208

BNWL-B-58

MASTER

RECEIVED BY DTIC JUN 9 1971

PDP-9/GOULD 4800 DEVICE HANDLER
AND GRAPHICS LIBRARY

MARCH 1971

PREPARED FOR
THE UNITED STATES ATOMIC ENERGY COMMISSION
UNDER CONTRACT AT(45-1)-1830

BATTELLE-NORTHWEST
PACIFIC NORTHWEST LABORATORIES DIVISION
BATTELLE MEMORIAL INSTITUTE
P. O. BOX 999, RICHLAND, WASHINGTON 99352

BNWL-B-58

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency Thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

INFORMATION CONCERNING USE OF THIS REPORT

PRELIMINARY REPORT

This report contains information of a preliminary nature prepared in the course of work under Atomic Energy Commission Contract AT(45-1)-1830. This information is subject to correction or modification upon the collection and evaluation of additional data.

LEGAL NOTICE

This report was prepared as an account of work sponsored by the United States Government. Neither the United States nor the United States Atomic Energy Commission, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness or usefulness of any information, apparatus, product or process disclosed, or represents that its use would not infringe privately owned rights.

PACIFIC NORTHWEST LABORATORY

operated by

BATTELLE MEMORIAL INSTITUTE

for the

U. S. ATOMIC ENERGY COMMISSION

Under Contract AT(45-1) 1830

PDP-9/GOULD 4800 DEVICE HANDLER
AND GRAPHICS LIBRARY

by

L. H. Gerhardstein

This report was prepared as an account of work sponsored by the United States Government. Neither the United States nor the United States Atomic Energy Commission, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness or usefulness of any information, apparatus, product or process disclosed, or represents that its use would not infringe privately owned rights.

COMPUTERS AND CONTROL

March 1971

PDP-9/GOULD 4800 Device Handler and Graphics LibraryINTRODUCTION

The WADCO PDP-9/AD-4 hybrid computer has recently been equipped with a GOULD 4800 graphics line printer. This system, operated by Measurement and Control Technology Department - Control and Data Systems Section, is used for hybrid and digital simulation of nuclear and other processes.

The standard PDP-9 software contains a device independent I/O monitor system for communication with most I/O devices. In order that the GOULD 4800 could be used with the PDP-9 system, an I/O device handler and a graphics library were written. This software enables the PDP-9 programmer to use the GOULD 4800 either in a standard high-speed line printer mode or in graphics mode.

INPUT/OUTPUT DEVICE HANDLER - CIA

The device handler written for the GOULD 4800 printer is formatted according to the PDP-9 Advanced Software system requirements ^(1,2). Therefore, this printer can be used by any standard system program (MACRO, PIP, FORTRAN) or user program for making listings, etc. Alphanumeric mode (listings) is established by assigning either Unit 1 (CIA1) or 2 (CIA2). Unit 1 corresponds to small font (5x7 dots) while 2 corresponds to large font (7x10 dots). Graphics mode is established when unit 0 (CIA0) is assigned. Appendix A gives detailed description of the CIA. device handler characteristics.

GRAPHICS LIBRARY

The graphics library is a set of subroutines to generate graphical data on the GOULD 4800 printer. A summary of the library components is

given in Appendix B, and detailed program characteristics are given in Appendix C.

The subroutines in the graphics library allow the user to

- Draw solid (or pseudo-solid) straight lines
- Draw individual dots
- Draw characters from an ASCII character generation table
- Draw characters from a user defined character generation table.
- Perform real-time or non-real-time graphics output.
- Use any program in the ASCII output library to generate alpha-numeric graphics. (3)

Looping Technique

A page of graphics is generated by a user program by a "looping" technique. The core resident buffer that the graphics routines use is 10 scan lines in length (1/8 inch). By looping the user graphics generation routine, a graphics page of any length (up to 1638 inches) can be generated.

Example listing #1 shows the looping technique. The macros GRXL, GRYL, and GRMSG are defined by

```
.DEFIN      GRXL,XØ,YØ,L
.GLOBL      GR.I,GR.XL
.DEC
LAC         (XØ
DAC*        GR.I
LAC         (YØ
JMS*        GR.XL
L
.ENDM
.DEFIN      GRMSG,R,XØ,YØ,MSG
.GLOBL      GR.I,GR.YL
.DEC
LAC         (XØ
DAC*        GR.I
```

```

LAC          (YØ
JMS*         GR.YL
L
ENDM

.DEFIN       GRMSG,R,XO,YO,MSG
.DEC
.GLOBL      WMSG,GRWMSG,GR.P57
LAW
JMS*         GR.P57
XØ
YØ
JMS*         WMSG.
.DSA        MSG$4
.ENDM

```

Macro GRXL defines a line in the x direction starting at XØ,YØ, L dots in length extending to the right. The GRYL macro defines a line in the Y direction starting at XØ,YØ, L dots in length extending down (increasing y). The GRMSG macro defines a message text graphics starting at XØ,YØ with rotation R. The message is standard 5 x 7 ascii (.ASCII pseudo-op. in MACRO,HYMAC).

Referring to example listing 1 and figure 1:

Line 1: Initializes CIA. device handler and sets up API interrupt

Line 2: Initializes page pointers at top of page

Line 3-16: Graphics loop to draw a box with a message inside the box

Line 3: Clears the ten scan line buffer

Lines 4-7: Draws the box sides

Line 8: Generate the message at GR.PY = 11₁₀, no rotation

Line 9-10: Causes the graphics line buffers to be written to DAT slot No. 1

Lines 11-13: Tests to determine if 300 scan lines have been sent

Line 14: Jumps out of graphics loop when drawing is finished

Line 16: Continues the loop if page vertical posn. < 300

Line 17: Defines the message, MESE, sent by GRMSG at line 8.

Example Listing 1

```

1      .INIT      1,1,0      /INIT DEVICE HANDLER
2      JMS*       GR.IN      /INIT PAGE COUNTERS
3 LOOP JMS*       GR.CLR     /CLEAR TEN S.L. BUFFER
4      GRXL      100,100,161 /BOX TOP
5      GRYL      100,101,29  /BOX LEFT SIDE
6      GRYL      260,101,29  /BOX RIGHT SIDE
7      GRXL      100,130,161 /BOX BOTTOM
8      GRMSG      0,110,11,MESS /MESSAGE
9      LAW        1          /DAT SLOT FOR .WRITE
10     JMS*       GR.WR      /.WRITE AND .WAIT
11     LAC*       GR.PJ      /VERT. PAGE POSN
12     .DEC
13     SAD        (300      /300 SCAN LINES?
14     JMP        DONE      /END OF PAGE
15     .OCT
16     JMP        LOOP      /PROCESS NEXT TEN SCAN LINE
                             BUFFER
17 MESS .ASCII    /GOULD GRAPHICS/<177>

```

FIGURE 1: Graphics Output and Coordination
of Example Listing 1.

The "looping technique, described above, will work only because the drawing of graphics components by GRXL, etc., is conditional, i.e.: Library subroutine GR.XL will perform a drawing procedure only if the YØ coordinate of the line is within the current scan line buffer.

$$(GR.PJ) \quad Y \emptyset \quad (GR.PD) + 9$$

A similar rule applies to other graphics routines (GR.YL, GR.MSG, etc.).

Graphics Device Independence

Graphics output from the graphics routines is device independent in nature in that the binary output from GR.WR, etc., can be sent to any device capable of handling binary data (paper punch, DEC tape, etc., but not teletype, etc.). To obtain I/O device flexibility from a graphics routine, it is therefore desirable to use automatic device assignment (as opposed to manual assignment with the keyboard monitor). The automatic method is shown in example listing #2. The three entry points will select Gould printer, paper punch, and dectape unit 1 respectively. The selected device handler is assigned to DAT slot 1.

Example Listing 2

.DAT = 135

	.GLOBL	CLA.,PPA.,DTD.
GOULD	LAC	CLA.
	JMP	TAPE+2
PUNCH	LAC	PPA.
	JMP	TAPE+2
TAPE	LAC	DTD.
	XOR	(10000000
	DAC*	(.DAT+1
	.INIT	1,1,0
	.ENTER	1,FILE

```

      JMS*      GR.IN      / file name & extension /
      { ETC. }
      .CLOSE      1
      FILE      .SIXBT      / file name & extension /

```

Graphics Line Buffer Compression

When graphics data is to be sent to a device other than the GOULD 4800 it is desirable to use the compressed data form. The ten scan line buffers are compressed by subroutine GR.CKS. The compressed data file on decatape or disc will be 1/3 or less the size of an uncompressed file providing a substantial saving in mass storage. When line buffer compression is desired, subroutine GR.CKS should be called immediately before GR.WR is called:

```

      JMS*      .GLOBL      GR.CKS, GR.WR
      JMS*      GR.CKS      /COMPRESS LINE BUFFERS
      LAW      1      /DAT SLOT NO.
      JMS*      GR.WR      /.WRITE AND .WAIT

```

Double Buffered Graphics Output

An increase in graphics output speed will be realized when using double buffering techniques, as shown in example listing 3. Two buffers are used, the internal graphics buffer and GRLB2. The address of the graphics routine internal buffer is reported by GR.IN. Subroutine WRITE first waits for any outputting of the previous line buffer to be completed and then writes the current buffer using RTWRIT. Subroutine RTWRIT first reformats the current buffer to be written into a single line buffer and executes a write instruction. The CLA. device handler will write the data at the interrupt level so that no further user programming is required to send the 10 scan lines. After the write is initiated, the buffer pointer GR.LB is switched to the other buffer.

Example Listing 3

```

BGN      {Initialize}
          JMS*      GR.IN      /INITIALIZE PAGE POSN.
          DAC       GRLB1#     /INTERNAL BUFFER ADDR
          DAC*      GR.LB
          JMS*      GR.CLR     /CLEAR BUFFER
          JMS       RIWRIT     /WRITE ZEROS TO START ACTION
          JMS*      GR.IN      /INITIALIZE PAGE POSN.

LOOP      { Calculate
          { Graphics Line Buffers }

          JMS       WRITE
          {Test for end-of-page}

          JMP       LOOP

WRITE     Ø
          JMS*      GR.CK      /DONE WITH PREVIOUS BUFFERS
          SKP
          JMP       .-2       /NO
          JMS       RIWRIT     /REAL-TIME WRITE
          JMP*      WRITE

RIWRIT    Ø
          LAC*      GR.LB      /CURRENT BUFFER ADDR
          DAC       IDX#
          LAC       (676ØØØØ /WPC = 276, IGNORE CHECK-SUM
          DAC*      IDX       /LINE BUFFER HEADER
          LAW       -11
          DAC       CNT#      /9 ADDITIONAL LINE BUFFERS
          LAC       IDX

WRIT1     TAD       (46
          DAC       IDX       /ADVANCE TO NEXT LINE BUFFER
          DZM*      IDX       /CLEAR HEADER WORD
          ISZ       IDX       /CLEAR CHECK-SUM WORD
          LAW       -1
          TAD       IDX
          ISZ       CNT       /LAST L.B.
          JMP       WRIT1     /NO
          JMS*      GR.RLW     /START .WRITE
          LAC*      GR.LB      /CURRENT L.B.
          SAD       GRLB1     /SWITCH L.B.'S
          JMP       .+3
          LAC       GRLB1     /LIBRARY BUFFER
          JMP       .+2
          LAC       (GRLB2     /USER BUFFER
          DAC*      GR.LB      /NEW L.B. POINTER
          JMS*      GR.CLR     /CLEAR NEW BUFFER
          JMP*      RIWRIT

GRLB2     .BLOCK      574      /GRAPHICS BUFFER

```

EXAMPLE PROGRAM

An example graphics routine (GRTTEST) is shown in example listing 4. The output from this program is shown in Figure 2. The characteristics of this example program and its techniques are:

- GRTTEST outputs eleven inches of graphics (880 scan lines).
- The four macros, GRXL, GRYL, GRMSG and GRLIN perform the generation of graphics constants. (Axis lines, header messages, etc.).
- One of three output device handlers, CLA., PPA., or DTD. can be selected by starting at BEGIN, PCH, or DT.
- Double buffered output is used. The standard size for ten scan line buffers is $380_{10} = 574_8$.
- The graphics loop address is LOOP. This loop is executed 88 times to generate an 11 inch page.
- The portion of GRTTEST that draws the curve gets its input from TABLE, a table of scaled fixed point numbers. The curve is drawn by segmenting straight line segments with GR.LIN.
- Subroutine WRITE writes the current line buffer and then switches line buffers. The new buffer is cleared.
- Location SWITCH is a switch which, when set to NOP, causes line buffer compression to occur. (GR.CKS)
- When 880 scan lines have been sent, control of the CPU is returned to HYDDT9. (DDTSA.)

FIGURE 2: Output from Example Listing #4.

The figure shows the GOULD output from example listing #4. Printing starts at the top and proceeds down, ten scan lines at a time.

EXAMPLE LISTING 4

.TITLE GRAPHICS TEST

/L H GERHARDSTEIN

/JAN 27, 1971

.DAT=135

.SCOM=1000

/MACRO DEFINITIONS FOR GRAPHICS

/X LINE

```

      .DEFIN      GRXL,X0,Y0,L
      .GLOBL      GR.I,GR.XL
      .DEC
      LAC         (X0
      DAC*        GR.I
      LAC         (Y0
      JMS*        GR.XL
      L
      .ENDM

```

/Y LINE

```

      .DEFIN      GRYL,X0,Y0,L
      .GLOBL      GR.I,GR.YL
      .DEC
      LAC         (X0
      DAC*        GR.I
      LAC         (Y0
      JMS*        GR.YL
      L
      .ENDM

```

/TEXT MESSAGE

```

      .DEFIN      GRMSG,R,X0,Y0,MSG
      .DEC
      .GLOBL      WMSG., GR.P57
      LAW         R
      JMS*        GR.P57
      X0
      Y0
      JMS*        WMSG.
      .DSA        MSG$4
      .ENDM

```

```

/LINE
.DEFIN      GRLIN,X1,Y1,X2,Y2
.DEC
.GLOBL      GR.LIN
JMS*        GR.LIN
X1
Y1
X2
Y2
.ENDM

.GLOBL      CLA.,PPA.,DTD,
.GLOBL      GR.IN,GR.CLR,GR.CKS,GR.R57
.GLOBL      GR.RLW,GR.CK,GR.PJ,GR.LB
.GLOBL      GR.CTØ,GR.CHT
.GLOBL      DDTSA.

BEGIN      LAC      CLA.          /GOULD-4800 D.H. ADDR
           SKP
PCH        LAC      PPA.          /PAPER PUNCH D.H. ADDR
           JMP      DT+2
DT         LAC      DTD.          /DECTAPE D.H. ADDR
           XOR      (1ØØØØØØØ    /DECTAPE UNIT 1
           DAC*     (.DAT+1      /ASSIGN TO DAT+1
           LAC      GR.CTØ       /CHARTABLE ADDRESS
           DAC*     GR.CHT       /CHAR GEN CHAR TABLE ADDR

           .INIT     1,1,Ø       /INITIALIZE D.H.
           .ENTER    1,FILE      /ENTER FILE

           JMS*      GR.IN       /INIT PAGE POINTERS
           DAC*      GR.LB       /SET L.R. LOCN
           DAC       GR.LB#      /POINTER TO L.B. #1
           JMS*      GR.CLR      /CLEAR BUFFERS
           JMS       RTWRIT      /WRITE ZERO BUFFERS
           JMS*      GR.IN       /INIT PAGE POINTERS

LOOP=.

/DRAW AXES & HEADER MESSAGES
GRYL      16Ø,19Ø,461
GRXL      155,25Ø,5
GRXL      15Ø,3ØØ,1Ø
GRXL      155,35Ø,5
GRXL      15Ø,4ØØ,1Ø
GRXL      155,45Ø,5
GRXL      15Ø,5ØØ,1Ø
GRXL      155,55Ø,5
GRXL      15Ø,6ØØ,1Ø
GRXL      15Ø,2ØØ,351

```


GRYL	210,195,5
GRYL	260,190,10
GRYL	310,195,5
GRYL	360,190,10
GRYL	410,195,5
GRYL	460,190,10

/DRAW COORD X'S

GRXL	250,300,21
GRXL	350,300,21
GRXL	450,300,21
GRXL	250,400,21
GRXL	350,400,21
GRXL	450,400,21
GRXL	250,500,21
GRXL	350,500,21
GRXL	450,500,21
GRXL	250,600,21
GRXL	350,600,21
GRXL	450,600,21

GRYL	260,290,21
GRYL	360,290,21
GRYL	460,290,21
GRYL	260,390,21
GRYL	360,390,21
GRYL	460,390,21
GRYL	260,490,21
GRYL	360,490,21
GRYL	460,490,21
GRYL	260,590,21
GRYL	360,590,21
GRYL	460,590,21

GRLIN	490,190,499,199
GRLIN	490,210,499,201
GRLIN	150,640,159,649
GRLIN	170,640,161,649
GRMSG	0,260,14,MSG1
GRMSG	0,156,17,MSG2
GRMSG	1,130,20,MSG3
GRMSG	1,100,30,MSG4

/DRAW CURVE

LAC*	GR.PJ	/PAGE Y COORD
DAC	Y1	
TAD	(11	
DAC	Y2	/Y2 = Y1 +9
LAC*	GR.PJ	
TAD	(-310	/-200(10)

	SPA		
	JMP	END	/Y < 200
	DAC	SAVE#	
	TAD	(-620	/-400 (10)
	SMA		
	JMP	END	/Y ≥ 600
	LAC	SAVE	/Y1 - 200
	JMS	CALX	
	DAC	X1	
	LAC	SAVE	
	TAD	(12	/Y2 - 200
	JMS	CALX	
	DAC	X2	
	JMS*	GR.LIN	/DRAW LINE (X1,Y1); (X2,Y2)
X1	0		
Y1	0		
X2	0		
Y2	0		
END	JMS	WRITE	/WRITE L.B.'S
	LAC*	GR.PJ	/CURRENT PAGE POSN. SCAN LINES
	SAD	(1560	/880 (10)
	JMP	DONE	/END OF DISPLAY
	JMP	LOOP	/GENERATE NEXT TEN LINE BUFFER
DONE	JMS*	GR.CK	/CHECK IF DONE
	SKP		
	JMP	.-2	
	.CLOSE	1	
	.WAIT	1	
	JMS*	GR.R57	/RESTORE P57.
	JMP*	DDTSA.	/RETURN TO DDT
/CALCULATE X			
CALX	0		/AC = 100*(Y-Y0)
	1DIV120000		
	12		
	LACQ		/10*(Y-Y0)
	TAD	(TABLE	/ADDR OF X(Y-Y0)
	DAC	IDX#	
	LAC*	IDX	/2**13*X
	MUL		
	.DEC		
	6400		/2**18*(100/2**13)*2
			/AC,MQ = 100*2**19*X
			/AC = 2*(100*X)
	TAD	(160*2+1	/AC = 2*(100*X+160) +1
	.OCT		
	RCR		/AC = 100*X+160 ROUNDED
	JMP*	CALX	

/WRITE TEN L.B. BUFFER

WRITE	Ø		
	JMS*	GR.CK	/CHECK IF DONE WITH PREVIOUS BUFFER
	SKP		/YES
	JMP	.-2	/NO - WAIT
	JMS	RIWRIT	/INITIATE .WRITE
	JMP*	WRITE	

RIWRIT
SWITCH

Ø		
SKP		/OR NOP TO COMPRESS

JMP	WRITE3
-----	--------

LAC*	GR.LB	/CURRENT L.B. ADDR
------	-------	--------------------

DAC	IDX
-----	-----

LAC	(6760000	/276 WP = 574 WD CNT
-----	----------	----------------------

DAC*	IDX
------	-----

LAW	-11
-----	-----

DAC	CNT#
-----	------

LAC	IDX
-----	-----

WRITE1

TAD	(46
-----	-----

DAC	IDX
-----	-----

/NEXT L.G.

DZM*	IDX
------	-----

/CLEAR L.B.H. AND CKS

ISZ	IDX
-----	-----

DZM*	IDX
------	-----

LAW	-1
-----	----

TAD	IDX
-----	-----

ISZ	CNT
-----	-----

JMP	WRITE1
-----	--------

GIF	
-----	--

WRITE3

JMS*	GR.CKS
------	--------

WRITE2

LAW	1
-----	---

/.DAT SLOT NO. 1

JMS*	GR.RLW
------	--------

/REAL-TIME WRITE

LAC*	GR.LB
------	-------

/CURRENT L.B. ADDR

SAD	GRLB1
-----	-------

/SWITCH LINE BUFFERS

JMP	+.3
-----	-----

LAC	GRLB1
-----	-------

JMP	+.2
-----	-----

LAC	(GRLB2
-----	--------

DAC*	GR.LB
------	-------

JMS*	GR.CLR
------	--------

/CLEAR BUFFER

JMP*	RIWRIT
------	--------

GRLB2

.BLOCK 574

/SECOND GRAPHICS BUFFER

/FILE NAME SPECIFICATION

FILE	.SIXBT /FORCE/
	.SIXBT /GRF/

/MESSAGES

MSG1	.ASCII /FORCE (LB)/<177>
------	--------------------------

MSG2 .ASCII /Ø 1 2 3/<177>
 MSG3 .ASCII /Ø 1 2 3 4/<177>
 MSG4 .ASCII /DISPLACEMENT (IN)/<177>

/TABLE OF X VALUES OF PLOTTED CURVE

/	X	Y
TABLE	Ø.Ø	/Ø.
	Ø.Ø4	/Ø.1
	Ø.1	/Ø.2
	Ø.17	/Ø.3
	Ø.25	/Ø.4
	Ø.34	/Ø.5
	Ø.42	/Ø.6
	Ø.5Ø	/Ø.7
	Ø.6Ø	/Ø.8
	Ø.69	/Ø.9
	Ø.79	/1.Ø
	Ø.88	/1.1
	Ø.98	/1.2
	1.Ø7	/1.3
	1.16	/1.4
	1.25	/1.5
	1.34	/1.6
	1.43	/1.7
	1.52	/1.8
	1.6Ø	/1.9
	1.69	/2.Ø
	1.77	/2.1
	1.85	/2.2
	1.93	/2.3
	2.ØØ	/2.4
	2.Ø8	/2.5
	2.15	/2.6
	2.21	/2.7
	2.28	/2.8
	2.35	/2.9
	2.4Ø	/3.Ø
	2.46	/3.1
	2.51	/3.2
	2.56	/3.3
	2.6Ø	/3.4
	2.64	/3.5
	2.68	/3.6
	2.71	/3.7
	2.74	/3.8
	2.77	/3.9
	2.8Ø	/4.Ø

.END

PROGRAMMING TECHNIQUES

Appendix D gives a partial listing of the graphics library. These programs show the basic procedures to be used when the programmer wishes to define graphics procedures for the GOULD 4800.

User Defined Character Fonts

The listing of the graphics library character set is shown on pages D-1 and D-2. Each entry in the character table requires four words where each word consists of two eight bit character lines. "Left to right" on the GOULD printer corresponds to "Left to right" in an 8 bit character line for unrotated characters. Top and bottom on the GOULD printer corresponds to C1 and C8 in the character table for unrotated characters. The eight bit character lines are left adjusted so that a solid 8 bit character line is given as 776_8 .

To establish the use of the ASCII character table (GR.CTØ) the user must initialize the character generator (GR.CHR) table pointer (GR.CHT):

```
.GLOBL    GR.CØ,GR.CHT
LAC       GR.CTØ
DAC*      GR.CHT
```

The user can establish his own character fonts by setting up a table similar to that in Appendix D. The character table pointer (GR.CHT) can be switched at any time:

```
LAC       (TABLE    /USER TABLE
DAC*      GR.CHT    /TABLE POINTER
```

The standard ASCII character set is shown in figure 3. This GOULD output was generated by the program listed on page D6 (GRIST2).

User Defined Graphics

The Draw X-line subroutine (GR.XL) on page D3 shows the necessary steps for generation of a graphics element.

Subroutine GR.CKJ (page D4) checks the PJ coordinate (Y) to determine if it is within the current scan line buffer:

```

0123456789:;<=>?@ABCDEFGHIJKLMN O PQRSTU VWXYZ[\ ]↑←!
123456789:;<=>?@ABCDEFGHIJKLMN O PQRSTU VWXYZ[\ ]↑←!"
23456789:;<=>?@ABCDEFGHIJKLMN O PQRSTU VWXYZ[\ ]↑←!#"
3456789:;<=>?@ABCDEFGHIJKLMN O PQRSTU VWXYZ[\ ]↑←!#"#
456789:;<=>?@ABCDEFGHIJKLMN O PQRSTU VWXYZ[\ ]↑←!#"#%
56789:;<=>?@ABCDEFGHIJKLMN O PQRSTU VWXYZ[\ ]↑←!#"#%&
6789:;<=>?@ABCDEFGHIJKLMN O PQRSTU VWXYZ[\ ]↑←!#"#%&'
789:;<=>?@ABCDEFGHIJKLMN O PQRSTU VWXYZ[\ ]↑←!#"#%&'()
89:;<=>?@ABCDEFGHIJKLMN O PQRSTU VWXYZ[\ ]↑←!#"#%&'()*
9:;<=>?@ABCDEFGHIJKLMN O PQRSTU VWXYZ[\ ]↑←!#"#%&'()*+
:;<=>?@ABCDEFGHIJKLMN O PQRSTU VWXYZ[\ ]↑←!#"#%&'()*+,-
<=>?@ABCDEFGHIJKLMN O PQRSTU VWXYZ[\ ]↑←!#"#%&'()*+,-./
=>?@ABCDEFGHIJKLMN O PQRSTU VWXYZ[\ ]↑←!#"#%&'()*+,-./0
?@ABCDEFGHIJKLMN O PQRSTU VWXYZ[\ ]↑←!#"#%&'()*+,-./01
@ABCDEFGHIJKLMN O PQRSTU VWXYZ[\ ]↑←!#"#%&'()*+,-./012
ABCDEFGHIJKLMN O PQRSTU VWXYZ[\ ]↑←!#"#%&'()*+,-./0123
CDEFGHIJKLMN O PQRSTU VWXYZ[\ ]↑←!#"#%&'()*+,-./0123

```

FIGURE 3: Software generated ascii character set - GR.CTØ

$$(GR.PJ) \leq PJ \leq (GR.PJ) + 9$$

Subroutine GR.CKJ reports above, below, or with-in the current scan line buffer. If the requested X-line is above or below, exit from GR.XL occurs (JMP OUT). If the PJ coordinate is within the current buffer, drawing of the line is initiated. Drawing of the individual dots is performed by GR.SET.

If the user wishes to program the setting of dots within the scan line buffers, the GR.SET listing (page D5) should be used as a guide. This "set dot" subroutine first checks if the I (X) coordinate is within its legal limits

$$0 \leq I \leq 599$$

The relative word position of the dot within the scan line buffer is:

$$Rel. Posn. = (N_1 * J) + [I/18] + N_H + 1$$

where:

N_1 = Total no. words in one line
buffer = 38

N_H = No. words in line buffer
header = 2

I = Horizontal posn. of dot

J = Vertical posn. of dot
within current buffer

$[x] \stackrel{\Delta}{=} \text{Integer value in } x$

The dot bit position within the addressed word is:

$$\begin{aligned} \text{Bit Posn.} &= \text{Rem } (I/18) \\ &= I - ([I/18]*18) \end{aligned}$$

REFERENCES

- (1) "PDP-9 User Handbook" - Digital Equipment Corp., Maynard, Mass.
- (2) "Advanced Software System Monitors" - DEC-9A-MADO-D Digital Equipment Corp., Maynard, Mass.
- (3) "Hybrid Software System Library and Macros", October 21, 1970, L. H. Gerhardstein, BNWL-B-26.

APPENDIX A

GOULD 4800 DEVICE HANDLER PROPERTIES

APPENDIX A
GOULD 4800 DEVICE HANDLER PROPERTIES

Device Handler Name:	CLA
Legal Data Modes:	IOPS Binary IOPS Ascii
Interrupt:	API Channel 64
No. Words:	$744_8 = 484_{10}$
IOPS Functions:	
.INIT	Performs .SETUP Initializes mode & font characteristics
.FSTAT	Ignored
.RENAM	
.DELETE	
.SEEK	Illegal
.ENTER	Sends Auto Form Feed
.CLEAR	Ignored
.CLOSE	Sends Auto Form Feed Releases control of printer
.MTAPE	Ignored
.READ	Illegal
.WRITE	Transfers one line buffer of data to the printer in the mode selected by .INIT
.WAIT	I/O wait until printing finished
.WAITR	Real-Time I/O wait on printing done
.TRAN	Illegal

Character Generator Font Control:

The Mode and character font are determined by the unit number assigned when .INIT was executed. Unit 0 (CIA0) is for bit mode data while Units 1, 2, and 3 are for ascii character data. Unit 1 (CIA1) generates upright small font, and Unit 2 (CIA2) generates upright large font. Unit 3 can be used if the user wishes to specify his own font configuration.

The printer control words for the various unit assignments are:

Unit No.	Font	EF Word	Pre Wrd 1	Pre Wrd 2	Char Sp	Line Sp	Char Per Line	Lines Per 9" Page
0	Bit	00	0	144 ₈	0	0	100	720
1	Small	11 ₈	2	144 ₈	1	2	100	102
2	Large	61 ₈	2	113 ₈	1	2	75	60
3	-	CL.F1	CL.F2	CL.F3	-	-	-	-

CIA. Internal Global Addresses:

There are seven addresses in CIA. accessible by the user program:

Global Name	Normal Contents	Description
CL.F1	LAW 11	Unit 3 "EF" Word
CL.F2	2	Unit 3 Preamble Word No. 1.
CL.F3	144 ₈	Unit 3 Preamble Word No. 2.
CL.F4	120 ₈	Upper Page Margin (scan lines)
CL.F5	1320 ₈	No. Scan Lines per Page
CL.F6	120 ₈	Lower Page Margin (scan lines)
CL.F7	-12 ₈	Negative Tab Size (characters)

APPENDIX B

GRAPHICS ROUTINES SUMMARY

APPENDIX B
GRAPHICS ROUTINES SUMMARY

The following table gives a basic description of the programs and global addresses of the graphics package.

<u>PROGRAM NAME</u>	<u>INT. GLOBALS</u>	<u>PURPOSE</u>
GR.CTØ	GR.CTØ	ASCII Character Table
GR.CLR	GR.CLR	Subroutine to set ten scan line buffers to zeros.
	GR.IN	Subroutine to set Page Position pointers to top of page (y = 0)
	GR.LB	Line Buffers pointer
	GR.PJ	Vertical page position (scan lines)
	GR.PJN	Negative Vertical page position (scan lines)
	GR.PY	Vertical page position (text lines or no. of ten scan line buffers)
GR.CKS	GR.CKS	Subroutine to compress the ten scan line IOPS line buffers, and calculate the headers and check-sums.
GR.WR	GR.WR	Subroutine to write the ten scan line buffers to an arbitrary output device. (Non-real-time)
	GR.RLW	Subroutine to write the ten scan line buffers to an arbitrary output device. (Real-time)
	GR.CK	Subroutine to check done status of the real-time write subroutine GR.RLW.

GR.SET	GR.SET	Subroutine to set one point within a ten scan line buffer to a 1.
	GR.I	Horizontal Position of dot (0 to 599)
	GR.J	Vertical Position of dot within the ten scan line buffer (0 to 9)
GR.CKJ	GR.CKJ	Subroutine to check if a given vertical position is within the current ten scan line buffer.
GR.XL	GR.XL	Subroutine to draw a horizontal straight solid line with a specified starting position and length.
GR.YL	GR.YL	Subroutine to draw a vertical straight solid line with a specified starting position and length.
GR.CHR	GR.CHR	Subroutine to store a character representation in the ten scan line buffer with a specified position and rotation.
	GR.ROT	Character Rotation 0 Right-side-up 1 Clock-wise 2 Upside-down 3 Counter clock-wise
	GR.TI	Character horizontal position (dots)
	GR.TY	Character vertical position (text lines)
	GR.CHT	Character table pointer.
GR.LIN	GR.LIN	Subroutine to draw a pseudo-straight solid line between two points (x_1, y_1) ; (x_2, y_2) .

GR.P57

GR.P57

Subroutine to setup the IOPS ascii
output library to graphics mode.

GR.R57

Subroutine to restore the IOPS ascii
output library to normal mode.

APPENDIX C

GRAPHICS LIBRARY PROGRAM DESCRIPTIONS

APPENDIX C

GRAPHICS LIBRARY PROGRAM DESCRIPTIONS

The following pages give detailed descriptions of the graphics routines and their basic calling procedures. Execution times are for a PDP-9 digital computer (1 μ sec cycle time).

These pages can be removed and added to "Hybrid Software System Library and Macros" - BNWL-B-26.

GR.CTØ

Library form:

ASCII CHARACTER SET GENERATION TABLE

/To initialize:

.GLOBL	GR.CTØ, GR.CHT
LAC	GR.CTØ
DAC*	GR.CHT

/To calculate a character address

LAC	ASCII	/character
AND	(77	/6 Bit
ALSS+2		/4* Char
TAD*	GR.CHT	
DAC	ADDR	/Character Address

Execution time

N.A.

Description:

The GR.CTØ character table contains 8 x 8 representations for the 64 printing ascii characters (Ø4Ø to 137, seven bit codes). The address of a character in the table is:

$$\text{GR.CHT} + (4 * C)$$

where C = the six bit trim code.

The contents of a character table entry is defined by the macro definition:

```
.DEFIN CHAR, C1, C2, C3, C4, C5,
                                C6, C7, C8
```

```
C1*400+C2*2
```

```
C3*400+C4*2
```

```
C5*400+C6*2
```

```
C7*400+C8*2
```

```
.ENDM
```

C1 through C8 are a bit representation for the eight scan lines of the character starting at the top. A character line of 8 dots is represented by 776₈. e.g.: The coding for the letter "A" is:
CHAR 060,110,204,402,776,402,402,402

400 200 100 40 20 10 4 2 Bit Values

GR.CLR

Variables:

Library form:

Execution Time:

CLEAR GRAPHICS SCAN LINE BUFFERS

GR.LB = Line buffer printer

```
.GLOBL GR.LB, GR.CLR
```

```
Set GR.LB (optional)
```

```
JMS* GR.CLR
```

3031 μ sec (PDP-9)

Description:

Subroutine GR.CLR should be used to clear the graphics buffer at the beginning of each graphics program loop. The contents of GR.LB points to the current buffer address.

$380_{10} = 574_8$ words are required for a graphics buffer. ($38_{10} = 46_8$ words per scan line). The contents of each scan line buffer after execution of GR.CLR is:

423000	/Line buffer header
0	/Check-sum
400042	/Zero/data count
$\left\{ \begin{array}{l} 34_{10} = 42_8 \\ \text{zeros} \end{array} \right\}$	/Data
77777	/end of-line flag

GR.ININITIALIZE GRAPHICS PAGE POSITION

Variables:

GR.PJ = Page Vertical posn. (scan lines)
 GR.PY = Page Vertical Posn. (text lines)
 GR.PJN = Negative Page Vertical
 posn. (scan lines)

Library form:

```
.GLOBL GR.IN
JMS*   GR.IN
/AC = Address of graphics
/buffer in graphics
/subroutines (ignore if
/not double-buffering)
```

Execution Time:

13 μ sec (PDP-9)

Description:

Subroutine GR.IN initializes the graphics package page position to:

Ø → GR.PJ

Ø → GR.PY

Ø → GR.PJN

GRLB → AC_{EXIT}

GR.CKS

COMPRESS BUFFERS & COMPUTE CHECK-SUMS

Variables:

GR.LB = line buffers pointer

Library form:

.GLOBL GR.CKS

JMS* GR.CKS

Execution time:

approx. 2.2 milli-sec (PDP-9)

Description:

Subroutine GR.CKS reformats the ten scan line buffers into a reduced form that eliminates all zero data words and thus compresses the size of graphics data files by as much as 1/5 or less. GR.CKS need not be used if the graphics data is sent directly to the printer rather than to a mass storage device. After GR.CKS has been executed, the format of a graphics line buffer is:

WPCØØØ	/header
CKS	/checksum
4 (N1)Ø (M1)	/zero/data count-block 1
{ M1 data }	
{ words }	
4 (N2)Ø (M2)	/zero/data count-block 2
{ M2 data }	
{ words }	
:	
.	
.	
777777	/End-of-line Switch
Ø	/optional to make
	/total no. words even.

GR.SET

Variables:

SET A POINT IN GRAPHICS BUFFER

GR.J = Vertical posn. in current ten scan line buffer

AC_{in} = Horizontal posn of dot.

Library form:

.GLOBL GR.J, GR.SET

LAC Vert. Posn. in this buffer

DAC* GR.J

LAC Horiz. Posn.

JMS* GR.SET

Execution time:

Approx. 54 μ sec. (PDP-9)

Description:

Subroutine GR.SET is used by all graphics routines to set points within the ten scan line graphics buffer. The contents of GR.J must have been set to a number from 0 to 9 by the calling program.

GR.WR

Variables:

WRITE GRAPHICS LINE BUFFERS

AC (9-17)_{in} = .WRITE Dot slot no.

GR.LB = Line buffer pointer

Library form:

.GLOBL GR.LB, GR.WR

/Set GR.LB (optional)

LAW Dat Slot No.

JMS* GR.WR

Execution time:

150 μ sec + 10* (.WRITE + .WAIT of specified device handler) (PDP-9)

Description:

GR.WR is a non-real time subroutine to write the ten scan line buffers to an arbitrary output device.

Listing:

```

GR.WR      Ø
           JMS      GR.RLW
WAIT       JMS      GR.CK
           JMP*     GR.WR
           JMP      WAIT

```

GR.RLWGRAPHICS BUFFERS REAL-TIME WRITE

Variables:

```

AC (9-17)in = .WRITE Dat Slot No.
GR.LB = Line buffer pointer
GR.PJ = Page Posn. (S.L.)
GR.PJN = Neg. Page Posn. (S.L.)
GR.PY = Page posn. (T.L.)

```

Library form:

```

.GLOBL      GR.LB,      GR.RLn
/Set GR.LB (optional)
LAW      Dat Slot No.
JMS*      GR.RLW

```

Execution Time:

36 µsec (PDP-9)

Description:

Execution of GR.RLW sets up Subroutine GR.CK to perform real-time output of the ten scan line graphics buffers. After GR.RLW has been called, GR.CK should be called frequently until data transfer is completed. Once GR.RLW has been called, GR.LB can be modified and a new set of buffers can be generated (double-buffering).

The action of GR.RLW is as follows:

```

ACin → .WRITE dat for GR.CK
ACin → .WAITR dat for GR.CK
-128 → COUNT for GR.CK
GR.PJ+1010 → GR.PJ
-GR.PJ      → GR.PJN
GR.PY+1     → GR.PY
GR.LB       → GRLB for GR.CK

```

GR.CK

Variables:

Library form:

Execution time:

Description:

GRAPHICS REAL-TIME WAIT

N.A.

.GLOBL GR.CK

JMS* GR.CK

... /DONE RETURN

... /BUSY RETURN

9 μ sec + .WAITR if done return
 7 μ sec + .WAITR } if Busy
 or 19 μ sec + .WAITR + .WRITE } return

Subroutine GR.CK performs a real-time wait and write on the dat slot specified by GR.RLW. The action taken by GR.CK is:

GR.CKJCHECK VERTICAL POSITION

Variables:

AC_{in} = Vert. Page Posn. of dot
 GR.PJN = Neg. Vert. Page Posn. of
 current scan line buffers.
 GR.J = Vert. Posn. within current
 scan line buffers.

AC_{out} = GR.J

Library form:

```
.GLOBL  GR.CKJ
LAC      Vert. Dot Posn.
JMS*     GR.CKJ
...      /Above; AC = -DIST
...      /Below; AC = +DIST
...      /In   ; AC = Posn. in
           this buffer.
```

Execution time:

9 μ sec (PDP-9) above
 15 μ sec (PDP-9) below
 21 μ sec (PDP-9) with-in.

Description:

Subroutine GR.CKJ is used by such programs as GR.LIN, GR.CHR, and GR.YL to conditionalize the setting of dots with-in the ten scan line buffer.

```
.GLOBL  GR.PJN, GR.J
.GLOBL  GR.CKJ
```

Listing:

```
GR.CKJ  Ø
TAD*     GR.PJN
SPA!STL
JMP*     GR.CKJ      /Above
TAD      (-12
SMA!STL
JMP      .+4         /Below
TAD      (12
DAC*     GR.J        /Posn
ISZ      GR.CKJ
ISZ      GR.CKJ
JMP*     GR.CKJ
```


GR.XL

Variables:

DRAW HORIZONTAL SOLID LINE

AC_{in} = Vert. Posn. on Page at starting point.

GR.I = Horizontal Posn. on Page at starting point.

(JMSsp+1) = No. Dots in line

Library form:

.GLOBL GR.I, GR.XL

/Set GR.I

LAC Vert. page posn.

JMS* GR.XL

No. Dots in line

... /Return

Execution time:

20 μ sec (PDP-9) if outside current buffer.

or 37 μ sec + 68*N μ sec (PDP-9),
where N = No. dots, if within current buffer.

Description:

Subroutine GR.XL draws a solid horizontal line with specified starting point and length. The starting point is defined as the left line end point (minimum X).

GR.YL

Variables:

DRAW VERTICAL SOLID LINE

AC_{in} = Vert. posn. of starting point

GR.I = Horizontal posn. of starting point.

(JMS pt+1) = No. dots in line

Library form:

.GLOBL GR.I, GR.YL

/Set GR.I

LAC Vert. Posn.

JMS* GR.XL

No. dots in line

... /Return

Execution time:

$18 + 17N_A + 85.N_I + 22.N_B \text{ } \mu\text{sec (PDP-9)}$
 where: N_A = No. Pts above current buffer
 N_I = No. Pts in current buffer
 N_B = 1 if line extends below
 current buffer
 N_B = 0 if line doesn't extend
 below current buffer

Description:

Subroutine GR.YL draws a solid vertical line with specified starting point and length. The starting point is defined as the upper line and point (minimum y).

GR.CHRDRAW ALPHA-NUMERIC CHARACTER

Variables:

AC_{in} = 6 Bit ASCII trim code
 $GR.TI$ = Character I coordinate
 $GR.TY$ = Character text line no.
 $GR.ROT$ = Character Rotation
 0 Normal
 1 c.w.
 2 upside down
 3 c.c.w.

$GR.CHT$ = Character generation table
 pointer

Library form:

$.GLOBL \quad GR.TI, GR.TY, GR.ROT$
 $.GLOBL \quad GR.CHT, GR.CT0$
 $.GLOBL \quad GR.CHR$
 $LAC \quad GR.LT0$
 $DAC* \quad GR.CHT \quad /Set \text{ pointer}$
 $/Set \text{ } GR.TI, GR.TY, GR.ROT$
 $LAC \quad 6 \text{ Bit char}$
 $JMS* \quad GR.CHR$

Execution time:

40 μ sec (PDP-9) if character is not in the current buffer

1480 + (70 + 17R) N_1 μ sec (PDP-9) if character is in the current buffer

where R = Rotation (0-3)

N_1 = No. ones in char. font.

e.g.: The execution time for drawing the letter "A" is 3.0 to 4.1 milli-sec depending on rotation.

Description:

GR.CHR draws a character at specified position and rotation. A square 8 x 8 character font is used. (See GR.CTØ for format). The text character position is updated after each execution of GR.CHR as follows:

<u>GR.ROT</u>	<u>Action</u>
0	GR.TI + 10 $\rightarrow$ GR.TI
1	GR.TY + 1 $\rightarrow$ GR.TY
2	GR.TI - 10 $\rightarrow$ GR.TI
3	GR.TY - 1 $\rightarrow$ GR.TY

Updating of the character position in the latter manner causes sequential calling of GR.CHR to draw a character string in the direction specified by GR.ROT.

GR.LIN

DRAW PSEUDO-STRAIGHT SOLID LINE
BETWEEN TWO POINTS

Variables:

$$\left. \begin{aligned} (\text{JMS pt} + 1) &= X_1 \\ (\text{JMS pt} + 2) &= Y_1 \\ (\text{JMS pt} + 3) &= X_2 \\ (\text{JMS pt} + 4) &= Y_2 \end{aligned} \right\} \begin{array}{l} \text{Where the two} \\ \text{points are } (X_1, Y_1) \\ \text{\& } (X_2, Y_2) \end{array}$$

Library form:

.GLOBL GR.LIN

JMS* GR.LIN

X1

Y1

X2

Y2

... /Return

Execution time:

If $|X_2 - X_1| > |Y_2 - Y_1|$: $84 + 57 N_{OUT} + 116 N_{IN} \text{ } \mu\text{sec (PDP-9)}$ If $|X_2 - X_1| \leq |Y_2 - Y_1|$: $81 + 21 N_A + 116 N_{IN} \text{ } \mu\text{sec (PDP-9)}$

where:

 N_A = No. points above current buffer N_{OUT} = No. points outside current
buffer N_{IN} = No. points inside current
buffer.

Description:

Subroutine GR.LIN draws a pseudo-straight line between two specified points (A series of data, vertical lines, and horizontal lines that simulate the line).

GR.P57

SETUP IOPS ASCII OUTPUT LIBRARY TO
SEND GRAPHICS

Variables:

 $AC(16-17)_{in}$ = Character Rotation $(JMS_{pt} + 1)$ = Text starting I (x)₂
position. $(JMS_{pt} + 2)$ = Text starting text
line position.

Library form:

```

.GLOBL      GR.P57
LAW         Rotn. No.
JMS*        GR.P57
Starting X posn.
Starting text line
...         /Return
.
.
.
LAC         ASCII Char
JMS*        P57.      /Sends Character
                        /to graphics.

```

Execution time:

48 μ sec (PDP-9)
 Add 26 μ sec to GR.CHR when using
 P57.

Description:

Execution of GR.P57 causes all the
 IOPS ASCII output routines (WOCT.,
 WDEC., WMSG., etc.) to send their
 output to GR.CHR, the graphics
 character routine. The normal con-
 dition of P57. is:

```

P57.  Ø
      SKP
      Ø
      { Program to Pack }
      { 5 x 7 ASCII      }
      JMP*  P57.

```

After GR.P57 is executed, the condi-
 tion of P57. is:

```

P57.  Ø
      JMP*  P57.+2
      CATCH

```

Where CATCH is a routine in GR.P57
 that transfers control to GR.CHR.

GR.R57RESTORE P57.

Variables:

None

Library form:

.GLOBL GR.R57
JMS* GR.R57

Execution time:

16 μ sec (PDP-9)

Description:

Subroutine GR.R57 restores P57.
for normal packing of 5 x 7 ASCII.

Listing:

	.GLOBL	P57.
	.GLOBL	GR.R57
GR.R57	$\emptyset$	
	LAC	P57.
	DAC	POINT
	ISZ	POINT /P57.+1
	LAC	(SKP
	DAC*	POINT
	JMP*	GR.R57

APPENDIX D

PARTIAL GRAPHICS LIBRARY LISTINGS

.TITLE GRAPHICS CHARACTER TABLE

/ 9-28-70
/ L H GERHARDSTEIN

.DEFIN CHAR,C1,C2,C3,C4,C5,C6,C7,C8
C1*400+C2*2
C3*400+C4*2
C5*400+C6*2
C7*400+C8*2
.ENDM

GR.CT0	.GLOBL	GR.CT0	
CHAR	170,204,462,512,514,470,202,174	/ 00	@
CHAR	060,110,204,402,776,402,402,402	/ 01	A
CHAR	770,404,404,770,404,402,402,774	/ 02	B
CHAR	174,202,400,400,400,400,202,174	/ 03	C
CHAR	770,404,402,402,402,402,404,770	/ 04	D
CHAR	776,400,400,770,400,400,400,776	/ 05	E
CHAR	776,400,400,770,400,400,400,400	/ 06	F
CHAR	174,202,400,400,436,402,202,174	/ 07	G
CHAR	402,402,402,776,402,402,402,402	/ 10	H
CHAR	174,020,020,020,020,020,020,174	/ 11	I
CHAR	076,010,010,010,410,410,220,140	/ 12	J
CHAR	404,410,420,740,420,410,404,402	/ 13	K
CHAR	400,400,400,400,400,400,400,776	/ 14	L
CHAR	402,606,512,462,402,402,402,402	/ 15	M
CHAR	402,602,502,442,422,412,406,402	/ 16	N
CHAR	170,204,402,402,402,402,204,170	/ 17	O
CHAR	774,402,402,402,774,400,400,400	/ 20	P
CHAR	170,204,402,402,422,412,204,172	/ 21	Q
CHAR	774,402,402,402,774,410,404,402	/ 22	R
CHAR	374,402,400,374,002,002,402,374	/ 23	S
CHAR	376,020,020,020,020,020,020,020	/ 24	T
CHAR	402,402,402,402,402,402,204,170	/ 25	U
CHAR	402,402,402,204,110,110,060,060	/ 26	V
CHAR	402,402,402,402,462,462,264,110	/ 27	W
CHAR	402,204,110,060,060,110,204,402	/ 30	X
CHAR	202,202,104,050,020,020,020,020	/ 31	Y
CHAR	776,404,010,174,040,100,202,776	/ 32	Z
CHAR	170,100,100,100,100,100,100,170	/ 33	[
CHAR	400,200,100,040,020,010,004,002	/ 34	\
CHAR	170,010,010,010,010,010,010,170	/ 35	]
CHAR	020,070,124,020,020,020,020,020	/ 36	↑
CHAR	000,100,200,776,200,100,000,000	/ 37	←
CHAR	000,000,000,000,000,000,000,000	/ 40	SPACE
CHAR	060,060,060,060,060,000,060,060	/ 41	!
CHAR	154,154,154,000,000,000,000,000	/ 42	"
CHAR	110,110,776,110,110,776,110,110	/ 43	#
CHAR	374,512,510,374,112,112,512,374	/ 44	\$
CHAR	776,504,710,020,040,116,212,416	/ 45	%
CHAR	060,110,110,162,224,410,424,342	/ 46	&
CHAR	030,030,010,020,000,000,000,000	/ 47	'
CHAR	010,020,040,040,040,040,020,010	/ 50	(
CHAR	100,040,020,020,020,020,040,100	/ 51	)
CHAR	204,110,060,776,060,110,204,000	/ 52	*

CHAR	020,020,020,376,020,020,020,000	/ 53 +
CHAR	000,000,000,000,060,060,020,040	/ 54 ,
CHAR	000,000,000,376,000,000,000,000	/ 55 -
CHAR	000,000,000,000,000,000,060,060	/ 56 .
CHAR	002,004,010,020,040,100,200,400	/ 57 /
CHAR	172,204,214,224,244,304,204,570	/ 60 0
CHAR	160,020,020,020,020,020,020,174	/ 61 1
CHAR	374,402,402,002,174,200,400,776	/ 62 2
CHAR	174,202,002,074,002,002,402,374	/ 63 3
CHAR	030,050,110,210,776,010,010,010	/ 64 4
CHAR	776,400,400,774,002,002,402,374	/ 65 5
CHAR	374,402,400,574,602,402,402,374	/ 66 6
CHAR	776,404,010,174,040,100,100,100	/ 67 7
CHAR	170,204,204,170,204,402,402,374	/ 70 8
CHAR	374,402,402,406,372,002,402,374	/ 71 9
CHAR	000,060,060,000,000,060,060,000	/ 72 :
CHAR	000,060,060,000,060,060,020,040	/ 73 ;
CHAR	020,040,100,200,100,040,020,000	/ 74 <
CHAR	000,000,776,000,000,776,000,000	/ 75 =
CHAR	040,020,010,004,010,020,040,000	/ 76 >
CHAR	374,402,002,014,060,000,060,060	/ 77 ?

.END

.TITLE GRAPHICS DRAW X-LINE

/ 9-28-70

/ L H GERHARDSTEIN

/ CALLING SEQUENCE:

```

/      SET      GR.I
/      LAC      PJ
/      JMS*     GR.XL
/      LENGTH OF LINE (NO. DOTS)

```

```

GR.XL      .GLOBL      GR.XL
           Ø
           JMS*      GR.CKJ      / CHECK IF J WITHIN THIS S.L. BUFFER
           JMP      OUT      / ABOVE
           JMP      OUT      / BELOW
           LAC*     GR.XL      / WITH-IN LINE LENGTH
           CMA
           DAC
           JMP      CNT
LOOP      CHECK
           LAW      -1130      / -600(10)
           TAD*     GR.I
           SMA
           JMP      OUT      / LINE OFF PAGE
           LAC*     GR.I
           ISZ*     GR.I      / SET TO NEXT POINT
           JMS*     GR.SET
CHECK      ISZ      CNT
           JMP      LOOP
OUT        ISZ      GR.XL
           JMP*     GR.XL

```

.GLOBL GR.CKJ, GR.SET, GR.I

CNT Ø

.END

.TITLE GRAPHICS CHECK IF J IN S.L. BUFFER

/ 9-28-70

/ L H GERHARDSTEIN

/ CALLING SEQUENCE:

/	LAC	PJ	/ J POSN TO CHECK
/	JMS*	GR.CKJ	
/			/ ABOVE S.L. BUFFER AC = -DIST ABOVE
/			/ BELOW S.L. BUFFER AC = +DIST BELOW
/			/ WITH-IN S.L. BUFFER AC = POSN IN BUFFER

GR.CKJ	.GLOBL	GR.CKJ	
	Ø		
	TAD*	GR.PJN	/ -PAGE POSN
	SPAISTL		
	JMP*	GR.CKJ	/ ABOVE
	TAD	(-12	
	SMAISTL		
	JMP	.+4	
	TAD	(12	
	DAC*	GR.J	/ SET GR.J POSN
	ISZ	GR.CKJ	
	ISZ	GR.CKJ	
	JMP*	GR.CKJ	

.GLOBL GR.PJN, GR.J

.END

.TITLE GRAPHICS SET DOT

/ 9-28-70

/ L H GERHARDSTEIN

/ EXECUTION TIME = 62. TO 69 MICRO-SEC.

/ AVERAGE TIME = 65 MICRO-SEC.

/ CALLING SEQUENCE:

/ .GLOBL GR.SET
/ SET GR.J / 0<=J<=9
/ LAC I
/ JMS* GR.SET

GR.SET .GLOBL GR.SET
0
SPA / TEST IF I COORD LEGAL
JMP* GR.SET / NEGATIVE I ILLEGAL
TAD (-1130 / -600(10)
SMA
JMP* GR.SET / I > 599 ILLEGAL
TAD (1130 / RESTORE I IN AC
IDIV120000 / CLL!IDIV
22 / 18(10)
TAD (LRS / POSN IN WORD
DAC SHIFT
LACQ / QUOTIENT
DAC SAVE / NO. WORDS INTO L.B.
LAC GR.J / L.B. NUMBER (0-9)
MUL-16 / 4 BIT MUL (SHIFT LEFT 14 BITS)
46*20 / (2**4)*38(10)
/ A.M = (2**-18)*GR.J*(2**14)*(2**4)*38(10)
/ A.M = 38(10)*GR.J
TAD SAVE
TAD (3 / RELATIVE POSITION IN LINE BUFFERS
TAD* GR.LB / ADDRESS IN L.B. BLOCK
DAC INDEX
LAC (400000
SHIFT XX / SHIFT BIT INTO POSN
LMQ / SAVE IN MQ
LAC* INDEX / WORD FROM LB
OMQ / SET DOT
DAC* INDEX / RE-SET WORD FROM LB
JMP* GR.SET

 .GLOBL GR.LB
 .GLOBL GR.J, GR.I
GR.J 0
GR.I 0
INDEX 0
SAVE=INDEX

.END

.TITLE TEST GRAPHICS CHARACTER GENERATOR

/ L H GERHARDSTEIN
/ FEB 1, 1971

.DAT=135
.SCOM=100

	.GLOBL	CLA.
	.GLOBL	GR.IN, GR.CLR, GR.WR, GR.PJ
	.GLOBL	GR.TI, GR.TY, GR.ROT, GR.CHR
	.GLOBL	GR.CT0, GR.CHT
	.GLOBL	DDTSA.
BEGIN	LAC	CLA.
	DAC*	(.DAT+1
	LAC	GR.CT0
	DAC*	GR.CHT
	.INIT	1,1,0
	JMS*	GR.IN
	DZM	COUNT#
	DZM*	GR.TY
LOOP1	DZM*	GR.TI
	JMS*	GR.CLR
	LAC	COUNT
	DAC	CNT#
	LAW	-74
	DAC	CNTR#
	DZM*	GR.ROT
	LAC	CNT
	JMS*	GR.CHR
	ISZ	CNT
	IS7	CNTR
	JMP	.-4
	LAW	1
	JMS*	GR.WR
	ISZ*	GR.TY
	ISZ	COUNT
	LAC	COUNT
	SAD	(100
	JMP	BEGIN
	JMP	LOOP1
	.END	BEGIN

DISTRIBUTION

L. E. Addison	BNW	Math	R. J. Ausere	WADCO	324
R. A. Burnett	BNW	Math	B. E. Dozer	WADCO	703
D. B. Cearlock	BNW	PSB	R. A. Harvey (15)	WADCO	324
C. R. Cole	BNW	Math	J. A. Hubbard	WADCO	703
M. R. Compton	BNW	Math	S. A. Hunt	WADCO	324
V. L. Crow	BNW	Math	W. F. Lenzke	WADCO	324
D. W. Damschen	BNW	PSB	J. D. Lodge	WADCO	324
D. G. Daniels	BNW	PSB	W. R. Markillie	WADCO	324
P. J. Dionne	BNW	Math	W. H. Payzant	WADCO	324
G. E. Driver	BNW	Math	J. G. Sipe	WADCO	324
M. D. Erickson	BNW	Math	C. D. Swanson	WADCO	326
H. P. Foote	BNW	Math	L. D. Van Hees	WADCO	703
J. C. Fox	BNW	ROB	G. L. Waldkoetter	WADCO	703
D. R. Friedrichs	BNW	PSB	N. P. Wilburn	WADCO	324
L. H. Gerhardstein (5)	BNW	Math	G. W. Worth	WADCO	324
R. T. Jaske	BNW	PSB			
J. R. Kosorok	BNW	329			
G. W. Main	BNW	Math			
R. D. Mudd	BNW	PSB			
D. E. Olesen	BNW	PSB			
A. E. Reisenaur	BNW	PSB			
G. R. Taylor	BNW	Math			
C. J. Touhill	BNW	PSB			

AEC-DTIE (2)

Chicago AEC Patent Group (1)

R. K. Sharp (Richland-Chicago Patent) (1)

C. L. Robinson (1)