plus 1 become the contents of the DT register. The address Q becomes the content of the DPC. This instruction is analogous to a standard JMS instruction except for the use of the DT register rather than core memory to store a return address. Upon execution of a display return jump, the content of the DT register "jumps" back to the DPC. This is in effect a hardware duplicate of the function usually performed with an indirect Jump (JMP *). This feature is employed because the DRJM instruction requires only 1.8 μsec whereas the JMP * requires 3.6 μsec. Considering that a display list consists almost exclusively of DJMS commands, the time thus saved is significant for a refreshing display operating at 40 frames per second. (See comments on page 56 on the MDS-1 option, which makes an 8-level DT stack available.)

Display Jump to Address Q (DJMP Q)

Octal Code: 06 nnnn

Cycles: Fetch

Operation: This instruction, like the JMP instruction associated with the mini-computer, transfers program control to the instruction at address Q. For example, it is useful for putting patches into display lists when data is being input to the machine too fast to be inserted in the lists immediately, or when links are needed between graphic subroutines.

Q => C(DPC)

LONG VECTOR INSTRUCTION FORMAT

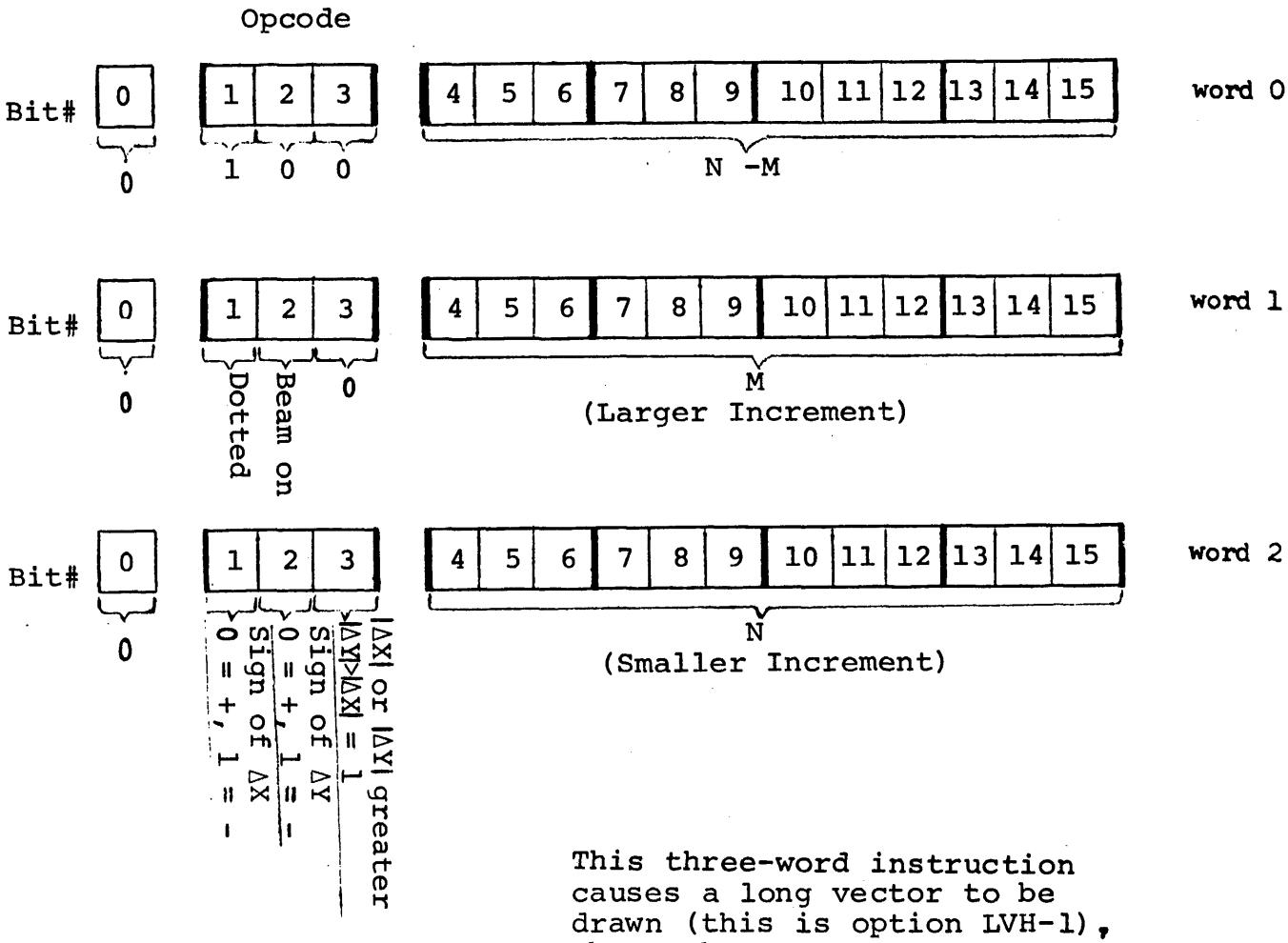
This is a special three-word instruction, which begins with the word: 04nnnn.

Note: M = the greater absolute value;

$|\Delta X|$ or $|\Delta Y|$ $\qquad M \neq 0$

N = the smaller absolute value;

$|\Delta X|$ or $|\Delta Y|$

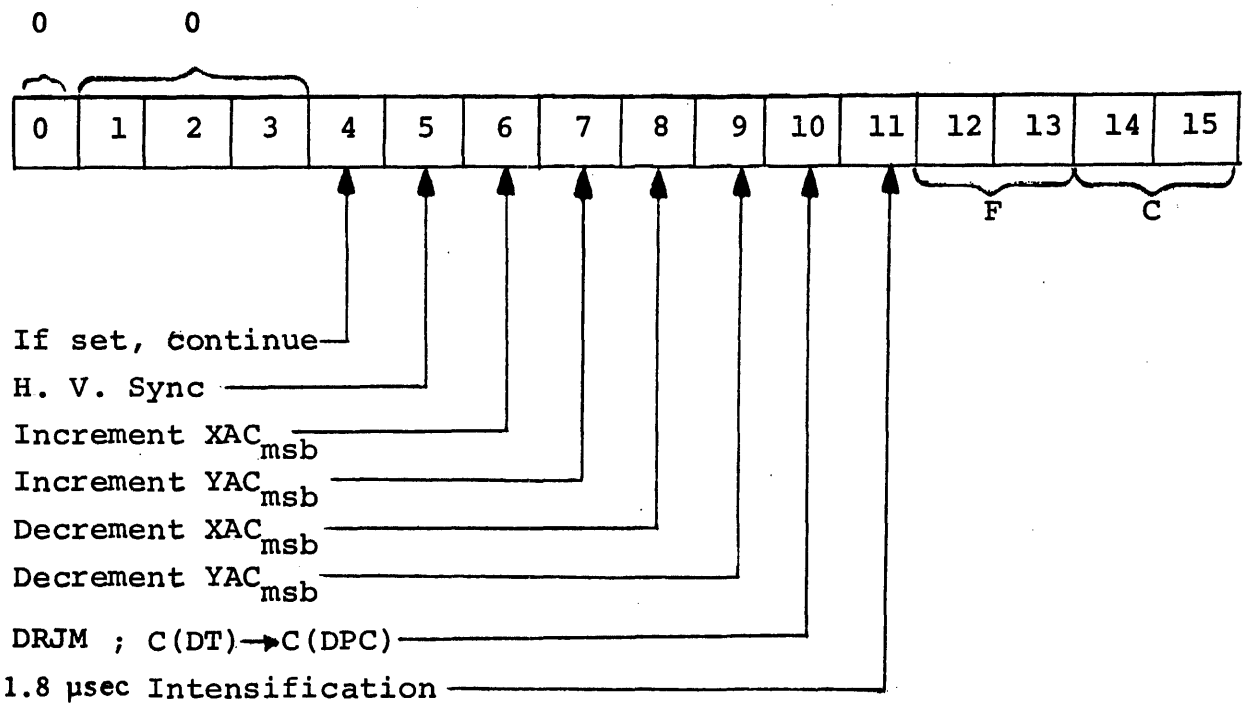


This three-word instruction causes a long vector to be drawn (this is option LVH-1), where the parameters are as given. No escape byte is necessary, as sequential processing resumes automatically.

See the listing of Assembler pseudo-ops for explanation of the DLVH opcode for specifying Long Vector.

DISPLAY OPERATE MICRO INSTRUCTIONS

These operates are analogous to those of the Mini-computer in that they manipulate or control the display registers, but do not affect memory contents. As with the Display Orders, the display must be in processor mode to interpret the following commands.



If F = 00: C is not decoded; If F = 11: Display Arm Light Pen (sensitize)

If F = 01: Set Scale per C

- C = 00: Scale 1/2
- C = 01: Scale 1
- C = 10: Scale 2
- C = 11: Scale 3

If F = 10: Set Memory Block per C

- C = 00: Set Block 0
- C = 01: Set Block 1
- C = 10: Set Block 2
- C = 11: Set Block 3

Display Halt (DHLT)

Octal Code: 000000

Cycles: Fetch

Operation: The display processor is halted.

Display No Operate (DNOP)

Octal Code: 004000

Cycles: Fetch

Operation: The state of the machine is preserved for 1.8 µsec, i.e., no register contents change. This instruction may be used for time consumption and/or synchronization.

Display Set Scale (DSTS 0, 1, 2, 3)

Octal Code: 004004, 05, 06, 07

Cycles: Fetch

Operation: This operator sets the drawing scale; i.e., it changes displayed character size without altering character description. Because a drawing vector's magnitude (length) is represented by the digital number in the X and Y accumulators, that length may be proportionately altered by simply scaling the digital quantity in the accumulators.

Specifically, a vector's basic length (1/2, 1, 2, or 3 least significant bit units (lsb)) is decoded from an 8 bit byte by control logic, which increments the XAC_{lsb} and YAC_{lsb}. Scaling is thus achieved by selectively inputting these pulses to bits 8 and/or 9, or bit 10 of the XAC_{lsb} and YAC_{lsb}. For example, to draw a 3 unit vector in scale 2, 3 serial pulses would be switched into AC bit 8 ; to draw a 2 unit vector in scale 3, 2 serial pulses would be switched into bits 8 and 9. (See Mechanics of Drawing on page 59.)

Display Set Block 0 (DSTB 0)

Display Set Block 1 (DSTB 1)

Octal Code: DSTB 0 - 004010
DSTB 1 - 004011

(as noted on previous page, the DSTB instruction can now be coded for 12K or 16K, using DSTB 2 or DSTB 3)

Cycles: Fetch

Operation: This instruction is available only on machines with 8K or more of core. It is used to alter the information bit(s) which specifies the particular 4K block of memory to be referenced, and precedes DJMP and DJMS calls to display subroutines when these subroutines and their calls are not in the same 4K block of memory. This instruction does not directly affect the Display Program Counter.

Display 1.8 µsec Intensification (DDSP)

Octal Code: 004020

Cycles: Fetch

Operation: The CRT beam is turned on for 1.8 µsec at the current X, Y location. This instruction may be combined with DIXM, DIYM, DDXM, or DDYM, and used to draw a line on the screen without entering increment mode.

Display Increment X Accumulator Most Significant Bits (DIXM)

Octal Code: 005000

Cycles: Fetch

Operation: Add a binary 1 to bit 5 of the X_{msb} accumulator. This is equivalent to incrementing bit 9 sixteen times.

Display Increment Y Accumulator Most Significant Bits (DIYM)

Octal Code: 004400

Cycles: Fetch

Operation: Analagous to DIXM.

Display Return Jump (DRJM)

Octal Code: 004040

Cycles: Fetch

Operation: This command is used to return jump from a DJMS subroutine back to the calling routine. Its execution causes the address previously deposited in the DT register to become the contents of the display program counter. (See page 56, on MDS-1 option.)

Display Decrement X Accumulator Most Significant Bits (DDXM)

Octal Code: 004200

Cycles: Fetch

Operation: Subtract a binary one from the most significant bits (bit 5). This has the effect of moving the beam to the left by one most significant bit.

Display Decrement Y Accumulator, MSB (DDYM)

Octal Code: 004100

Cycles: Fetch

Operation: Analagous to DDXM.

Display H.V. Sync (DHVC) (this instruction is required only on early PDS-1 models)

Octal Code: 006000

Cycles: Fetch

Operation: This instruction excites the D.C. high voltage supply oscillator for the CRT and synchronizes it with the rest of the system at the chosen refresh rate. It should be executed once per frame, either at the beginning or end of the display routine.

Sensitize Light Pen (DOPR 15)
Desensitize Light Pen (DOPR 14) } (LPA-1 option)

Octal Code: 004015, 004014

Cycles: Fetch

Operation: All visible information displayed on the screen after a DOPR 15 has been performed will be light-pen sensitive. Any visible information displayed on the screen after a DOPR 14 has been performed will not be sensitive to the light pen.

PDS-1 DISPLAY ADDRESSING (8K)

There are only 2 display instructions which address memory locations:

```
05 0000          DJMS   E
06 0000          DJMP   E
```

Each of these instructions has a 12 bit address field E, which allows it to address 4K of memory directly.

For a PDS-1 with 4K of core, the effective address, E, of a DJMS or a DJMP is determined entirely by the 12 bit address field of the instruction.

For an 8K PDS-1 the contents of an information bit becomes the contents of bit 3 of E; bits 4-12 of E are still determined from the address field of the instruction.

The information bit (s) is only found on machines with more than 4K of core. There are only two ways in which the information bit can be altered (or initialized). The first way is from the PDS-1 mini-computer accumulator on a DLA 001003 / C(AC) ⇒ C(DPC); bit 3 of the accumulator also goes to the information bit.

The second way of setting or altering the information bit is from the PDS-1 display processor on a set block instruction:

```
004010          DSTB  0     / 0 ⇒ C(info. bit)
004011          DSTB  1    /  1 ⇒ C(info. bit)
```

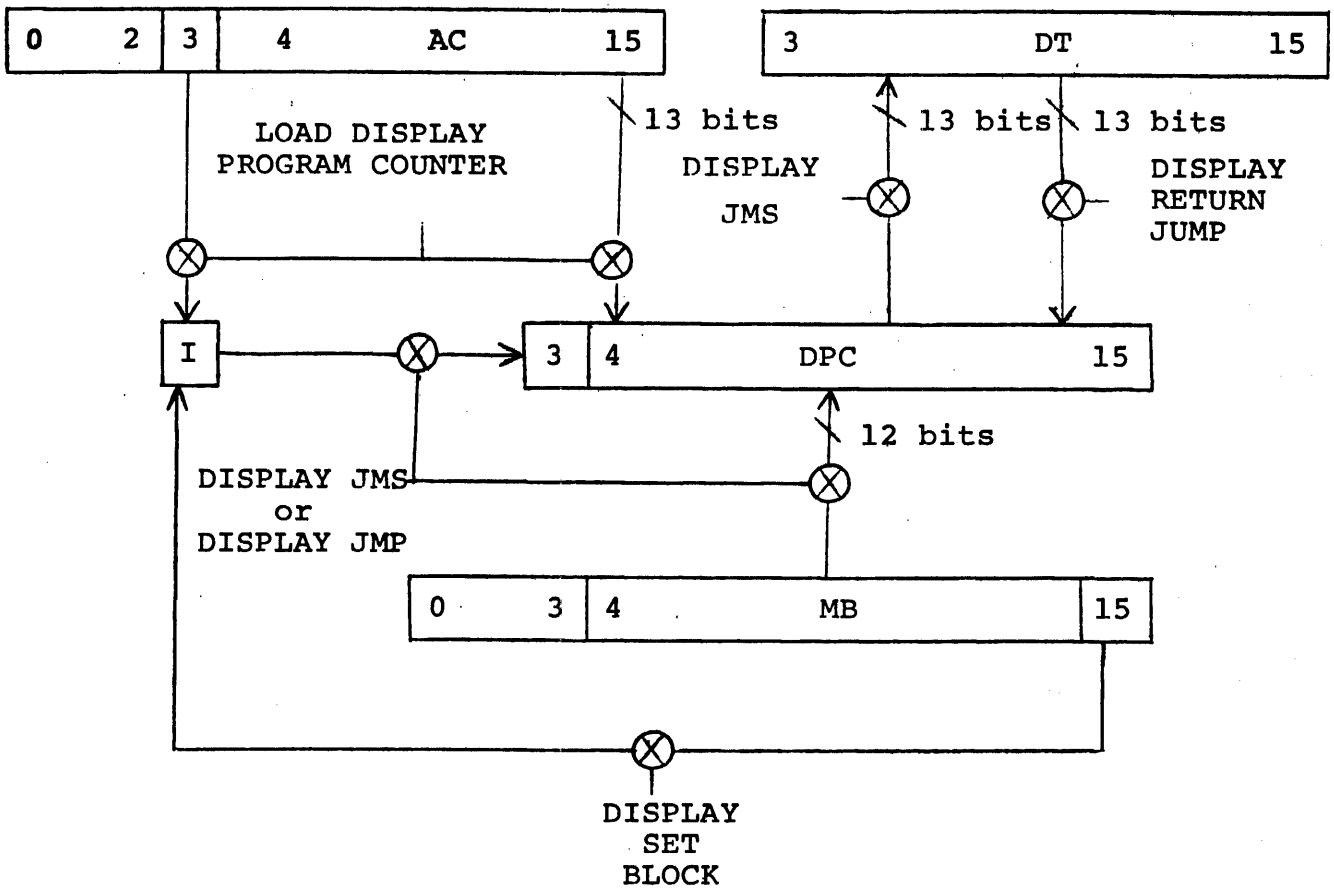
On an 8K PDS-1, if the display list and all the character subroutines start in the same 4K block of core, the information bit need only be set once per frame - from the list processor accumulator when the display program counter is initialized. From then on, all DJMS's or DJMP's refer to the same 4K block of core.

However, if the program is organized such that some character subroutines are not in the same 4K block as the rest, then calls to these subroutines have to be preceded by the appropriate display set block instruction.

Bit 3 of the DPC is automatically complemented when the DPC counts across a 4K boundary such as $7777_8/10000_8$.

(As noted previously, DSTB 2 and DSTB 3 can also be used.)

SCHEMATIC DESCRIPTION OF
DISPLAY ADDRESSING ON 8K PDS-1



Display Increment Mode Instructions

After the EIM instruction is executed, display instructions will be interpreted in one of two formats:

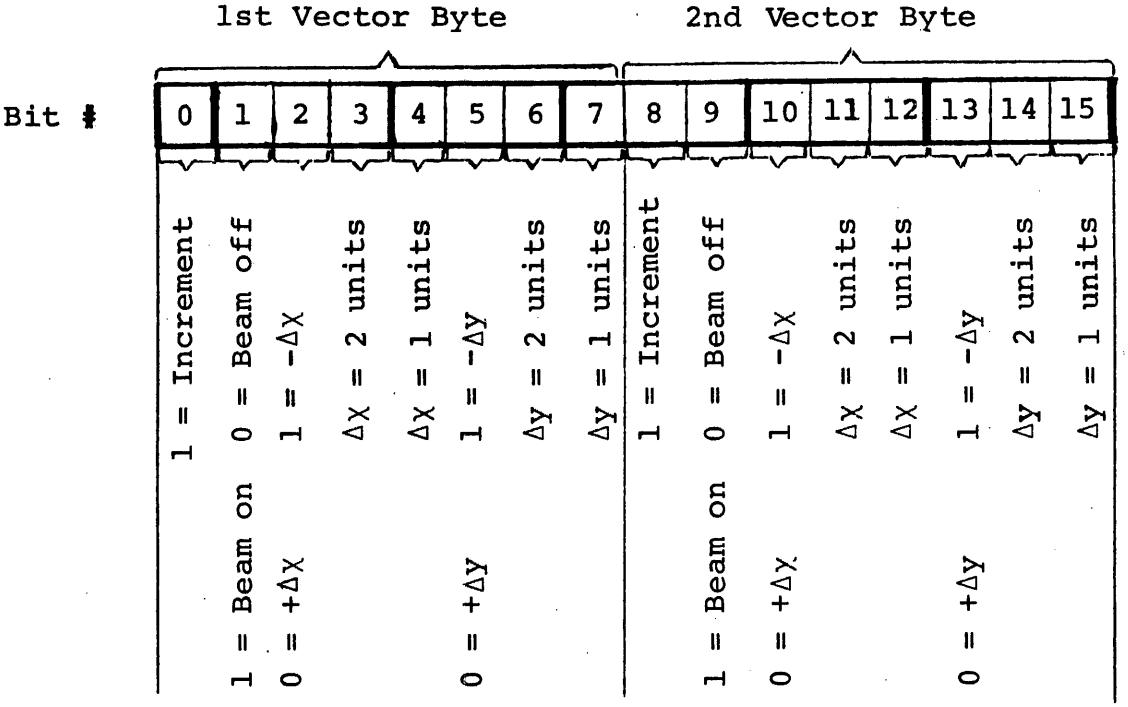
1. If bit 0 is set, the following 8 bits will be decoded as a vector (increment) byte.
2. If bit 0 is not set, the following 8 bits will be decoded as an escape byte which, when executed, returns the display processor to processor mode.

A vector byte specifies whether or not the beam is turned on, the direction (+ or -) of the X and Y components, and the magnitude (1 to 3 least significant bit units) of the X and Y components.

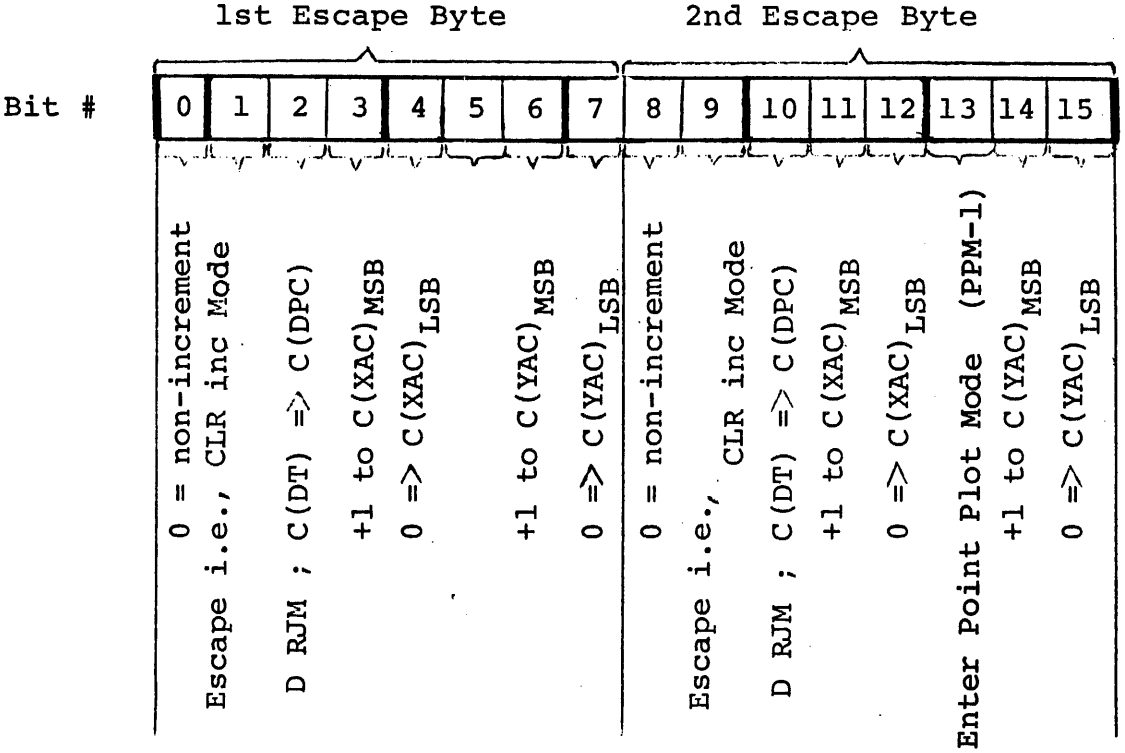
The escape byte may be coded in either the first or last 8 bits of a memory word, as shown on the next page, and hence may be combined with the last vector byte to form the final word of a subroutine, should the total number of vector bytes be even.

However, if the total is odd, the escape byte may be specified as the first 8 bits of the final memory word. Upon execution of this byte, the machine immediately enters Processor Mode, without wasting a memory cycle attempting to decode the second byte of the memory word.

By not setting the display break (bits 1 and 9), the control byte may also be used to clear LSB's and increment MSB's without leaving increment mode.



Increment Byte Decoding



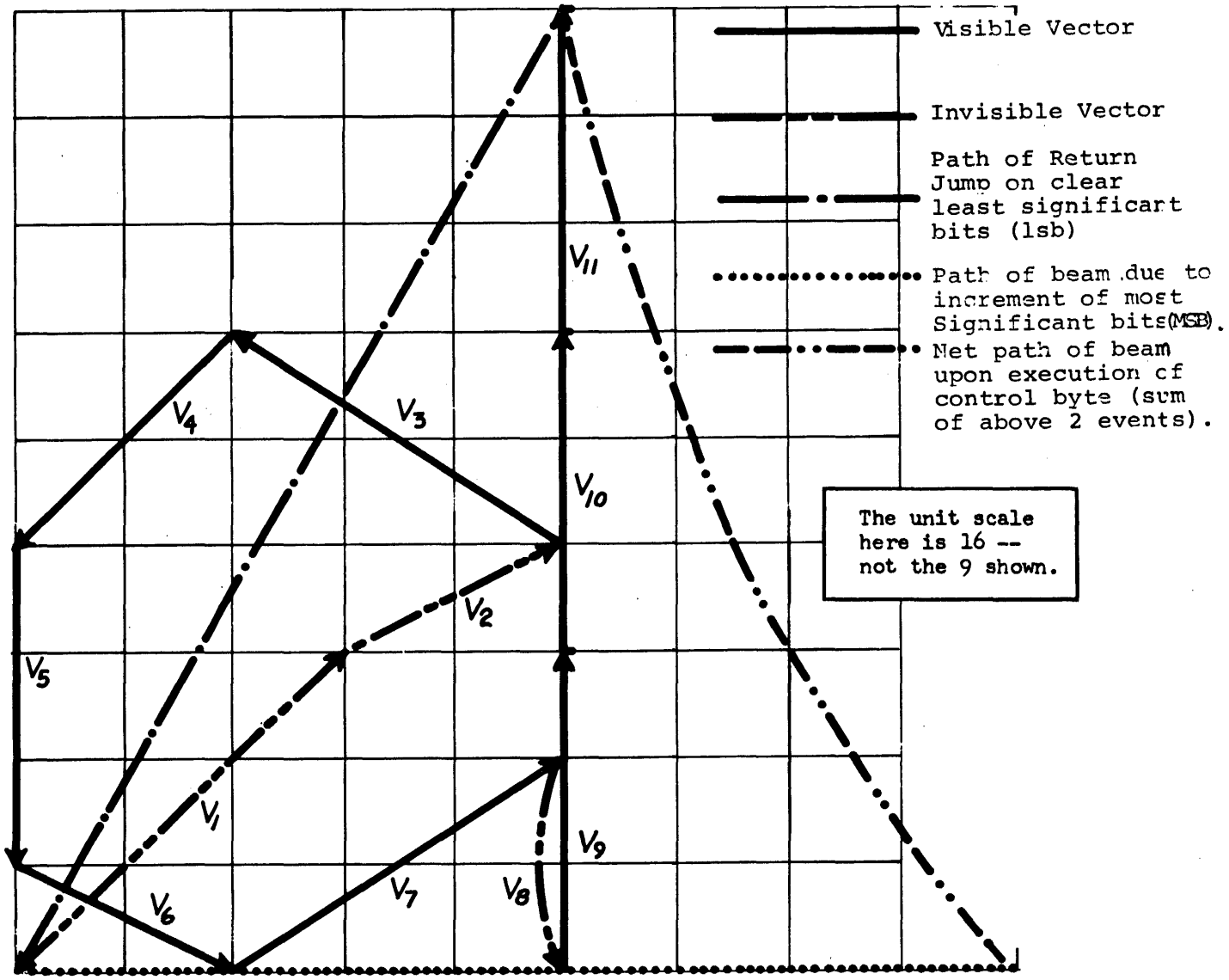
Control Byte Decoding

A return from display subroutine should be accompanied by an exit increment mode. An Enter Point Plot increment mode must be accompanied by an exit regular increment mode and the control byte must be in the right half word.

Vector Description of Letter "d"

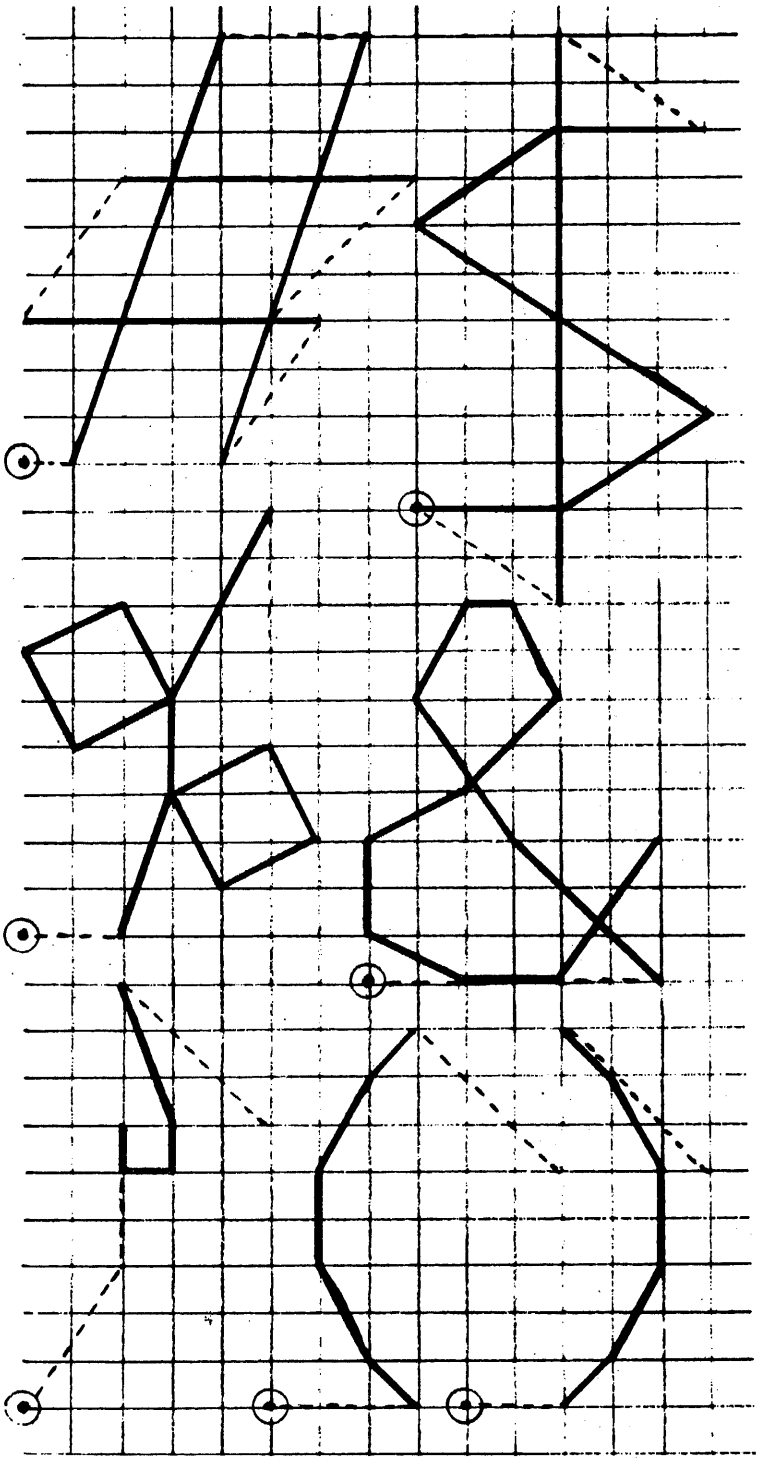
Memory Location	Octal Representation	Vector Description			
5440	030233	E	D+3+3	; EIM	V 1
5441	110772	D+2+1	B-3+2	; V 2	V 3
5442	173307	B-2-2	B+0-3	; V 4	V 5
5443	152732	B+2-1	B+3+2	; V 6	V 7
5444	103303	D+0-2	B+0+3	; V 8	V 9
5445	141703	B+0+3	E+0+3	; V 10	V 11
5446	074571	F	F	; ESCAPE	+ RETURN JUMP

E=Enter Increment Mode (sets flag in machine)
D=Dim (beam off)
B=Beam on
F=Escape



Here is an example of some increment mode character descriptions (for number sign, dollar sign, percent sign, ampersand, apostrophe, and left and right parentheses), where each origin is indicated here by ⊙ . For each increment byte, the maximum displacement that may be specified is + or - 3 in the X (horizontal) and/or Y (vertical) directions.

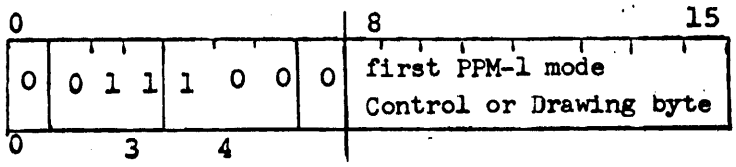
```
030210 DISNO: INC   E D+1+0   ; #
145713   INC   B+1+3 B+1+3
145630   INC   B+1+3 D+3+0
167757   INC   B-1-3 B-1-3
167623   INC   B-1-3 D+2+3
174370   INC   B-3+0 B-3+0
111730   INC   D+2+3 B+3+0
154277   INC   B+3+0 D-3-3
074400   INC   F A000

030236 DISDL: INC   E D+3-2   ; $
141703   INC   B+0+3 B+0+3
141703   INC   B+0+3 B+0+3
117370   INC   D+3-2 B-3+0
177336   INC   B-3-2 B+3-2
157376   INC   B+3-2 B-3-2
174171   INC   B-3+0 F

030220 DISPC: INC   E D+2+0   ; %
145721   INC   B+1+3 B+2+1
147365   INC   B+1-2 B-2-1
165302   INC   B-1+2 B+0+2
165365   INC   B-1+2 B-2-1
147321   INC   B+1-2 B+2+1
145312   INC   B+1+2 B+1+2
103571   INC   D+0-3 F

030230 DISAM: INC   E D+3+0   ; &
114373   INC   D+3+0 B-3+3
171712   INC   B-2+3 B+1+2
144316   INC   B+1+0 B+1-2
173365   INC   B-2-2 B-2-1
143325   INC   B+0-2 B+2-1
150323   INC   B+2+0 B+2+3
074400   INC   F A000

030223 DISAP: INC   E D+2+3   ; '
101705   INC   D+0+3 B+0-1
144301   INC   B+1+0 B+0+1
165637   INC   B-1+3 D+3-3
074400   INC   F A000

030230 DISLP: INC   E D+3+0   ; (
164752   INC   B-1+1 B-1+2
141312   INC   B+0+2 B+1+2
144637   INC   B+1+1 D+3-3
074400   INC   F A000

030220 DISRP: INC   E D+2+0   ; )
144712   INC   B+1+1 B+1+2
141352   INC   B+0+2 B-1+2
164637   INC   B-1+1 D+3-3
074400   INC   F A000
```



POINT PLOT MODE HARDWARE, w/16 LEVELS OF Z-AXIS INTENSIFICATION (PPM-1)

PPM is available for applications which require up to 16 programmable levels of beam intensity or an efficient method for plotting point matrices.

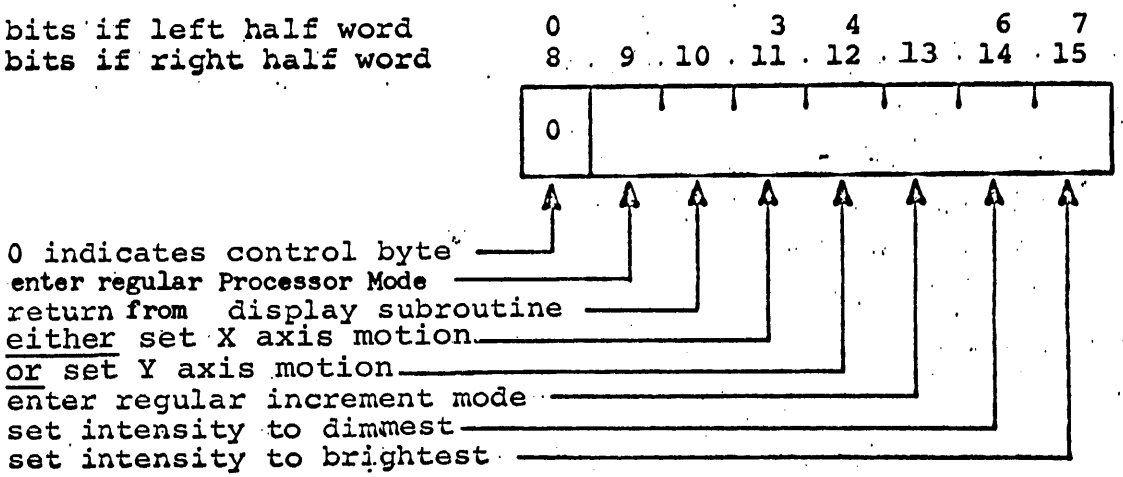
PPM may be entered from Increment Mode by setting bit **thirteen** of an Increment Mode Control Byte, (see page 48) or directly from Processor Mode via the execution of 034nnn:



This is a main display instruction and may not be executed in either increment mode. This instruction is different from the enter regular increment mode by only bit 4 (this is 034nnn, as opposed to 030nnn for the regular D EIM).

In order to change coordinate directions, a PPM-1 Control Byte is used:

PPM-1 CONTROL BYTE

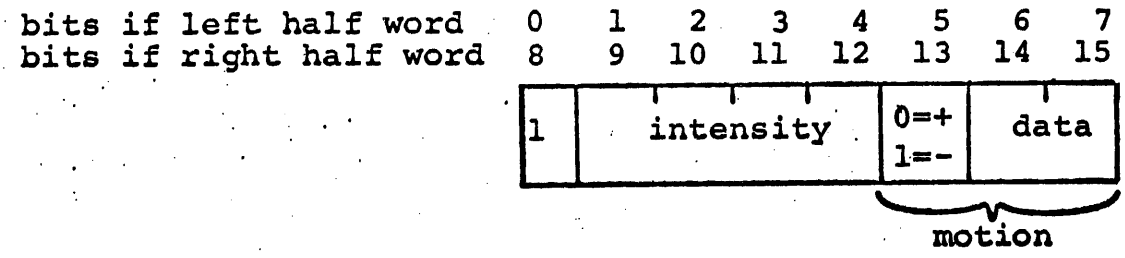


A 1 in the indicated bit position causes the specified action. A return from display subroutine should be accompanied by an enter Processor Mode or enter Increment Mode. On enter regular increment mode, the control byte must be in the right half word.

By setting the bits as indicated, it is also possible to Return Jump, change to Processor Mode, change to Increment Mode, and to set Maximum or Minimum Intensity.

Each point is specified by an 8 bit **Drawing Byte, where bits 1-4 (or bits 9-12)** allow the specification of a different intensity for each point plotted. In addition, it is possible to move the beam 0, 1, 2, or 3 vector positions before intensification; hence, a matrix of points may be drawn in this mode. These movements are restricted to the X or Y direction.

PPM-1 DRAWING BYTE



A 1 in bit 0 or 8 indicates a drawing byte
bits 1-4 or 9-12 specify the new intensity
bits 5-7 or 13-15 specify the motion

a) the axis of motion must be set by a PPM-1 control byte (the X axis is set when the display is turned on at the beginning of each frame)
b) bit 5 or 13 specifies the direction of motion
c) bits 6-7 or 14-15 specify the magnitude of motion

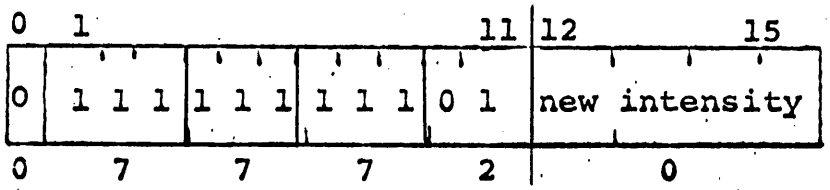
The action of a drawing byte can be described as "move - plot". The motion occurs first (during time pulses 2, 3, and 4) while the beam is blanked. Then (during time pulses to 6, 7, 8, 9, 10, 1) a point is displayed at the specified intensity. The motion is affected by scaling like regular increment mode.

When DON is used, axis = X,
brightness = maximum,
PPM mode is off,
video is off,

ADDITIONAL (OPTIONAL) DISPLAY PROCESSOR INSTRUCTIONS

07766n
07767n — see description below on Automatic Increment and Intensify Feature for SGR-1 (this is the ASG-1 option)

07772n
07773n — Variable Intensity Control (VIC-1 option):



This is a display operate instruction and may not be executed in increment mode. When the display is turned on by the CPU at the beginning of each frame, maximum intensity is set.

07774n
07775n — Monitor Control Interface (MCI-1 option):

The same Display Processor is used to drive all CRT's. Therefore, all monitors receive identical deflection signals. The beam in each monitor, however, may be switched on or off via display operate instruction 07774n or 07775n. In this manner, the programmer is able to select the subset of information to be written on each CRT.

If bit 12 is set, Monitor #4 (for example, STO-1 option) is on.
If bit 13 is set, Monitor #3 (for example, HCY-1 option) is on.
If bit 14 is set, Monitor #2 (for example, SLM-1 option) is on.
If bit 15 is set, Monitor #1 (this is the normal monitor) is on.

07776n — Storage Tube Interface (STI-1 option); or Light Pen (LPA-1 option):

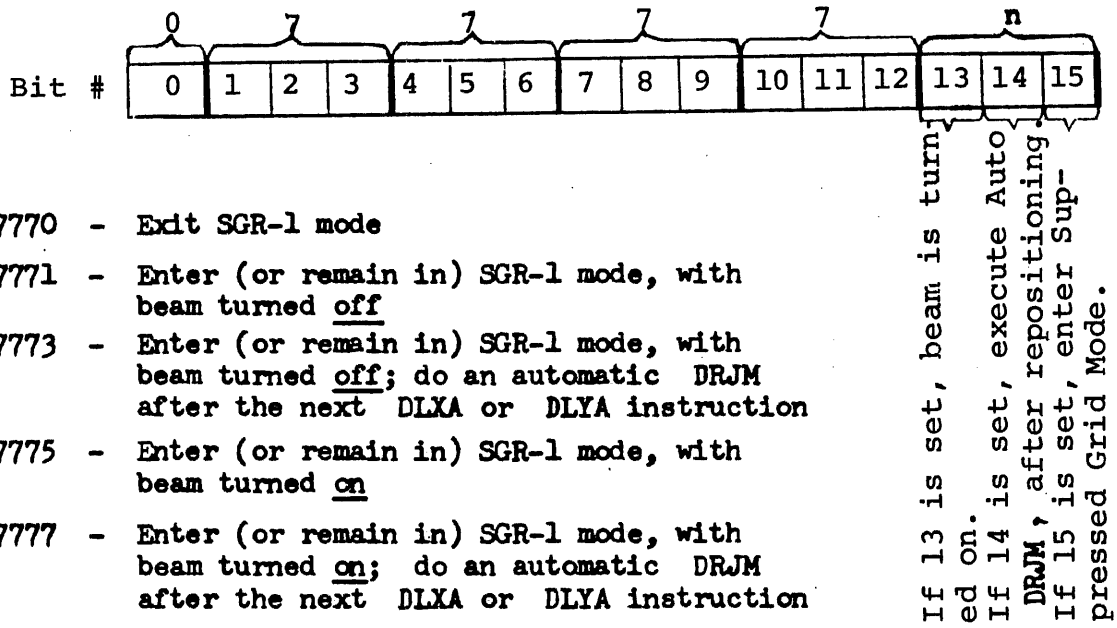
077760	Clear non-store (refresh) and write-through modes	storage tube
077761	Erase	
077762	Set store mode	
077763	Set write-through mode	
077764	Clear view mode	
077765	Set view mode	
077766	Clear light pen enable bit	(see page 56)
077767	Set Light pen enable bit	

07777n — Fast X or Y Vector Generator (SGR-1 option): see next page

SGR-1 INSTRUCTION DESCRIPTION

The Suppressed Grid feature allows the programmer to draw grid lines at a reduced intensity or to position the beam very rapidly without incorporating the usual timing routines to allow for beam settling. The Suppressed Grid instruction is 077 77n.

Decoded as follows:



077770 - Exit SGR-1 mode

077771 - Enter (or remain in) SGR-1 mode, with beam turned off

077773 - Enter (or remain in) SGR-1 mode, with beam turned off; do an automatic DRJM after the next DLXA or DLYA instruction

077775 - Enter (or remain in) SGR-1 mode, with beam turned on

077777 - Enter (or remain in) SGR-1 mode, with beam turned on; do an automatic DRJM after the next DLXA or DLYA instruction

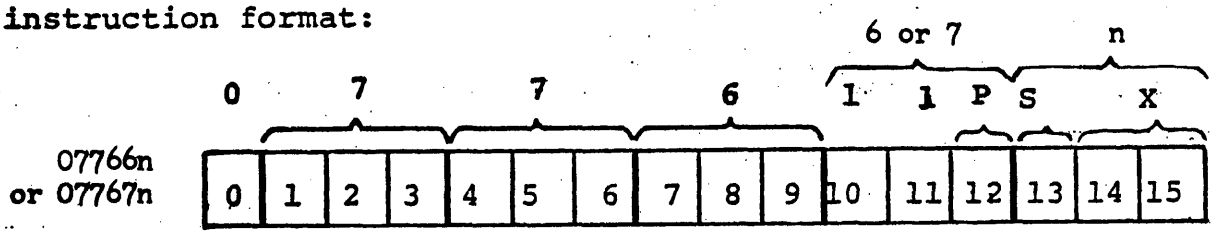
Suppressed Grid Mode is entered by executing instruction 077 77n with bit 15 set to a 1. After entering the mode, repositioning of the beam in X or in Y is accomplished via a DLXA or DLYA instruction. Suppressed Grid Mode may be exited by executing instruction 077 770.

If bit 14 is set before a repositioning takes place, an automatic Return Jump (DRJM) to the main program may be executed after the repositioning has occurred.

No display breaks will be taken during a repositioning and, hence, no increment drawing can occur at this time.

AUTOMATIC INCREMENT AND INTENSIFY FEATURE FOR SGR-1

The Automatic Increment and Intensify Feature (ASG-1), in combination with the suppressed grid (SGR-1), allows the rapid display of data from very compact data lists, especially those data representing time sequential samples. Automatic incrementing of the X axis and optional endpoint intensification are made available and determined by the following instruction format:



P=1 - intensify point at end of drawing stroke

S and X - Sign and Magnitude of ΔX movement (S=1 for minus)

The instruction 077660 (P and S and X all zero) turns off ASG-1

The optional endpoint intensification is independent of whether the line being drawn was intensified or not. The X increments are subject to whatever scale function has been set. Increments are executed in the first few microseconds of beam movement.

The ASG-1 feature is operative only while in SGR-1 mode, and then only when a Load Y command is interpreted. Thus, if the data to be displayed is a series of data points representing samples taken of a time varying wave form at uniform time intervals, the sample values may be stored in the address portions of a list of DLYA instructions (scaling and offset may be applied to the data values first). Then a subprogram of the following type will display the data (say 250 points) as a series of low intensity lines with intensified points at the actual data values:

SAMPLE ASG-1 SUBPROGRAM

```
        .
        .
        .                    } Previous display instructions, NOT in
        .                      suppressed grid mode
        .
        .
        .

SGR-1 mode
  077771          ; Enter Suppressed grid mode, beam off
   DLXA X_o       ; Set initial X axis address
   DLYA Y_o       ; Set initial Y axis address
  077775          ; Remain in suppressed grid mode, with beam on

  ASG-1 feature
  077673          ; Activate ASG-1 feature, with end point intensify, S and X = +3
   DSTS 1         ; Set scale =1 so that each auto. increment will
                  ; equal 3 raster units.
   DJMS DATA
  077660          ; Turn off ASG-1 feature
  077770          ; Exit  suppressed grid mode

        .
        .
        .                    } More display instructions
        .
        .
        .

DATA:  DLYA   Data_1
       DLYA   Data_2
        .
        .                    } 250 sequential data value
        .
        .
        .
       DLYA   Data_250
       DRJM             ; Return to the location following  DJMS DATA instruction
```

THEORY BEHIND MDS-1 OPTION

(Multiply-nested Display Subroutine Feature)

Unlike the PDS-1 mini-processor, the PDS-1 display computer does not write into core memory.

When executing a Jump to Subroutine (JMS) instruction, the mini-processor saves the return address at the location referenced by the operand of the JMS instruction, and jumps to the address following that location. To return from the subroutine, the program normally does an Indirect Jump (I JMP) off of the location which holds the return address.

Since the display computer cannot save the return address on a Display Subroutine Jump (D JMS) instruction by writing it into core memory at the location referenced in the D JMS instruction operand, the display computer hardware has to hold the return address in a special internal register. (This is the DT, Display Temporary, register.)

Therefore, on a D JMS instruction, control is transferred directly to the location referenced in the operand. When the display subroutine does a LRJM, the hardware automatically retrieves the saved return address and transfers it to the Display Program Counter, effectively transferring control back to the calling routine.

The MDS-1 Option provides 8 different return address save registers for use by the display computer. Thus, one display routine may D JMS to another subroutine, which inturn may D JMS to another subroutine, and so on, where subroutines may be nested up to eight levels deep.

On completion of each nested display subroutine, control returns following the D JMS instruction within the next higher subroutine that called it.

The 8-level DT register stack operates on the "push-pop" principle, where the last address saved is the first retrieved (LIFO -- Last In, First Out).

Note about use of Light Pen (LPA-1 option) with nested display subroutines:

As part of the MDS-1 and LPA-1 options, when both of these options are included, two additional display processor instructions (077766 and 077767) are provided to control light pen interrupt occurrence at specific display subroutine levels. This makes it unnecessary to disable interrupts to prevent a light pen interrupt from occurring at some level within nested subroutines where it is not desired.

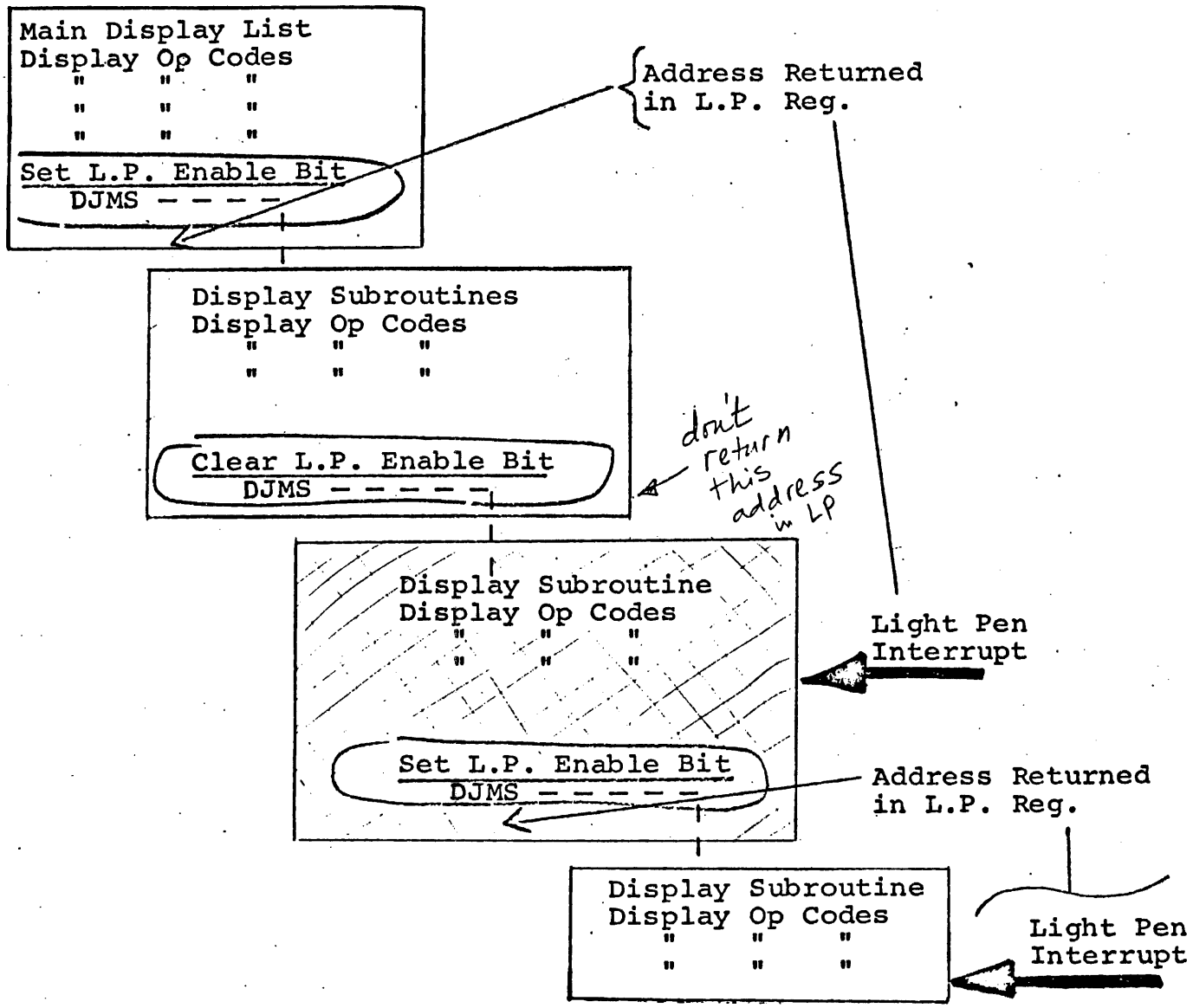
One of these two instructions is used prior to each DJMS jump-to-subroutine instruction (as noted in the illustration below), to indicate whether or not

an interrupt should be allowed to occur at the time of return to the routine which called the subroutine, if the presence of the interrupt was detected by the hardware while processing was going on within that subroutine.

In the DT register stack, there is an extra bit in each register that can be either set or cleared at the time that the return address is being saved. 077767 (Set Light Pen Enable Bit) causes the bit to be set. 077766 (Clear Light Pen Enable Bit) causes it to be cleared.

When the light pen is activated and receives a "hit", the display processor will test each word as it is popped (removed from the DT stack). When an address word is encountered with the light pen bit set, that address will be stored in the Light Pen Address register and an interrupt will occur.

If an address word is encountered with the light pen bit clear, the interrupt will be held off, and subsequent checks will be made of the light pen bit as subsequent address words are popped from the stack.



GRAPHIC MACHINE CHARACTERISTICS

The PDS-1D and the COMPOSER-15 are known as Graphic machines, which means that there is an automatic carry between the Least Significant Bits (LSB) and the Most Significant Bits (MSB) of the X and Y Display Accumulators. Most later model PDS-1 systems are also Graphic machines. All earlier model PDS-1 systems are what is known as Alpha machines, where there was NO CARRY between the LSB and the MSB.

The following information summarizes the characteristics of a Graphic machine, which is somewhat easier to program than an Alpha machine, since the LSB and MSB portions of each Display Accumulator can be thought of as a single continuous entity.

CONTINUOUS D/A SYSTEM

The 11 bit, single D/A system has 6 MSB's and 5 LSB's. The additional LSB (bit 10, see drawings) is used only for scale 1/2 drawing and is not affected by a direct load instruction or normal scale 1 character drawing. There is arithmetic carry between the two bit groups and, hence, in scale 1 this system acts as a continuous 10 bit register. The 10 bits in X and Y yield a resolution of 1024 points by 1024 points within an 8" x 8" display area. (This point density is adjustable, i.e., it is possible to increase or decrease the display area.)

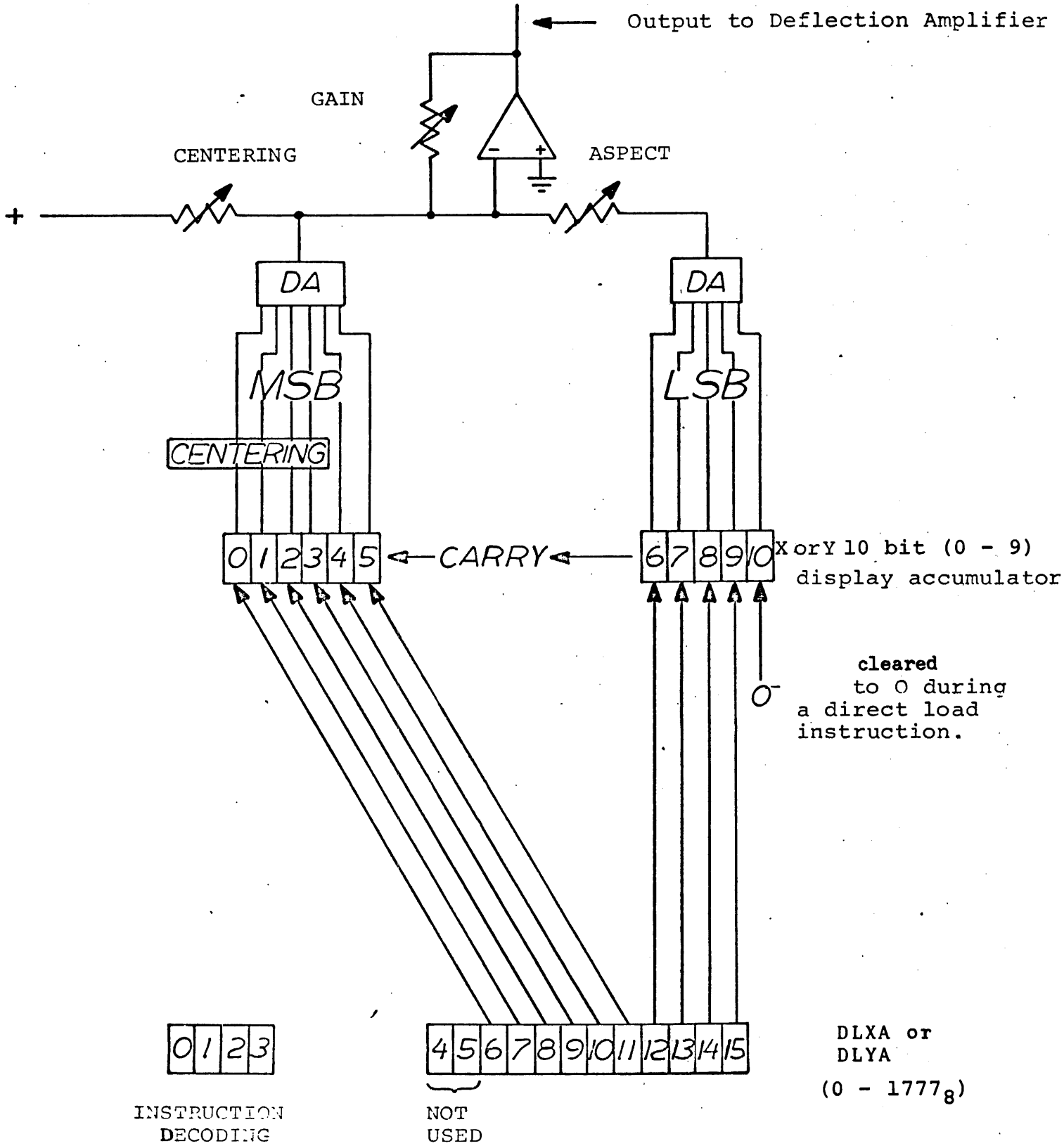
The principal advantage gained with the continuous system is the ease of graphics programming. It is no longer necessary to keep track of "box boundaries" when drawing lines, because all carry functions are automatic. Also, character symbols are not constrained to occupy a box of fixed dimensions but may be of variable width and height.

By allowing overlap, it becomes possible to draw (move the beam) outside of the "box boundaries" specified by the MSB positioning bits. This feature can be useful when displaying a character font whose symbols are not of a uniform width /height ratio.

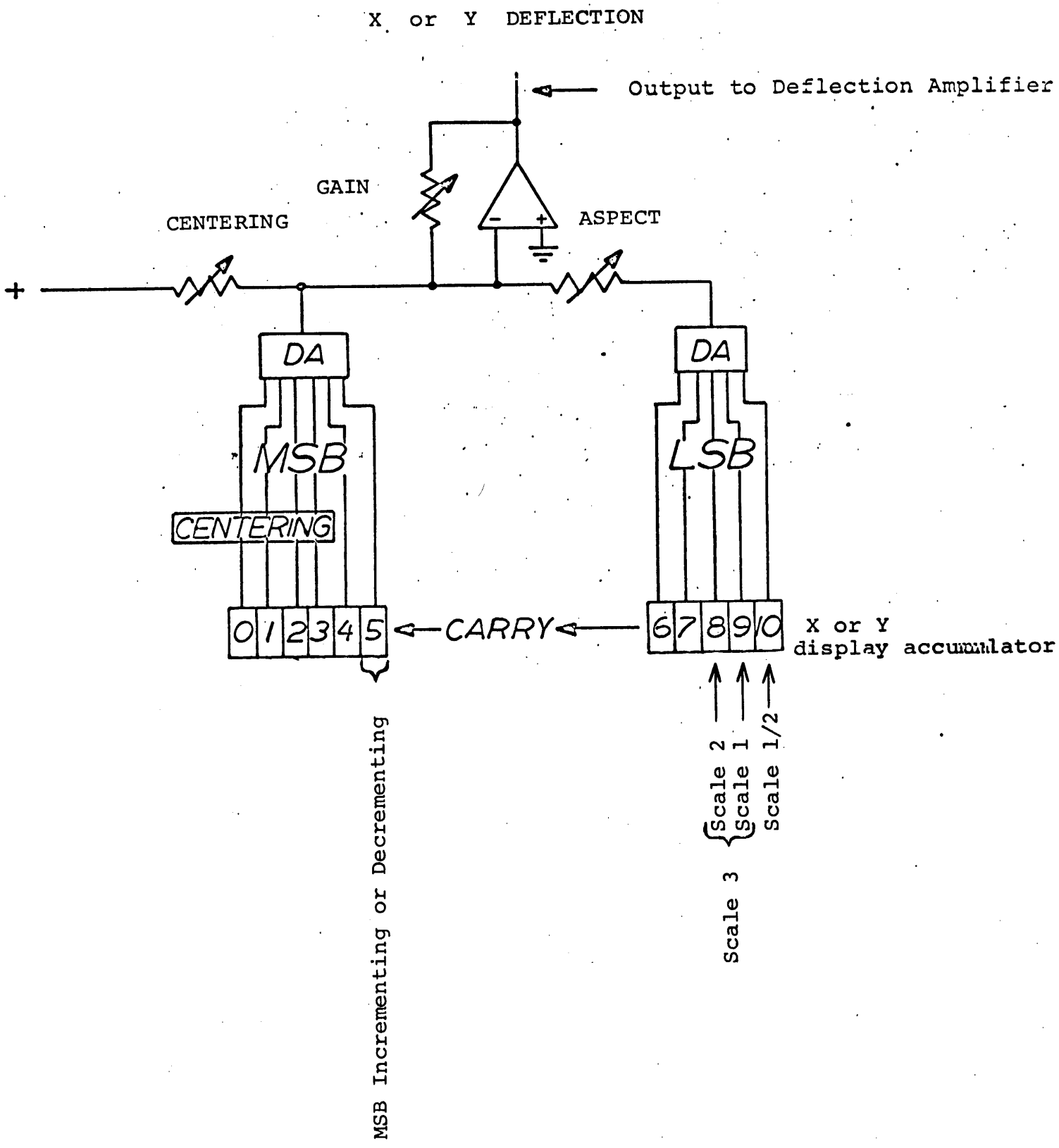
In scale 1, 16 LSB's are equivalent to 1 MSB.
In scale 2, 8 LSB's are equivalent to 1 MSB.

The graphic displays allow ten bits for addressing an X or Y screen position (0-1777_8), with 1000_8, 1000_8 being the center of the screen. All display commands that affect the MSB's or LSB's can still be employed, however.

SINGLE 10 BIT X/Y DISPLAY ACCUMULATOR SYSTEM



MECHANICS OF A DISPLAY
DIRECT LOAD INSTRUCTION



MECHANICS OF DRAWING