27
6/28/77
25-DATIS

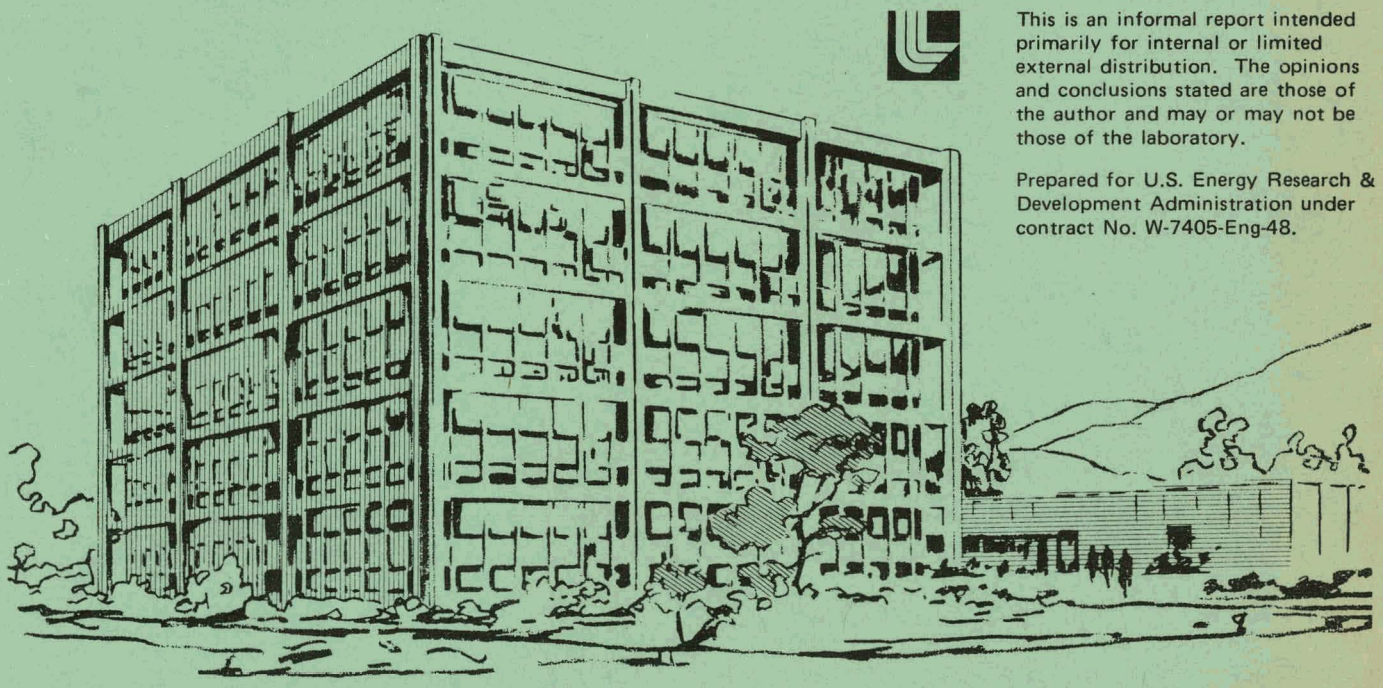
UCID- 17377

Lawrence Livermore Laboratory

AVMAC: AN ASSEMBLY LANGUAGE FOR THE AYDIN MODEL 5214A REFRESH DISPLAY GENERATOR

R. E. Werner

March 3, 1977



Prepared for U.S. Energy Research & Development Administration under contract No. W-7405-Eng-48.

DISCLAIMER

THIS PAGE

WAS INTENTIONALLY

LEFT BLANK

CONTENTS

AVMAC: AN ASSEMBLY LANGUAGE FOR THE AYDIN 5214A REFRESH DISPLAY GENERATOR

ABSTRACT

We present an instruction set and program listing for AVMAC, an assembly language for the Aydin 5214A Refresh Display Generator. AVMAC runs on a Varian V-70 series computer. We describe the mnemonics and appropriate parameter list, for each Aydin operation, needed to generate the Aydin machine code.

INTRODUCTION

AVMAC is an assembly language for the Aydin 5214A Refresh Display Generator, a cross assembler that runs on a Varian V-70 series computer with Vortex System.[1-3] The code consists of macros, for the Varian DASMR Macro Assembler, which define Aydin display generator operations. Aydin mnemonics along with appropriate parameter lists generate the Aydin machine code. Often used Aydin operations or sequences of operations can be defined with macros.

In the following we describe each operation of AVMAC. Words in italics are optional or default to a zero value. A six-character alphanumeric symbol can be used as a tag in the first position in each line. All lines which start with an * (asterisk) are treated as comments. The last line of the code must be an END card. Each Aydin operation is described by its mnemonic followed by a parameter list where each item is delimited by commas. Two adjacent commas may be used to specify 0. Commas are not necessary if the label, OP code, and variable field conventions of the DASMR assembler are observed.

For mnemonics, the mneominic usually corresponds to the value 1. Default is the opposite case and corresponds to the value 0. Note that all numbers may be written as decimal or octal. All octal numbers must be preceded by a 0; decimal numbers may not be preceded by 0.

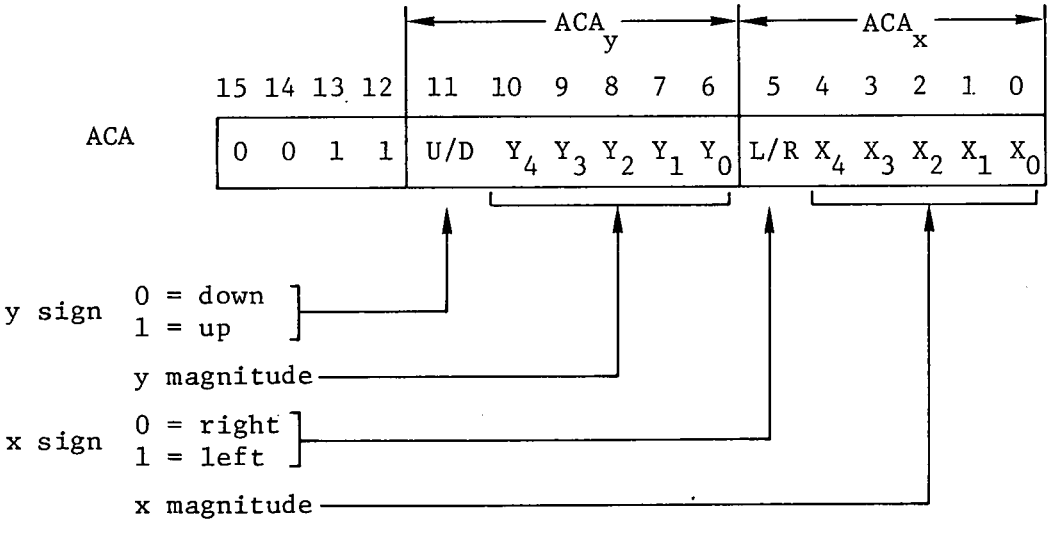
Some optional symbols are defined at the end of the program listing in Appendix B.

INSTRUCTION SET

Each display generator instruction is a 16-bit word, consisting of an OP code field (4 to 7 bits) and one or more additional fields. All of the instructions are single word instructions except BXM (block transfer mode), and the optional instructions LLU (load look-up) and LPF (load programmable font), which must be followed by one or more data words.

Instruction mnemonics, word formats, and bit assignments are given with each instruction description below.

Automatic Cursor Advance (ACA) Word



Usage. Type

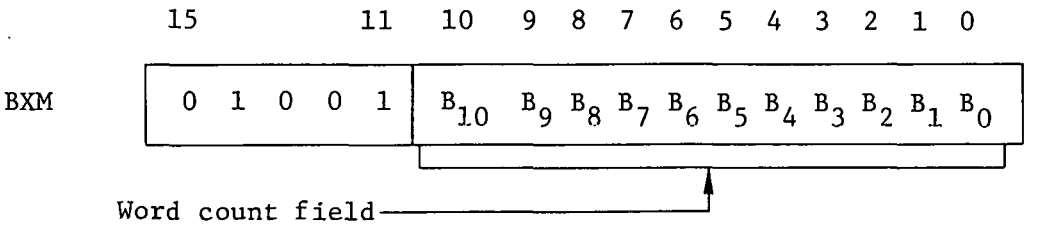
symbol ACA, [up / down / 0 / 1 / *] , y value, [left / right / 0 / 1 / *] , x value

where * is any user defined symbol equal to either 0 or 1, or omission of this parameter for 0.

The ACA word specifies the amount by which the cursor is to be moved each time a memory write operation is performed in the graphic mode. As noted in the MCW (mode control word) description below, the cursor movement is predetermined by the selected font in the alphanumeric mode if the A/N flag of MCW is 0.

Each of the fields, ACA_y and ACA_x, specifies a magnitude and direction. The magnitude of the cursor advance may be 0 to 31 raster lines in the y direction and 0 to 31 picture elements in the x direction.

Block Transfer Mode (BXM)



The BXM instruction is used to transfer a block of data words from the host computer to the refresh memory. The block length is given by the number of words specified in the word count field (up to 2047). BXM must always be followed by the number of words specified in the word field count field. The data words following BXM may represent graphic data or character codes, as explained below. Vectors and conics can be written only by the EXC (execute conic instruction) command, and cannot be specified by a block transfer, except as graphic data.

Usage. Type

symbol, BXM, wordcount

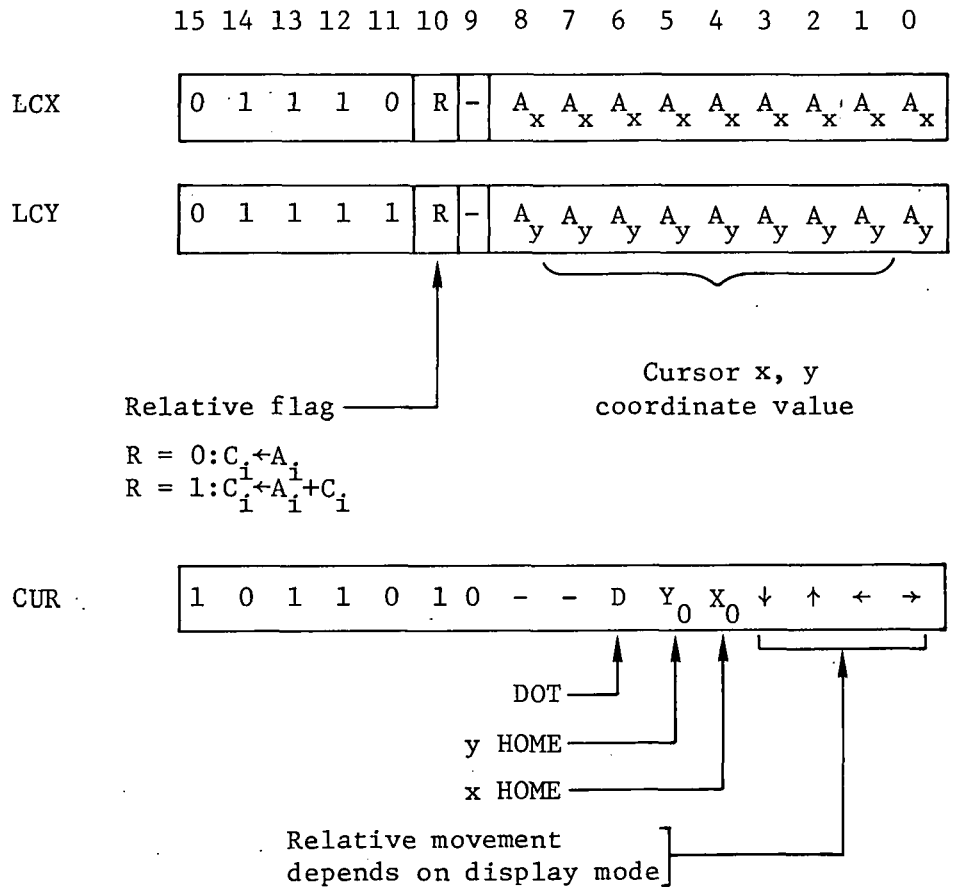
symbol, DATA, word 1, word 2, ..., word n

for graphics or

symbol, DATA, 'abcdefg'

for alphanumerics where each word is decimal or octal. The two characters per word in the Varian are packed in reverse order in the Aydin. Therefore, to write 'abcdefg' you must type 'badcfe g'.

Cursor Positioning (LCX, LCY, and CUR)



The start address of refresh memory read and write operations is specified by the cursor x and y coordinates in the picture element matrix. The least significant 9 bits of LCX, LCY are loaded into the cursor counters upon execution. If the relative flag is set, the values of A_x or A_y are added to the present cursor location C_x or C_y. If the result of LCX or LCY places the cursor outside the current values of the limits, an overflow error message is sent to the computer.

The CUR instruction allows movement of the cursor in both the x and y directions without issuing a pair of LCX, LCY commands. This instruction is similar to a relative LCX or LCY, except that the number of picture elements by which the cursor position is incremented is specified as shown in Table 1. With the appropriate bit settings, CUR provides common functions such as carriage return and line feed (CR/LF), backspace and line feed (BS/LF), HOME, etc.

Table 1. Cursor advance vs display mode.

Display mode	Cursor x	Cursor y
Graphic	ACA_x (0 to 31)	ACA_y (0 to 31)
Alphanumeric (A/N bit of MCW = 1)	ACA_x (0 to 31)	ACA_y (0 to 31)
Alphanumeric (A/N bit of MCW = 0)		
5 × 7 font	8	10
7 × 9 font	10	14
10 × 14 font	16	16
16 × 15 prog. font	x cur. adv.	y pitch (see LPF)

Usage. Type

symbol, LCX, [Re / 0 / 1 / *] , x value

symbol, LCY [Re / 0 / 1 / *] , y value

symbol, CUR, octal expression, *dot*

In addition to the Aydin instructions the following operations are available. They all use the CUR instruction.

To move the cursor home, type

symbol, HOME

For carriage return and line feed, type

symbol, CRLF

For back space and line feed, type

symbol, BSLF

For line feed, type

symbol, LF

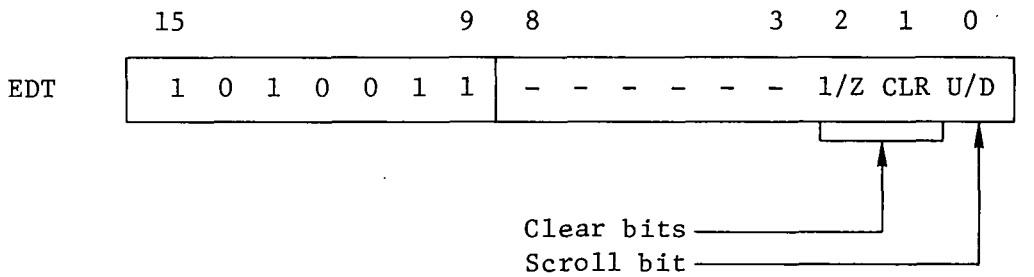
For carriage return, type

symbol, CR

For space; type

symbol, SPACE

Scrolling and Clearing: Edit Instruction (EDT)



The edit instruction provides for clearing any combination of selected channels simultaneously, either to all 1's or all 0's. In addition, all data within the rectangular limits may be scrolled up or down one raster line with the scroll instruction. The three least significant bits of the edit instruction are decoded as shown in Table 2.

Table 2. Least significant three bits of edit instruction.

Bit No.			
2 1/Z	1 CLR	0 SCR	Meaning
0	0	0	Scroll down
0	0	1	Scroll up
0	1	x	Clear to 0's
1	1	x	Clear to 1's

Scroll

Zeros in bits 1 and 2 of EDT define the scroll instruction. This instruction causes the displayed data in selected channels to move up or down one vertical element. The scroll operation is performed only on data within the previously defined rectangular limits. Data in each selected channel is scrolled, starting with the lowest channel number and proceeding until all selected channels have been scrolled. The execution time for the scroll instruction depends on the number of channels selected, N, and the values of the rectangular limits:

$$t_{scroll} = N \left\{ \frac{RL-LL+1}{16} \times (BL-TL) \right\} \times 4.8\ \mu s$$

for RL-LL = 512, BL-TL = 512, N = 1, and t_{scroll} = 79 ms.

Clear

Bit 1 of EDT specifies that a memory clear operation is to be performed. All selected memory channels are cleared to 0's if bit 1 = 0 or are cleared to 1's if bit 1 = 1. Selected channels are simultaneously cleared within the current values of the rectangular limits, and the maximum execution time for 512 × 512 is 28 ms.

Usage.

To set to 0's, type

symbol, CLEAR

To set to 1's, type

symbol, SET1S

To scroll up, type

symbol, SCRLUP

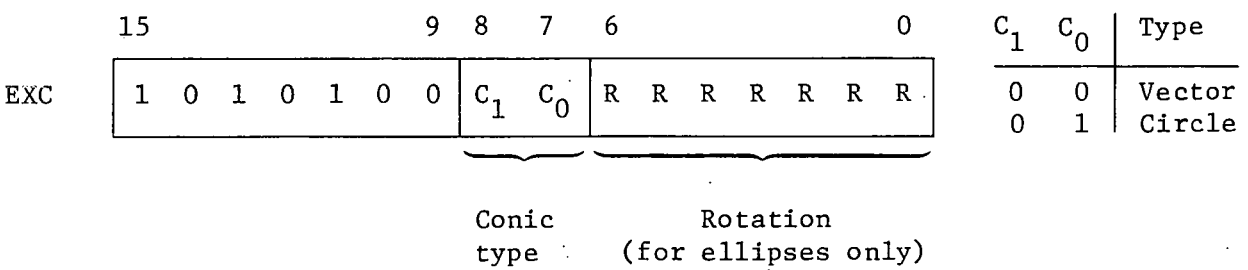
To scroll down, type

symbol, SCRLDN

Vector/Conic Generator: Execute Conic Instruction (EXC)

Discrete approximations to vectors, and circles, are computed and written into all selected memory channels by the EXC instruction. The

rectangular limits within which conics are written are specified by LRR (load rectangular limits, right), LRT (load rectangular limits, top), and LRB (load rectangular limits, bottom), instructions with the conic limit flag set. These limits are independent of the limits used to define areas to be scrolled, cleared, transmitted, or written into with BXM, LGE (load graphic elements), or LAC (load alphanumeric character) instructions.



C_1	C_0	Type
0	0	Vector
0	1	Circle

Vectors

The end point coordinates are loaded into the index x and y registers. The start point is the cursor location. Having specified these parameters, an EXC instruction, set for vectors, will cause the vector to be drawn. The cursor is positioned at the end point when the vector is complete. End-to-end vectors are drawn by respecifying the end point and issuing another EXC instruction. Random vectors are drawn by reloading both the cursor and index registers and issuing EXC. Computation and execution require 40 μs/point.

Usage. Type

symbol, VECTOR, x end point value, y end point value

This results in code which loads the x-index register and the y-index register and executes the vector.

Circles

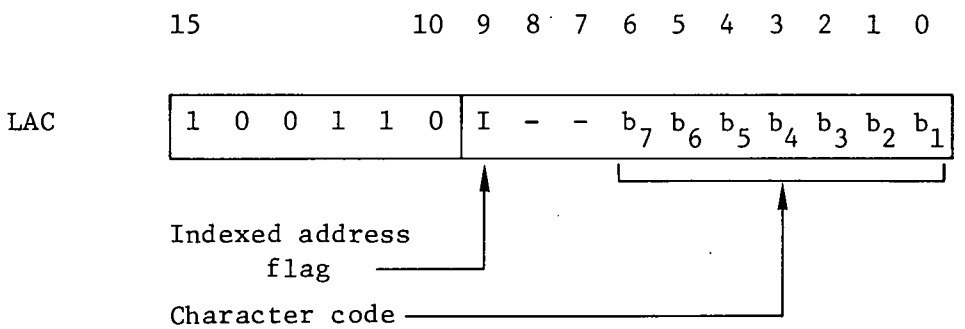
The radius is stored in the index x register, and the center point is the cursor location. The circle thus defined is then drawn by the EXC instruction set for circles. Upon completion of the circle, the cursor is positioned at $C_{x0} + R$, C_{y0}, where the subscripts x_0 and y_0 are the center coordinates. The time to execute a circle is 80 μs/point.

Usage. Type

symbol, CIRCLE, radius

This results in code that loads the x-index register and executes the circle.

Load Alphanumeric Character (LAC)



The seven least significant bits of LAC specify one of 128 possible character codes. This code addresses all of the character fonts, but only the output of the font specified in MCW will be written. Limit checks are performed before the character is written and the cursor is advanced after the write operation has been performed. Bit 9 of LAC permits indexed addressing as described for LGE. MCW must specify the alphanumeric mode when LAC is used. Characters may also be written in the block transfer mode.

Character Block Definition

Associated with each character font is a fixed character block area within which the character specified by LAC is written. The cursor position at the start of LAC is the top left picture element of the character block area, and the character is written one line at a time from top to bottom.

If different adjacent character spacings are required, the A/N ACA flag of MCW may be set and the desired spacings specified by an ACA instruction

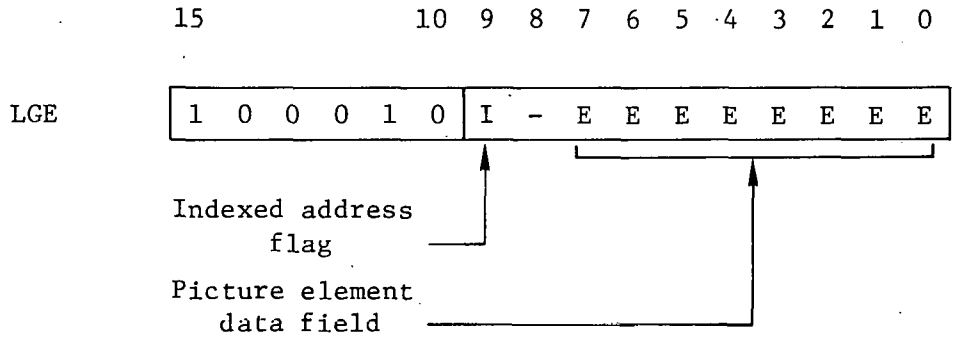
Usage. Type (for upper case)

symbol, LAC, 'ASCII character', [Ind / 0 / 1 / *]

The ASCII character to be displayed is enclosed in single quotes (shift 7). For lower case symbols use the octal or decimal equivalent of the ASCII character. Type

symbol, LAC, number, $\begin{bmatrix}\text{Ind}\\0\\1\\*\end{bmatrix}$

Load Graphic Elements (LGE)



This instruction writes eight bits of picture elements data into all selected memory channels, starting at the memory location addressed by the cursor x and y coordinates. If bit 9 is set, the actual starting address for the memory write is $C_x + I_x$, $C_y + I_y$, where C is the cursor coordinate and I denotes the index register content.

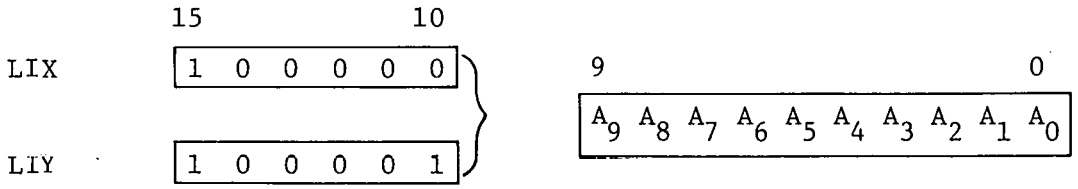
Data for single picture elements is entered using LGE with the OR 1's memory input mode selected. After LGE has performed the memory write, the cursor is advanced to the right by the number of picture elements specified by ACA. The programmer may enter from 1 to 8 graphic elements and move the cursor the desired amount (1 to 32 picture elements) by the setting of ACA_x. The graphic display mode must be selected before LGE is executed. Graphic data may also be entered in the block transfer mode.

Usage. Type

symbol, LGE, picture elements, $\begin{bmatrix}\text{Ind}\\1\\0\\*\end{bmatrix}$

where IND causes indexing.

Load Index Registers (LIX and LIY)



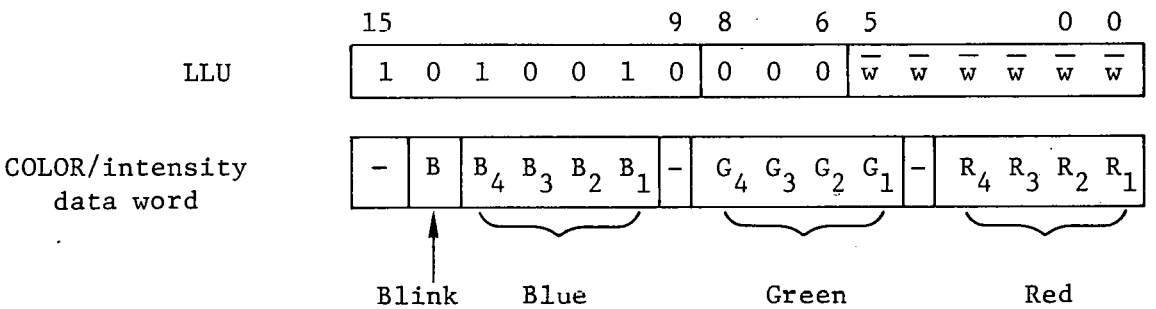
The least significant 10 bits of LIX and LIY specify the contents of the x and y index registers, respectively. The index registers provide indexed refresh memory addressing for the LAC and LGE display instructions, as well as storage of parameters used in specifying conics (see the EXC instruction).

Usage. Type

symbol, LIX, x value

symbol, LIY, y value

Load Look-Up Table (LLU)



LLU is a two-word instruction used to initially define the color data words stored in the look-up table. The w field of LLU specifies the address in the color table at which the subsequent color data word is stored. The color data word has three 4-bit fields, one for each primary color (red, green, and blue). Each field represents 1 of 16 possible intensity levels for each primary color. The use of 4 bits per primary color gives 2^{12} or 4096 different color data words. The color table provides storage for any 64 of the 4096 possible colors that the programmer wishes to use. However, only 2^N ($N \leq 6$ memory channel) color data words may be addressed. For example, 5 memory channels permit 32 colors to be simultaneously displayed. The LLU instruction may be used to alter the color of a displayed image without altering the contents of any of the refresh memory channels.

Usage. Type

symbol, LLU, word number, color

Note: word number is not complemented.

The assembler recognizes the following full intensity colors. BLACK, WHITE, RED, GREEN, BLUE, MAGENT, CYAN, YELLOW.

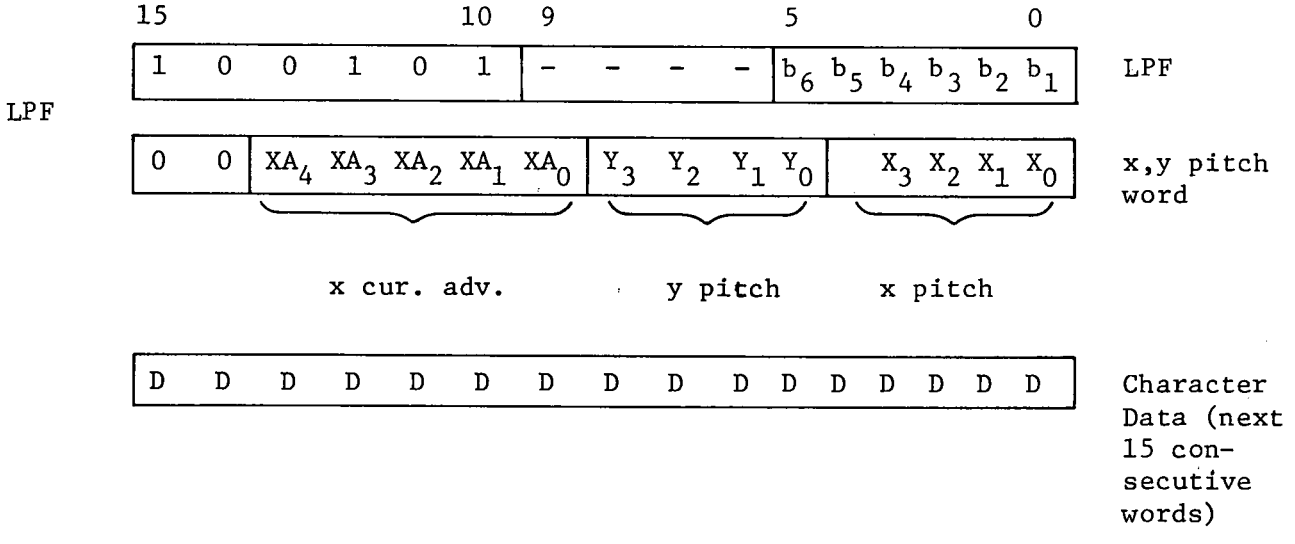
For blinking of the above colors use the following: BWHITE, BRED, BGREEN, BBLUE, BMAGEN, BCYAN, BYELLOW.

All the other possible colors of other intensities may be defined by the user in the following manner: type

color, EQU, octal expression or decimal number

where the octal expression is the color/intensity word described above. The color EQU statement must precede its use in an LLU statement.

Programmable Font: Load Programmable Font (LPF)



LPF is a multiple word instruction used to specify any one of the 64 characters in the programmable font. The 16 words which follow LPF are stored in sequential locations of the programmable font RAM (random access memory), starting at the address specified by bits 0 through 5 of LPF. To load all 64 characters, it is necessary to issue 64 different LPF instructions, each followed by 16 data words in the format shown above. An x pitch field value of 0 specifies and x pitch of one picture element; the value 15 specifies an x pitch of 16.

Each programmed character may have a y dimension of 2 to 15 picture elements, and an x dimension of 1 to 16 picture elements is specified in the x and y pitch fields. Cursor advance information is also specified by the pitch word. After a programmed character is written, the XA (x advance) field determines the amount of cursor x advance, and the y pitch field determines the cursor y advance when a new line is begun. The y pitch field also specifies the character block y dimension. Note that when loading the font, LPF must be followed by a pitch word and 15 data words, even if the programmed character occupies less than 15 scan lines. When the character is displayed, only the number of lines specified in the y pitch field are displayed.

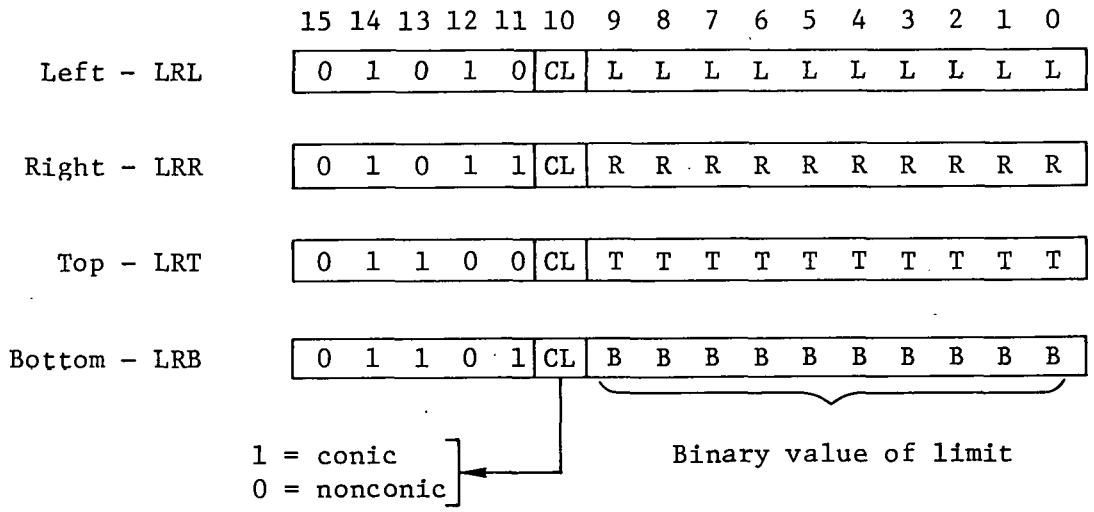
Usage. Type

symbol, LPF, [symbol name / octal expression / decimal number] , x cur. adv., y pitch, x pitch

where the symbol name is defined by an EQU statement.

symbol, DATA, word 1, word 2, ... , word 15

Load Rectangular Limits (LRR, LRL, LRT, and LRB)



The 10 least significant bits of the limit instructions are the binary values of the rectangular limits, though only the 9 least significant bits are normally used. Two independent sets of limits are stored - one set for use in defining arcs of conics in conjunction with the EXC instruction (entered with the conic flag bit set), and one set for use in all other

operations. Limit values entered with the conic flag bit reset define a rectangular area within which all subsequent operations on refresh memory data are performed. Data which has been previously stored outside the current limits are still displayed. The limit values apply to all memory channels in the unit.

The origin of x,y coordinates is in the upper left-hand corner of the CRT screen. The positive x direction is from left to right, and the positive y direction is from top to bottom. The top and bottom limits determine the range through which the cursor y coordinate may move. When the cursor y exceeds LRB, the cursor y is automatically reset to LRT (vertical wraparound). LRT must always be less than LRB. When the cursor x coordinate exceeds LRR, the cursor x is automatically repositioned at the left limit (horizontal wraparound) and the cursor y coordinate is advanced as directed by ACA or the font specified in MCW.

A specification of the rectangular limits is essential for block transfer inputs (see the BXM instruction for further details), defining areas in which data is to be read back to the computer (see transmit selected channel, TSC, below), and defining areas to be scrolled or cleared (see the EDT instruction).

For graphic or alphanumeric block transfers, as well as for the scroll and transmit operations, the left and right limit values are truncated by the hardware so that the quantity (LRR - LRL + 1)/16 is an integer between 1 and 32; the four LSB's of LRL are truncated to 0's and the four LSB's of LRR are rounded to 1's.

Usage. Type

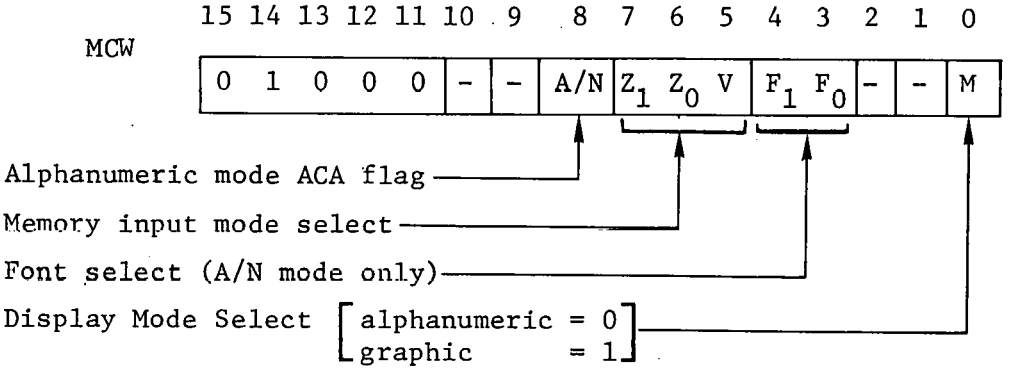
symbol, [LRL / LRR / LRT / LRB] , [* / CONIC / 1 / 0] , octal expression or decimal number

Example:

symbol, LRL

will set nonconic left limit to 0.

Mode Control Word (MCW)



Display Mode Selection

If bit 0 is set, graphic information may be written into the refresh memory using the LGE or BXM instructions. If bit 0 is reset, alphanumerics may be written using LAC or BXM instructions.

Font Selection

In the alphanumeric mode (bit 0 = 0), bits 3 and 4 of MCW are decoded to select the outputs of one of four character fonts. After a character is written, the cursor is advanced in x and, if the right limit is exceeded, in y by the number of picture elements shown in Table 3.

Table 3. Standard font cursor advance.

Bit No. 4	Bit No. 3	Font selected	Cursor advance in selected font (number of picture elements) x	y
0	0	5 × 7	8	10
0	1	7 × 9	10	14
1	0	10 × 14	16	16
1	1	16 × 15 (programmable font)	x cur.adv.	y pitch

Bits 3 and 4 have no effect in the graphic mode. The x and y cursor advances shown are used unless bit 8 of MCW is set.

Alphanumeric Mode Cursor Advance Flag

When bit 8 of MCW (the A/N flag) is set, the cursor advance values shown above do not apply. Instead, the x and y cursor advance information is specified by the ACA word.

Memory Input Mode Selection

The Z_1, Z_0, and V bits (bit numbers 7,6,5) of MCW determine the manner in which data is entered into the selected refresh memory channels. The Z bits are decoded to select one of three memory input modes.

MCW bit No.	7	6	5	
Designation	Z_1	Z_0	V	Memory input mode
	0	0	0	OR 1's
	0	1	1/0	Replace data
	1	0	0	Erase 1's
	1	1	X	Not defined

Note that the V bit (bit 5) applies only in the "Replace data" mode. Any memory input mode may be selected while in either the alphanumeric or graphic display mode. The three modes differ in the manner in which input data is conditioned prior to being written into the refresh memory. An example is given for each mode in the paragraphs which follow.

OR 1's Memory Input Mode

In this memory input mode, only input data bits which are logic 1's are entered into the refresh memory, while data bits which are logic 0's are ignored. In this manner, input data is essentially OR'ed with existing data. For example, suppose the eight-bit pattern shown below (representing eight picture elements on a raster line) is written in the OR 1's mode, and that the existing data in the same locations is as shown. By writing only the input data logic 1's, the input data is essentially OR'ed with the existing data.

```
1 0 1 1 1 1 0 1 - Existing data
1 0 1 0 1 0 1 0 - Input data
---------------
1 0 1 1 1 1 1 1 - Result when OR 1's mode is
                  specified
  ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑
  |_|_|_|_|_|_|_|_ 1's written in locations
  |   |   |   |     corresponding to logic 1's
  |   |   |   |
  |___|___|___|_____ Existing data left unchanged since
                      input 0 bits are not written
```

Erase 1's Memory Input Mode

In this mode, input data bits that are logic 1's are written as 0's, and as in the OR 1's mode, input data bits which are 0's are ignored. The result or an erase 1's memory input is shown below for the same bit patterns used in the OR 1's example.

```
1 0 1 1 1 1 0 1 - Existing data
1 0 1 0 1 0 1 0 - Input data
---------------
0 0 0 1 0 1 0 1 - Result when erase 1's mode
                  is specified
  ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑
  |_|_|_|_|_|_|_|_ 0's written in locations
  |   |   |   |     corresponding to input logic 1's
  |   |   |   |
  |___|___|___|_____ Existing data left unchanged since
                      input 0 bits are not written
```

Replace Data Memory Input Mode

This memory input mode simply replaces existing data with new data. If the V bit (bit 5), of MCW is 0, the input data replaces the existing data. If the V bit is 1, the existing data is replaced by the logical inversion of the input data.

```
1 0 1 1 1 1 1 1 - Existing data
1 0 1 0 1 0 1 0 - Input data
---------------
1 0 1 0 1 0 1 0 - Result when replace data mode
                  is specified and V = 0
0 1 0 1 0 1 0 1 - Result when replace data mode
                  is specified and V = 1
```

The V bit is normally used to display alphanumerics as colored symbols on a black background (V = 0) or as black symbols on a colored background (V = 1).

Usage. Type

symbol, MCW, [*, 1, 0] , [ORIS, RPDALS, *, OCTAL#, ERS1S, RPDA] , [F5x7, F7x9, F10x14, FPROG, OCTAL#, *] , [*, 1, 0]

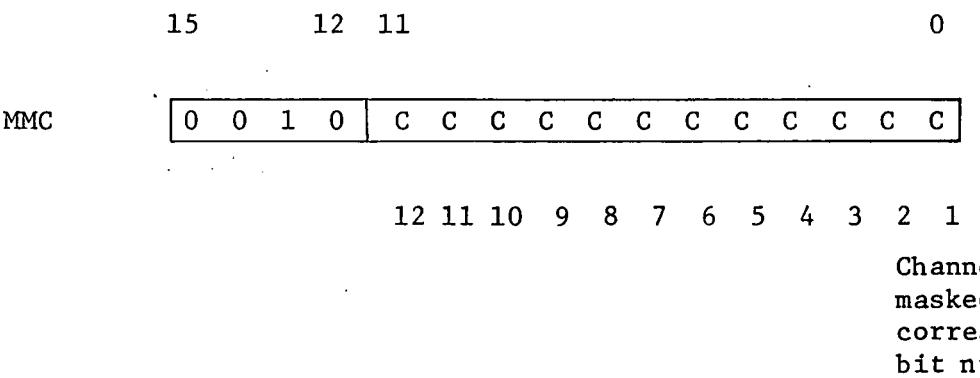
[*, 1, 0]: To use ACA type 1 or *

[ORIS, RPDALS, *, OCTAL#, ERS1S, RPDA]

[F5x7, F7x9, F10x14, FPROG, OCTAL#, *]

[*, 1, 0]: Mode graphic(1) or alpha-numeric(0)

Masking.



MMC causes the memory channel inputs indicated in the channel mask field to be disabled, inhibiting further entry of data into the corresponding memory channel. This instruction is mainly for use by an executive routine for memory protection.

To ensure consistency in programs involving protected channels, only unselected channels may be masked, and only unmasked channels may be selected. Selecting a masked channel causes an error, deselection of the offending channels, and selection of only unmasked channels thereafter. Similarly, the error caused by masking a selected channel is cleared by unmasking the channel.

Usage. Type

symbol, MMC, octal expression

Example.

symbol, MMC, 016

This masks channels 2, 3, and 4.

No Operation (NOOP)

Usage. Type

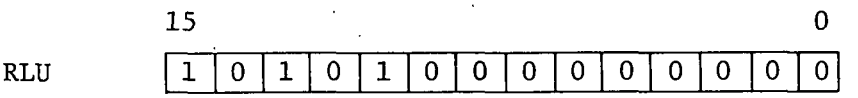
symbol, NOOP

This generates a 0 word.

The Aydin OP code mnemonic NOP conflicts with the usage of NOP by the Varian DASMR assembler. Thus the name has been changed to NOOP.

Read Look-Up Table (RLU)

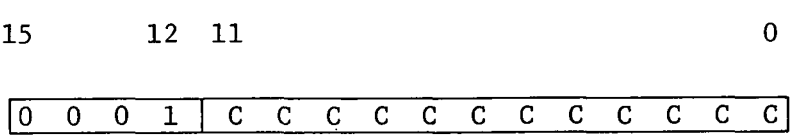
The RLU instruction causes the display generator to transmit to the computer the contents of the look-up table. The display generator will transmit an LPF word specifying the specific look-up table and address as well as the corresponding data word for each of the 64 locations.



Usage. Type

symbol, RLU

Channel Selection: Select Major Channel (SMC)



The SMC instruction contains a 12-bit channel select field. Each bit of the channel select field (bits 0 to 11) enables the inputs of one of up to twelve 512 × 512 refresh memory channels. The SMC instruction is used to select any combination of memory channels for use in subsequent instructions.

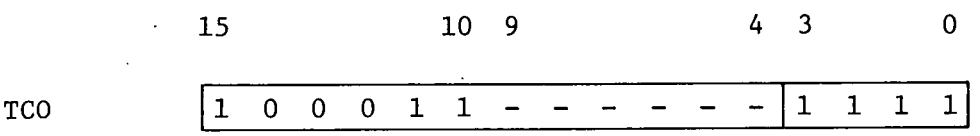
After SMC, any data written into the refresh memory (characters, vectors, etc.) are entered in the same locations in each of the selected channels. A channel that is not selected will receive no further input data, but its previous contents will continue to be displayed. SMC is used in conjunction with MMC to provide memory protection (described earlier). SMC may be issued many times to construct a complex picture.

Usage. Type

symbol, SMC, [octal expression | ALL | *]

Cursor Position Readback: Transit Cursor Only (TCO)

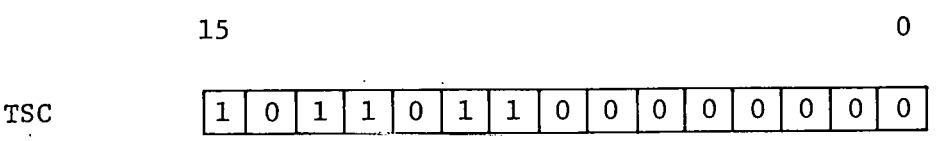
The TCO instruction causes the display generator to transmit the current contents of the cursor x and y registers to the computer.



Usage. Type

symbol, TCO

Memory Data Readback: Transit Selected Channel (TSC)



The TSC instruction causes the display generator to transmit the data within the truncated rectangular limits in each selected memory channel, one channel at a time. The data returned are picture element information for each raster line of each memory channel. Display instructions are inserted

where necessary so that the display generator output of memory data may be retransmitted at a later time to recreate the displayed image originally read back.

Usage. Type

symbol, TSC

REFERENCES

1. *Programming Information, Model 5214A Display Generator by Aydin Controls*, Aydin Controls, Fort Washington, PA, pp. 1-32.
2. *Varian V73 System Handbook*, Publication 98A9906 011, Varian Data Machines, Irvine, CA, pp. 15-1 — 15-27.
3. *Varian Software Handbook, Vol. 1*, Publication 98A 9952 201, Varian Data Machines, Irvine, CA, pp. DAS 1-DAS 40.

KLC/hr/jn/mla

APPENDIX A: SUMMARY OF USAGE

Automatic cursor advance.

symbol, ACA, [Up / Down / 0 / 1 / *] , y value, [LEFT / RIGHT / 0 / 1 / *] , x value

where * is any user defined symbol equal to either 0 or 1, or omission of this parameter for 0.

Block transfer.

symbol, BXM, wordcount

symbol, DATA, word 1, word 2, ..., word n

(for graphic) or

symbol, DATA, 'abcdefg'

(for alphanumerics).

Cursor positioning.

symbol, CUR, octal expression, *dot*

Move cursor home.

symbol, HOME

Carriage return and line feed.

symbol, CRLF

Back space and line feed.

symbol, BSLF

Line feed.

symbol, LF

Carriage return.

symbol, CR

Space.

symbol, SPACE

To set zeros.

symbol, CLEAR

To set ones.

symbol, SET1S

To scroll up.

symbol, SCRLUP

To scroll down.

symbol, SCRLDN

To draw a circle.

symbol, CIRCLE, radius

This results in code that loads x-index register and executes the circle.

To display a character.

symbol, LAC, [octal expression / decimal number / 'ASCII character'] , [IND / 0 / 1 / *]

Load cursor x.

symbol, LCX, [Re / 0 / 1 / *] , x value

Load cursor y.

symbol, LCY, [Re / 0 / 1 / *] , y value

Load graphic elements.

symbol, LGE, [Ind / 1 / 0 / *] , picture elements

Load index x.

symbol, LIX, x value

Load index y.

symbol, LIY, y value

Load look-up table.

symbol, LLU, word number, color

color, EQU, octal expression or decimal number

Load programmable font.

symbol, LPF, [symbol name / octal expression] , X cur. adv., y pitch, x pitch

symbol, DATA, word 1, word 2, ... , word 15

Load limits.

symbol, [LRL / LRR / LRT / LRB] , [* / CONIC / 1 / 0] , octal expression or decimal number

Mode control word.

symbol, MCW, [* / 1 / 0] , [OR1S / RPDA1S / ERS1S / RPDA / * / OCTAL#] , [F5×7 / F7×9 / F10×14 / FPROG / OCTAL# / *] , [* / 1 / 0]

Mask channels.

symbol, MMC, octal expression or decimal number

No operation.

symbol, NOOP,

Read look-up table.

symbol, RLU,

Select, channels.

symbol, SMC, [decimal number / octal expression / ALL / *]

Transmit cursor only.

symbol, TCO,

Transmit selected channel.

symbol, TSC,

Draw a vector.

symbol, VECTOR, x end point value, y end point value

APPENDIX B: PROGRAM LISTING

```
                1 * AYDIN CROSSASSEMBLER MAY 1976  BY  R. E. WERNER   NOT COPYRIGHTED
                2 NOOP    MAC
                3         DATA    0
                4         EMAC
                5 SMC     MAC
                6         DATA    010000+P(1)
                7         EMAC
                8 MMC     MAC
                9         DATA    020000+P(1)
               10         EMAC
               11 ACA     MAC
               12         DATA    (((06+P(1))*32+P(2))*2+P(3))*32+P(4)
               13         EMAC
000000 A       14 DOWN    EQU     0
000001 A       15 UP      EQU     01
000001 A       16 LEFT    EQU     01
000000 A       17 RT      EQU     0
               18 *  ACA,U/D,Y,L/R,X
               19 *
               20 *   ,MCW,FLAG,OPERATION,FONT,MODE
000000 A       21 OR1S    EQU     0
000003 A       22 RPDA1S  EQU     03
000002 A       23 RPDA    EQU     02
000004 A       24 ERS1S   EQU     04
000001 A       25 ALP     EQU     01
000000 A       26 NONALP  EQU     0
               27 MCW     MAC
               28         DATA    040000+P(1)*256+P(2)*32+P(3)*8+P(4)
               29         EMAC
000000 A       30 F5X7    EQU     0
000001 A       31 F7X9    EQU     01
000002 A       32 F10X14  EQU     02
000003 A       33 FPROG   EQU     03
               34 *
               35 * BXM,WORDCOUNT
               36 BXM     MAC
               37         DATA    044000+P(1)
               38         EMAC
               39 *
               40 * LRL,CONIC, LEFT LIMIT
               41 LRL     MAC
               42         DATA    (024+P(1))*1024+P(2)
               43         EMAC
000001 A       44 CONIC   EQU     01
               45 *
               46 * LRR,CONIC, RIGHT LIMIT
               47 LRR     MAC
               48         DATA    (026+P(1))*1024+P(2)
               49         EMAC
               50 *
               51 * LRT,CONIC TOP LIMIT
               52 LRT     MAC
               53         DATA    (030+P(1))*1024+P(2)
               54         EMAC
               55 *
               56 * LRB,CONIC, BOTTOM LIMIT
               57 LRB     MAC
               58         DATA    (032+P(1))*1024+P(2)
               59         EMAC
               60 *
               61 * LCX,RE,X VALUE
000017 A       62 ALL     EQU     017
000001 A       63 RE      EQU     01
               64 LCX     MAC
               65         DATA    (034+P(1))*1024+P(2)
```

```
                    66          EMAC
                    67 *
                    68 * LCY,RE,Y VALUE
                    69 LCY      MAC
                    70          DATA    (036+P(1))*1024+P(2)
                    71          EMAC
                    72 *
                    73 * LIX,  INDEX X VALUE
                    74 LIX      MAC
                    75          DATA    0100000+P(1)
                    76          EMAC
                    77 *
                    78 * LIY,  INDEX Y VALUE
                    79 LIY      MAC
                    80          DATA    0102000+P(1)
                    81          EMAC
                    82 *
                    83 * LGE,IND,PICTURE ELEMENTS
000001 A            84 IND      EQU     01
                    85 LGE      MAC
                    86          DATA    0104000+P(1)+P(2)*512
                    87          EMAC
                    88 *
                    89 * TCO        OR TCO,KA IFKEYBOARDS  THEN DATA,0106000P+(1)
000017 A            90 KA       EQU     017
                    91 TCO      MAC
                    92          DATA    0106017
                    93          EMAC
                    94 *
                    95 * LPF,SYMBOL NUMBER,X CUR ADV., Y PITCH, X PITCH
                    96 *,DATA,..15 DATA WORDS,..
                    97 LPF      MAC
                    98          DATA    0112000+P(1)
                    99          DATA    (P(2)*16+P(3))*32+P(4)
                   100          EMAC
                   101 *
                   102 * LAC,'CHAR',INX
000001 A           103 INX      EQU     01
                   104 LAC      MAC
                   105          DATA    0114000+P(1)+P(2)*512
                   106          EMAC
                   107 *
                   108 * LLU,WORD,COLOR
                   109 LLU      MAC
                   110          DATA    077-P(1)+0122000
                   111          DATA    P(2)
                   112          EMAC
000000 A           113 BLACK    EQU     0
036757 A           114 WHITE    EQU     036757
000017 A           115 RED      EQU     017
000740 A           116 GREEN    EQU     0740
036000 A           117 BLUE     EQU     036000
036017 A           118 MAGEN    EQU     036017
036740 A           119 CYAN     EQU     036740
000757 A           120 YELLOW   EQU     0757
076757 A           121 BWHITE   EQU     076757
040017 A           122 BRED     EQU     040017
040740 A           123 BGREEN   EQU     040740
076000 A           124 BBLUE    EQU     076000
076017 A           125 BMAGEN   EQU     076017
076740 A           126 BCYAN    EQU     076740
040757 A           127 BYELLO   EQU     040757
                   128 *
                   129 * CLEAR  SCREEN
                   130 CLEAR    MAC
                   131          DATA    0123002
                   132          EMAC
                   133 *
                   134 * SET TO ONES
                   135 SET1S    MAC
                   136          DATA    0123006
                   137          EMAC
                   138 *
```

```
                    139 * SCROLL UP
                    140 SCRLUP MAC
                    141         DATA    0123001
                    142         EMAC
                    143 *
                    144 * SCROLL DOWN
                    145 SCRLDN MAC
                    146         DATA    0123000
                    147         EMAC
                    148 *
                    149 * EXECUTE CONIC OR VECTOR
                    150 * VECTOR,X END POINT,Y END POINT
                    151 VECTOR MAC
                    152         LIX     P(1)
                    153         LIY     P(2)
                    154         DATA    0124000
                    155         EMAC
                    156 *
                    157 * CIRCLE,RADIUS
                    158 CIRCLE MAC
                    159         LIX     P(1)
                    160         DATA    0124200
                    161         EMAC
                    162 *
                    163 * READ LOOKUP TABLE
                    164 RLU     MAC
                    165         DATA    0125000
                    166         EMAC
                    167 *
                    168 * CURSOR, OCTAL CODE
                    169 *   .CUR,OCTALCODE,DOT
000001 A            170 DOT     EQU     1
                    171 CUR     MAC
                    172         DATA    0132000+P(1)+P(2)*64
                    173         EMAC
                    174 HOME    MAC
                    175         DATA    0132060
                    176         EMAC
                    177 CRLF    MAC
                    178         DATA    0132030
                    179         EMAC
                    180 BSLF    MAC
                    181         DATA    0132012
                    182         EMAC
                    183 BS      MAC
                    184         DATA    0132002
                    185         EMAC
                    186 LF      MAC
                    187         DATA    0132010
                    188         EMAC
                    189 CR      MAC
                    190         DATA    0132020
                    191         EMAC
                    192 SPACE   MAC
                    193         DATA    0132001
                    194         EMAC
                    195 *
                    196 EXC     MAC
                    197         DATA    0124000+P(1)
                    198         EMAC
                    199 *
                    200 * TRANSMIT SELECTED CHANEL
                    201 TSC     MAC
                    202         DATA    0133000
                    203         EMAC
                    204 *
                    205         END
```

```
ENTRY NAMES
EXTERNAL NAMES
SYMBOLS
000017 A ALL     000001 A ALP     076000 A BBLUE   076740 A BCYAN
040740 A BGREEN  000000 A BLACK   036000 A BLUE    076017 A BMAGEN
040017 A BRED    076757 A BWHITE  040757 A BYELLO  000001 A CONIC
036740 A CYAN    000001 A DOT     000000 A DOWN    000004 A ERS1S
000002 A F10X14  000000 A F5X7    000001 A F7X9    000003 A FPROG
000740 A GREEN   000001 A IND     000001 A INX     000017 A KA
000001 A LEFT    036017 A MAGEN   000000 A NONALP  000000 A OR1S
000001 A RE      000017 A RED     000002 A RPDA    000003 A RPDA1S
000000 A RT      000001 A UP      036757 A WHITE   000757 A YELLOW
  0 ERRORS ASSEMBLY COMPLETE
```

INDEX

Printed in the United States of America
Available from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Road
Springfield, VA 22161
Price: Printed Copy $; Microfiche $3.00

Page Range	Domestic Price	Page Range	Domestic Price
001–025	$ 3.50	326–350	10.00
026–050	4.00	351–375	10.50
051–075	4.50	376–400	10.75
076–100	5.00	401–425	11.00
101–125	5.50	426–450	11.75
126–150	6.00	451–475	12.00
151–175	6.75	476–500	12.50
176–200	7.50	501–525	12.75
201–225	7.75	526–550	13.00
226–250	8.00	551–575	13.50
251–275	9.00	576–600	13.75
276–300	9.25	601–up	*
301–325	9.75		

*Add $2.50 for each additional 100 page increment from 601 to 1,000 pages; add $4.50 for each additional 100 page increment over 1,000 pages.