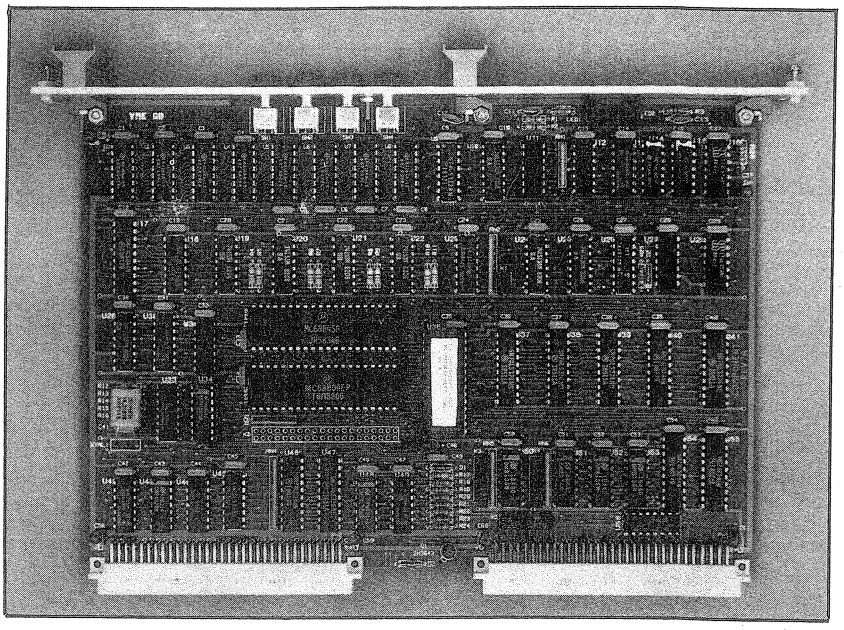
MicroGraphics, Inc.
P.O. Box 324
Cuyahoga Falls, Ohio 44221
216-923-3915
FAX (216) 923-6090

VME-GB-1

Intelligent Monochrome Graphics Controller



FEATURES:

* 512 x 512 or 480 x 512 resolution, software selectable
* 2 sets on A/N on board as well as scientific Greek Symbols
* downloadable character set supported (8 x 16 matrix)
* pies, arcs, and circles (ROM resident)
* on board CPU to minimize host burden and software overhead
* 12K Macro buffer to store up to 256 Macros on board
* user definable pattern for fills, or vectors
* Centronics type parallel port on P2 connector
* Firmware to allow 1:1 screen dump to Epson FX80/100
* LOW COST!!

DESCRIPTION:

The GB-1 is an economical, intelligent, monochrome graphics controller for the VME bus. The unit can most closely be compared to an intelligent X-Y recorder which draws on the screen rather than paper. An on-board 68B09 and extensive firmware allows the user to implement his/her application with a minimal amount of software development. Additionally, the Macro capability allows drawing and output routines to be stored and invoked using minimal commands from the host.

I/O:

The P-2 connector provides the output for a composite monitor and Centronics-type interface. The firmware is on board to allow a 1:1 screen dump to an Epson FX type printer. This output may be an entire screen or may be windowed.

The monitor output is composite video and may be configured to support many different scan rates.

FUNCTION:

Graphics functions are implemented by sending ASCII strings across the bus to the GB-1. The graphics functions may be by-passed and data sent directly to the Centronics port thereby offering the user a standard printer port for list functions.

Rev 8/87

SPECIFICATIONS: — VME-GB-1

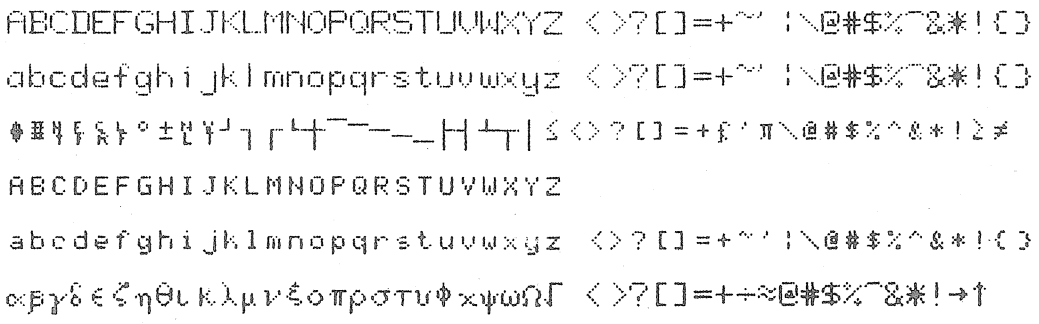
DESCRIPTION: Intelligent Monochrome Graphics Controller - 6U-VME

POWER — + 5V @ 1.2A
+ 12V @ .25A
CONFIGURATION.....A16 - D8 SLAVE
INTERRUPTS.....Any 1 of 7 (Jumper selected)
OUTPUTS (on P-2).....Composite Video
.....Parallel Centronics

TEMP.....0-55°C
HUMIDITY.....90% Non-Condensing

MICRO PROCESSOR — 68B09

ROM Resident Character Sets



FUNCTIONS — CONTROL CODES:

Basic Operation-
- $R - Activates graphics and formats incoming data
- $K - Clears screen and reverses drawing "color"
- $H - Holds all operations until CONTINUE button pushed
- $I - Timed wait like $H only for specific time (N x .1 sec.)
- $S - Deactivates graphics functions
- $N - Blanking of screen

Output to P2 connector-
- $TH - Dump to Epson (120 x 144 dots)
- $TL - Dump to Epson (60 x 72 dots)
- $TG - Dump to Epson (72 x 72 dots)
- $TV - Dump to Epson (same as DUMP button)

NOTE** All of the above may be windowed (i.e. only a portion may be dumped)
- $TT - Redefines the DUMP button (default is 72 x 72)
- $TC+A - Transfers A characters to printer
- $TD - Transfers 8 bit binary data to printer

Marco commands-
- $QD - Delete a specific macro
- $QE - Enter a macro
- $QX - Execute a macro
- $QZ - Clear all macros

NOTE** There may be up to 256 resident macros in the buffer as long as the total is less than 12K, space is dynamically allocated.

Character Drawing-
- $F - Format, size, type, vector and pattern specifier; modifies drawing modes, character types, sizes and area fill pattern
- $G - Graphic symbols (0-9 ROM resident)
- $C - Draw characters from ROM or down-load user character set
- $B - Same as $C but with block letters
- $Z - "Color" of "pen"

"Pen" motion-
- $O - Define origin
- $M - Move to a point relative to origin
- $L - Move to a point relative to present position
- $U - "Pen" up
- $D - "Pen" down
- $X - Auto-increment in the X axis
- $Y - Auto-increment in the Y axis

Drawing modes-
- $P - End points only
- $V - Vectors

Advanced functions-
- $AA - Generates arcs
- $AC - Generates circles
- $AP - Generates pies
- $AFB - Boundary seeking area fill
- $AL - Load bit pattern from host to graphics memory
- $AR - Read bit pattern from board to host

FRONT PANEL

Buttons-
- Reset- Reset CPU (same as power up)
- Continue - Clears $H or $I commands (waits)
- Dump - Dumps screen image to printer
- Clear - Clears screen to current color ($K)

Lights-
- Busy - lit when processing data
- Hold - lit when $H or $I received and in effect

NOTE ** Power up diagnostics are also implemented and any errors are indicated by these lights.

Specifications subject to change without notice

GRAPH BOX
Manual
Version 2.01
Revised 11/87

Table of Contents

I. INTRODUCTION- VMEGB1 MONOCHROME GRAPHICS CONTROLLER

The VMEGB1 monochrome graphics board consists of a 68B09 based command processor and a Centronix compatable parallel printer port for the VME bus.

The graphics command processor accepts simple ASCII character command sequences and performs drawing in 512 x 512 pixel display memory. The display output consists of a composite video signal and independent horizontal and vertical sync signals.

The parallel printer port supports screen dumps to any printer having the EPSON MX/FX series graphic commands. This port may also be used as an independent printer port.

Front panel switches allow the user to; reset the GB1 processor, initiate or terminate a screen dump, clear the display, and terminate a programmed hold . The graphics signals, printer port and front panel signal lines are all available at the P2 connector. The inclusion of the front the front panel signals provides a simple means of implementing a remote control panel for the graphics board.

Before using this board, four sets of jumper blocks must be configured for the particular system. In general, jumper blocks K1 and K5 select the interrupt request level of the module and jumper blocks K3 and K4 are used to select the base address of the graphics board. (A more complete explanation is found in section II)

For more detailed explanations and discussions of the capabilities of this product please refer to the appropriate section(s) as shown in the Table of Contents. If you need further assistance please call our offices. (216) 923-3915.

II. INTERFACING THE GB1

A. Hardware

Before using the GB1, 4 sets of jumpers must be configured for the particular application of the user. In general, jumper blocks K3 and K4 select the VME Bus address, and jumper blocks K1 and K5 configure the interrupt request and response of the board.

The P1 connector conforms to the standard VME spec (C.1). The P2 connector contains all the signals for the video monitor, the parallel port, and for an optional remote control panel for the module. See Below:

P2 Pinouts

Centronics parallel I/O		A	B	C		Centronics parallel I/O
1	strobe	1		1	GND	19
2	P0	2		2	GND	20
3	P1	3		3	GND	21
4	P2	4		4	GND	22
5	P3	5		5	GND	23
6	P4	6		6	GND	24
7	P5	7		7	GND	25
8	P6	8		8	GND	26
9	P7	9		9	GND	27
10		10		10	GND	28
11	busy	11		11	GND	29
12		12		12	GND	30
13		13		13		31
14		14		14		32
15		15		15		33
16		16		16		34
17	GND	17		17		35
18		18		18	GND	36
	GND	19		19	GND	
	GND	20		20	GND	
	video	21		21	GND	
	video	22		22	GND	
	video	23		23	GND	
	vert sync	24		24	GND	
	horiz sync	25		25	GND	
	/RST	26		26	GND	
	DUMP	27		27	GND	
	CLEAR	28		28	GND	
	CONT	29		29	GND	
		30		30		
	BUSY LED-	31		31	LED+	
	HOLD LED-	32		32	LED+	

Interrupt Request and Acknowledge

Jumper blocks K1 and K5 are used together to both request and acknowledge interrupt cycles on the VME bus. The GB1 board can request one interrupt. The graphics controller input register, output register, and printer register share this common interrupt request line. Jumper block K5 selects one of seven interrupt levels available on the VME bus. Jumper block K1 must be matched to the interrupt level selected by jumper K1.

JUMPER BLOCK K1

1	x x	2
3	x x	4
5	x x	6

JUMPER BLOCK K5
Interrupt level
(only one must be inserted)

level 1	1 x x	2
level 2	3 x x	4
level 3	5 x x	6
level 4	7 x x	8
level 5	9 x x	10
level 6	11 x x	12
level 7	13 x x	14

JUMPER BLOCK K1
Response configuration
(J= inserted jumper)

1 x x 2	3 x x 4	5 x x 6
J	J	
J		J
J		
	J	J
	J	
		J

Base address selection JUMPER BLOCKS K3 and K4

Since the GB1 is an I/O board it will respond to addresses that use modifier codes 29H and 2DH. By responding to these codes the GB1 only compares address lines A1 through A15. Board selection is accomplished by matching A4 through A15 with jumper selected levels. (jumper= LOW no jumper= HIGH)

K3

1 x x 2	A8	FE[illegible]
3 x x 4	A11	F7[illegible]
5 x x 6	A9	FD[illegible]
7 x x 8	A4	FE[illegible]
9 x x 10	A10	FB[illegible]
11 x x 12	A12	EF[illegible]

K4

1 x x 2	A6	FF[illegible]
3 x x 4	A14	BF[illegible]
5 x x 6	A13	DF[illegible]
7 x x 8	A7	FF[illegible]
9 x x 10	A5	FF[illegible]
11 x x 12	A15	7F[illegible]

All odd numbered pins are ground. To select an address it is necessary to install a jumper to match a low level on an address line and remove a jumper to match a high level on an address line.

GB1 Programming--

This section provides some of the basic information needed to program the GB1. For more information on the graphics command protocol refer to Sections III-VII of this manual.

GB1 Addresses --

The GB1 contains 5 data/control registers which are user programmable to affect data transfer to and from the GB1 module. These registers reside at the indicated base addresses.

HHH(1) = Write Data Register, WDR (write only)
8 Bit data to GB1, all commands are sent to the controller via this register

HHH(3) = Read Data Register, RDR (read only)
8-Bit data from GB1, pixel information may be returned through this register

HHH(5) = Printer Data Register, PDR (write only)
8-Bit data to parallel printer

HHH(7) = Control/Status Register, CSR (read or write)
ready status and interrupt enable register

HHH(9) = Interrupt Vector Register, IVR (write only)
holds programmable interrupt vector address

All of the above are byte addresses and must be addressed on an odd boundary.

Write Data Register, WDR--

This 8-bit write only register is the input data register to the graphics controller. All graphics commands and data pass through this register. The WDR ready bit (WDRDY, bit 7) of the Control/Status Register (CSR) will be at a '1' level when the graphics controller is ready to accept another input character. An internal 4K buffer for graphics commands allows fast loading of graphics sequences without waiting for the completion of the drawing. The WDRDY bit of the CSR register will be at a '0' level while placing the data in the internal buffer or if the buffer is full. Data written to this register with the WDRDY bit at the '0' level will be ignored. The WDR interrrupt enable bit (WDIE, bit 6) of the (CSR) may be set to a ;1' level to cause an interrupt when the WDR bit is at the '1' level (i.e. requesting more data).

Read Data Register, RDR

This 8-bit read only data register is the output register for the graphics controller. All graphics commands which require a response from the controller board will cause data to be transferred to the register.

The RDR Ready bit (RDRDY bit 5) of the (CSR) register will be at a '1' level when data is ready for the VME bus host.

The RDR interrupt enable bit (RDIE, bit 4) of the CSR may be set as a '1' level to cause an interrupt when the RDR bit goes to the '1' level.

Printer Data Register--

This 8-bit write only register is the input data register to the parallel printer port. PDR ready bit (PRDY, bit 3) of the CSR will be at a '1' level when the graphics controller is ready to accept another input character for the printer. An internal 12K byte buffer for the printer allows fast dumping of printer data without waiting for the printer to complete print task. The PRDY bit of the CSR will be at a '0' level while placing the data in the internal buffer or if the buffer is full. Data written to this register with the PDRDY bit at '0' will be ignored. The PDR interrupt enable bit (PDIE, bit 2) of the CSR may be set to a '1' level to cause an interrupt when the PDR bit is at the '1' level. (i.e. requesting data)

Control/Status Register, CSR

The Control/Status Register contains the ready and interrupt enable bits for the WDR, RDR, and PDR registers.

As follows:

bit 7 - WDRDY, Write Data Register ready bit '1' when new data may be written

bit 6 - WDIE, Write Data Register Interrupt Enable bit '1' enables an interrupt when WDRDY is a '1'

bit 5 - RDRDY, Read Data Register read bit '1' when data is available from GB1

bit 4 - RDIE, Read Data Register Interrupt Enable bit '1' enables an interrupt when RDRDY is a '1'

bit 3 - PDRDY, Printer Data Register ready bit '1' when data may be written to PDR

bit 2 - PDIE, Printer Data Register Interrupt Enable bit '1' enables an interrupt when PDRDY is a '1'

A note on interrupt processing:

The GB1 board uses a common interruptvector for interrupt processing. If more than one source is responsible for the interrupt and the second interrupt source is not processed, the board will be unable to process any further interrupts. Therefore, during interrupt processing by the host processor, each of the 3 possible interrupt sources must be checked and processed or the following processor code will be required to enable further interrupt processsing without regard to the multiple source requests:

a. move CSR data to stack
b. clear CSR register
c. restore CSR data from stack

This sequence reclocks the interrupt control logic on the GB1

Interrupt Vector Address Register--

This 8 bit write only register supplies the common interrupt vector address for interrupts generated by the GB1

B. Software

In appendix G. the user will find the source listing for two drivers written for PDOS. One sends characters to the GB1 for graphics functions and the other directly accesses the Centronics port available from the P2 connector. By using the Copy File command a user may easily list his/her program to the printer.

e.g. CF,(program),VGBP

One method to implement the GB1 from a language such as Basic, is to use the Spool Command . This will send the output from any "PRINT" statements directly to the GB1.

```
e.g.  10  SPOOL 16 TO "VGBG"      ' send print output to GB
                                    driver "VGBG"
```

The optimal implementation of driver and software is unique to each application; therefore this discussion is meant to serve as a starting point only.

III. OPERATING THE GB1

A. COMMAND SUMMARY

The Graph Box is controlled by the following commands. More complete summaries with format and uses are in the main body and Appendix B of this manual.

A. Basic control operation

$R- Activates the GB
$K- Clears screen and reverses 'color'
$H- Holds operation until CONT button pushed
$I- Timed wait (N x .1 sec)
$E- Xon/off and display setup
$N- Blanks screen
$S- Turns off Graph Box graphics

B. Motion of the "pen"

$O- Defines origin
$M- Moves to point relative to origin
$L- Moves to point relative to present position
$U- "Pen "up
$D- "Pen" down
$X- Auto increment X value
$Y- Auto increment Y value

C. Drawing modes

$P- Points (endpoints only)
$V- Vectors (lines)

D. Character Drawing

$F- format, size, type, vector & pattern specifier; modifies drawing modes, character types, sizes, and area/vector fill patterns
$G- Graphic character (symbols)
$C- Draws characters from ROM or down-loaded user-defined character set
$B- Same as $C but in block letters
$Z- "Color" of "pen" (black or white)

E. Output Controls

$TH-Dumps to printer (120 x 144 dots scale)
$TL-Dumps to printer (60 x 72 dots scale)
$TG-Dumps to printer (72 x 72 dots 1:1)(Default)
$TV-Dumps to printer (same as DUMP button)
$TT..-Redefines DUMP button
$TC+A- Transfers A characters to printer
$TD - Transfers 8 bit binary data to printer

F. Advanced Functions

$AA-Generates arcs
$AC-Generates circles
$AP-Generates pies
$AFB- Boundary seeking area fill (pattern definable)
$AL-Loads GB from host/term, pixel by pixel, or by bits
$AR-Reads GB to host/term, pixel by pixel, or by bits
$AFR- Rectangular area fill
$KR- Clears a rectangular area

G. Macros

$QD - Deletes specific macro
$QE - Enters a macro
$QX - Executes a macro
$QR - Restores saved drawing parameters
$QS - Saves drawing parameters
$QX - Executes a Macro
$QZ - Clears the Macro buffer

NOTE: ALL DISCUSSIONS HEREAFTER presume that the PDOS drivers are installed and the system is running in Basic. Any other language which passes ASCII characters should work similarly.

B. BASIC OPERATION

In order to implement graphics on the GB1 the unit must first receive the character $Rx. Where x is an optional modifier (1-9). The presence of the modifier indicates the GB is in the integer format. A 0 or nothing indicates free format. These are defined in the following paragraph. Default value is 0 indicating free format. Once specified, this format remains in effect until changed by reissuing the $R command with a different value.

INTEGER FORMAT ($R1-$R9)

Integer format expects X&Y co-ordinate data to be in a fixed format of 1-9 digits/co-ordinate. Spaces are counted. When in the integer format ($R1-$R9), if the first character is a - or a digit, data is assumed to follow in the specified integer format until a <CR>, <LF> or other control character (for GB1) is encountered. This does not apply to $R,$F,$C or $TC.

If the first character following the command is NOT one of the above, the GB1 enters the search mode. The GB1 searches for the character +.It then assumes all data immediately following to be in the proscribed format. Upon encountering a <CR>,<LF> or other control character it stops taking data and again enters the search mode.

FREE FORMAT ($R OR $R0)

In the free format mode, data may be taken with a varying number of digits. Nothing is recognized until a +,-, or (space) is encountered. Data acceptance is terminated by a <CR>,<LF> or ASCII control character. At which time, the GB resumes the search mode until a + is again encountered. Data collection then resumes in the free format mode.

A new graphics command is always recognized in the search mode. All commands may be either upper or lower case letters.

C. SOME OTHER PRELIMINARY COMMANDS

The $K command clears the monitor screen and reverses the background. i.e. white on black or black on white. The $H holds the program execution until the continue (CONT) button on the GB1 is pushed . The $I command executes a timed hold. It may be terminated by pressing the CONT button. This allows inspection of the graphics or possibly even a dump without interaction with the host. Lastly, the $S command turns off the GB1. No further graphics may be drawn until a $Rn is given.

Operation of the Graph Box may now begin. The GB1 uses 512 x 512 points addressing to display a 512 x 512 or 512 x 480 video plot on the monitor. The absolute origin is in the lower left hand corner. This may be relocated to anywhere on the screen with the $O command.

EXAMPLE: 1.

```
10 PRINT "$R4$002560256
```

Explanation: The $R4 specifies that data will be in the integer format of 4 character blocks. The origin now is in the middle of the screen with the lower left quadrant (-X,-Y), lower right (X,-Y), upper right(X,Y), and upper left(-X,Y).

The next command to discuss is the $M or move command. It is used to move the "pen" to a co-ordinate location. The $U and/ or $D may also be used in conjunction with this either to "drag the pen" as it moves or lift it and leave no "trail". As with the previous commands, the co-ordinates are expected to be in the format specified by the most recent $R command.

The drawing functions $P point, and $V vector will be illustrated in this example.

EXAMPLE: 2

a)
```
10 PRINT "$RJ$0256256"
20 PRINT "$U$P$M000000$D"
```

This should put a single point in the middle of the screen. Try other numbers following the $M to put points elsewhere.

b)
```
10 PRINT "$R4$K$002560256"
20 PRINT "$U$P$M00000000$D"
30 PRINT "+  00-125"
40 PRINT "+00000125"
```

Discussion

Several items are illustrated here. Note that since the pen is in the $P point mode (line 20) and has been activated or put down ($D), proper data following will be the co-ordinates of points on the screen. As soon as new data is received (lines 30 and 40), a point is displayed. Proper data is that which satisfies the format specified by the $R command most recently implemented. Also note that in the integer format, the spaces and minus, (line 30) are counted characters. Per the requirements, since a control character ($D) was printed in line 20, the GB goes to the search mode in line 30 to find a '+' which indicates more data. The same principles apply to the $V vector mode.

EXAMPLE: 3

```
10 PRINT "$R$Z0$K$O 0 0"
20 PRINT "$U$M 256 256"
25 PRINT "$F$V$D"
30 PRINT "$M 256 319 188 319 188 256 256 256"
40 PRINT "$M 256 188 256 188 319 256 256 256"
```

Here we see the same rules mentioned in the previous discussion. Make note of line 20. If the $U had not been given, the line would have drawn from (0,0) to (256, 256).

In summary of the preceding section, a few rules must be reiterated:

1. If the "pen" has been raised ($U) it must be lowered ($D) before anything can be written.

2. If the vector mode is given and the "pen" has not been raised, it remains "down" until raised. The $D command need not be used each execution.

3. If the point mode is invoked and the "pen" has not been lowered, it is necessary to lower the pen to write.

4. Once the mode ($V or $P) has been set, and the pen lowered, the $M will draw the points or vectors on the screen until a new control character is given.

The following program puts points on the screen and then connects them with vectors, illustrating the commands thus far explained.

EXAMPLE: 4

```
10 PRINT "$R$O 0 0 $D$P"
20 PRINT "$M 100 100 150 100 175 100 200 150"
30 PRINT "$V$U$M 0 0 $D"
40 PRINT "$M 100 100 150 100 175 100 200 150"
```

Note that with "pen" down (lines 10-20) point mode plots only endpoints. Upon changing to vector mode (line 30) we lifted the "pen" to return to origin without drawing from (200,150). We then lowered the "pen" so we could plot vectors in line 40.

C. ENHANCEMENTS

The $L, relative move, is a very powerful command. It may be used with the $P, or $V . It moves the "pen" up or down, or right or left to a position related to the last position. (+ equals up in the Y axis, - equals down ; + equals right in X axis, and - equals left)

For example:

```
PRINT "$V$L 000 100"
```

This moves the "pen" in the vector mode from the last position to the same X position and 100 pixels "UP" on the Y axis. This can be used with the $M command to put a shape anywhere on the screen.

An example program follows.

EXAMPLE: 5

```
10 PRINT "$R$Z0$K$0 256 256"
20 PRINT "$U$M 60 60"
30 GOSUB100
40 GOSUB110
50 PRINT "$U$M -220 0"
60 GOSUB100
70 PRINT $V :GOSUB110
100 PRINT "$U$L 70 70"
110 PRINT "$D$L 0 -140"
120 RETURN
```

DISCUSSION:

Line 10 sets the GB up to accept data in the free format and clears the screen to black. It also sets the origin in the middle of the screen (256,256). Lines 20-30 lift the "pen" and move it to the starting postion on the screen (60, 60). The subroutine (lines 100-120) first lifts the "pen", enters vector and relative mode and moves to the right 70 points and up 70 points (70, 70). (This centers the figure about the drawing location.) The "pen" is then lowered and the vector is drawn from the initial point to the same X but lower (-140) Y location.

Thus, with a minimum of software, one may create a library of subroutines for shapes which may be drawn anywhere on the screen. See also Section VII, on Macro Buffer Commands.

The $Z_ command is used to control the "color" of the writing on the screen. (Free format $Z0=black, $Z1=white.) This has two useful applications. One, it may be used with the $K to clear the screen to a desired color (not the reverse only). Two, it will "erase" lines presently on the monitor. The following examples will illustrate.

EXAMPLE: 6

```
    10 PRINT "$R"                 'TURNS ON GB IN FREE FORMAT
    15 PRINT "$O 0 0              'SETS ORIGIN at (0,0)
    20 PRINT "$Z0$K$Z1            'CLEARS SCREEN TO BLK
    30 PRINT "$M 100 256          'MOVES TO CO-ORDS
    40 PRINT "$F55 $C+CLEAR       'WRITES "CLEAR" IN WHITE
    50 PRINT "$I20"               'HOLDS FOR 2 SECS
    60 PRINT "$M 100 256          'MOVES TO OLD CO-ORDS
    70 PRINT "$Z0$C+CLEAR         'WRITES BLK ON CHARAC (ERASES)
```

DISCUSSION:

As in previous examples, the first lines set up the GB. The comments beside each line should clarify what each command does.

The next command is the **$X $Y** incremental mode. The first value following the command is the fixed increment. Subsequent data are the X or Y values relative to origin ($0). Remember that if $0 is not specified, it defaults to (0,0).

EXAMPLE: 7

```
10 PRINT "$R$V$Z0$K"
20 PRINT "$0 25 25 $Z1"
30 PRINT "$U$M 0 0 $F"
40 PRINT "$D$X 13"
50 PRINT "+ 16 56 111 98 42 125 98 13"
60 REM  NOW THE Y AXIS
70 PRINT "$U$M 250 125"
80 PRINT "$D$Y 13"
90 PRINT "+ 275 125 150 188 32 125 125 125 213"
```

DISCUSSION:

Line 10 enables the GB in Free Format, sets it to Vector mode, and clears the screen to black. Line 20 sets origin at (25,25) and prepares to draw in white. The next line lifts the 'pen', moves it to the new origin (25,25), and announces a format for data following. Lines 40-50 drop the 'pen', set incremental mode on X axis, name a 13 pixel increment, and feeds Y values for graphing from the origin. The graph proceeds automatically along the X axis. Lines 70 - 90 do the same along the Y axis.

D. CHARACTERS AND SYMBOLS

The Graph Box has two full A/N character tables and a third programmable RAM set as well as scientific Greek characters. These may be plotted anywhere on the screen. The group of commands which control this function and its permutations are as follows:

$C+ - Print following characters in specified type ($F command)

$B+ - Print following characters in specified type in Block Mode

These may be modified by the preceding $F command and one or more of the following additional modifiers.

H - Writes in horizontal (DEFAULT)

V - Writes along Y axis

L - Lower case English type

G - Scientific Greek (only with lower case input)

/ - All zeroes slashed (DEFAULT)

\ - All zeroes non-slashed

B - Uses "block size" characters

P - Uses "normal size" characters (DEFAULT)

C - Writes in complement of field (ERASES)

R - Replaces bit plane with line drawing bit pattern specified by binary equivalent of decimal number.

The digits 0-9 may also be used immediately after the command to specify the height and width of the characters. The first digit is the height and the second the width. The character set is chosen with the $CS(select)A,B,or C command. The chosen set will be used until changed by the same format. Further implementations of the character mode do not require the S or A,B,or C.

EXAMPLES:

$FV83	produces tall thin characters written vertically.
$FHLB55	produces normal characters written horizontally, in block size lower case English.
$CSB+	Selects the "B" character set until changed
$C +	Will draw from "B" character set since previous line Selected "B"

EXAMPLE: BLOCK SIZES AND BLOCK LETTERS

```
10 PRINT "$R$Z0$K"
20 PRINT "$O 0 0 "
30 PRINT "$M 256 256"
40 PRINT "$F36$C+TEST"
45 PRINT "$I20"
50 PRINT "$F36$B+TEST"
```

EXAMPLE:
A)

```
10 PRINT "$R$Z0$K"
20 PRINT "$O 0 0 "
30 PRINT "$M 256 256"
40 PRINT "$FV55$C+TEST"
50 PRINT "$I20"
60 PRINT "$B+TEST"
```

B)

```
70 PRINT "$M 64 64"
80 PRINT "$FH $B+MORE"
90 PRINT "$FG $B+sldkeoir"
100 PRINT "$M 112 64 $C+sldkeoir"
110 PRINT "$M 64 64 $FC99$B+TEST"
```

There are also 10 graphics markers which are affected by the $F command.

$G0 - Diamond	**$G5** - I
$G1 - +	**$G6** - Right Pointing Triangle
$G2 - Square	**$G7** - Left Pointing Triangle
$G3 - Tilted Square	**$G8** - Down Pointing Triangle
$G4 - Octagon	**$G9** - Up Pointing Triangle

The $G_ does not change current plotting mode during execution. However, $G_ always plots in Vector mode and always lifts the 'pen'. Drawing these symbols will not advance the cursor as writing text would.

EXAMPLE: 8

```
120 PRINT "$Z0$K"
130 PRINT "$M 256 256"
140 PRINT "$F99$G0"
150 PRINT "$F39$G0"
160 PRINT "$F33$G0"
```

The $F command has other functions which will be discussed in Section IV :Advanced Drawing Functions.

Character fonts or symbols may be downloaded to be substituted for a standard ROM character. When this is done and the "A" option is selected when character plotting ($CSA or $BSA), the downloaded pattern will be drawn whenever the previous character would normally be shown. Commands follow:

$CL+ - Load point character RAM with new character.

$BL+ - Load point character RAM (same as $CL).

EXAMPLE: $CL+M+N0+N1+N2...N15

Where M = Decimal value of ASCII character (0-127), by which new character can be recalled.
N0-N15 = Decimal value of row from new character plot.

	128/64	/32	/16	/ 8	/ 4	/ 2	/ 1		
N0	X					X		N0=	130
N1	X			X		X		N1=	146
N2	X		X		X	X		N2=	170
N3		X			X			N3=	68
N4		X			X			N4=	68
N5			X		X			N5=	40
N6				X				N6=	16
N7				X				N7=	16
N8				X				N8=	16
N9				X				N9=	16
N10			X		X			N10=	40
N11		X			X			N11=	68
N12	X					X		N12=	130
N13	X					X		N13=	130
N14	X					X		N14=	130
N15	X					X		N15=	130

The command would be:
"$CL+65+130+146+170+68+68+40+16+16+16+16+40+68+130+130+130+130"

This will create a pattern callable by "**$CSA+A**".
This character will be called when the letter "A" is typed and the "A" character set has been selected.

Character plotting summary of commands

$CSA+ - Selects character set A (programmed set).

$CSB+ - Selects character set B (from ROM).

$CSC+ - Selects character set C (from ROM).

$CSR+ - Restores last-used ROM character set, dropping out of programmed set.

$BSA+ - Selects character set A (or B or C) to print in block mode.

$BSR+ - Restores last-used ROM character set to print in block mode.

$CSA+TEST - Prints out "TEST" in your programmed character set.

$CSB+TEST - Prints "TEST" in ROM character set B.

$CSC+TEST - Prints "TEST" in ROM character set C.

$CSR+TEST - Restores most recently used ROM character set, dropping out of programmed set to print "TEST". Use to enter text after setting blueprint symbols, for instance.

The $C or $B Command initiates a character drawing sequence:

1. The Pen is lifted.
2. Search mode starts at '+' ($CSA+...).
3. All characters after the '+' are drawn.
4. Any control character <CR>,<LF> terminates the $C command.
5. The pen is lifted at completion of $C command.

IV. ADVANCED DRAWING FUNCTIONS

Four new commands implement the advanced drawing functions.

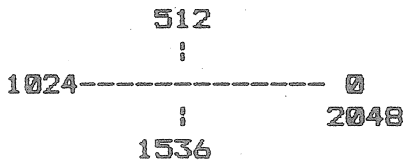
$AA -Draws an arc centered at current position
E.G. $AA +A +B +C

$AC -Draws a circle centered at current position
E.G. $AC +A

$AP -Draws a pie centered at current position
E.G. $AP +A +B +C

Where : A = Arc/Pie/Circle Radius
B = Beginning Angle
C = Ending Angle

Angle is in units of 2048/360 degrees. Angle starts at 3 o'clock (0 degrees) and continues counter-clockwise. E.G. 12 o'clock = 512, 9 o'clock = 1024, etc.



30 -- 171
45 -- 256
60 -- 375
90 -- 512

$AFB - AREA FILL TO BOUNDARY

All of the preceding commands are modified by $F format strings. The $F will determine the character size and attributes and specify the vector pattern and fill pattern.

The first number (decimal value) in the format specified by the previous $R command is the 16 bit pattern of the number that follows. It determines the vector pattern. The LSB is plotted first.

The area fill pattern is defined by the next 16 16- bit words (in decimal format). They are drawn LSB to MSB proceeding down the screen.

Thus it is possible to make any line or area fill pattern you would need. A low resolution (16 X 16) company logo could give graphics a mark of distinction and aid in presentations.

EXAMPLE: 10

```
10 PRINT"$R$Z0$K"
20 PRINT"$O 0 0"
30 PRINT"$M 256 256"
35 PRINT"$F"              ' DECLARE A FORMAT FOR VECTORS
40 PRINT"$AC 32"          ' DRAW CIRCLE 32 PIXEL RADIUS
50 PRINT"$F 326"          ' NOTE bit pattern of 326 binary
60 PRINT"$AP 25 1024 512"   ' MAKE PIE WITH PATTERN VECT
65 PRINT"$AC 64"            ' MAKE A CIRCLE TOO
70 PRINT"$F  -1 326 327 327 -1 -1 -1 256 256 256 -1  -1  -1
  +1024 1024 1024 -1 "      ' DEFINE THE AREA PATTERN
80 PRINT"$I20"              'WAIT 2 SECS
90 PRINT"$M 250 250 $AFB"   ' MOVE TO FILL (SEE BELOW)
100 PRINT"$M 125 125"       ' MOVE
110 PRINT"$AA 32 1024 1500"' DRAW AN ARC
110 PRINT"$AC 128 365 50"   ' MAKE MORE CIRCLES WITH ONLY RAD
```

DISCUSSION:

Line 40 draws a circle of 32 radius in the middle of the screen. At line 50 a new format for vectors is specified and will be used to draw a pie within the circle in line 60. THIS FORMAT FOR VECTORS WILL REMAIN UNTIL CHANGED. IT WILL ALSO AFFECT PRINTED LETTERS! In line 70 two items are changed. The first value following the $F which had specified the previous vector pattern is now a -1 (which gives solid vectors). The subsequent values represent the bit patterns which will be used for the area fill executed in line 90 ($AFB). The 'pen' was moved in 90 because THE AREA FILL WILL THINK IT HAS FILLED IF IT IS SITTING ON A VECTOR OF THE FILL COLOR!

In line 110 an arc is drawn at location 125 125 (line 90). The last line illustrates how the Graph Box continues to process data until another command is given. That is, a series of radii following the $AC command produces a series of circles without the need to re-issue the $AC command.

There are four other commands in this group.

1. $AL+ N - Load display memory by transferring a character (0-1) per pixel and color (0-1) of the screen for N pixels.

2. $ALB+ N - Load display memory by byte (8 pixels per byte) beginning at current position (Modulus 8). Data follows as one 8 bit character per 8 pixels.

3. $AR+ N - Read display memory, controller sends a character per pixel on screen for N pixels, converts color code (0-1) to ASCII codes (0-1) and transmits to host.

4. $ARB+ N - Read display memory, controller sends N bytes (8 pixels per byte) beginning at current position (Modulus 8) and transmits to host.

* NOTE '0' = pixel off, '1' = pixel on. Reading and loading always proceed from left to right, top to bottom. That is, with increasing X and decreasing Y.

The above commands allow the host either to read or write into the GB1 memory. This can be time consuming if an entire screen is to be read, however, by using the Byte mode ($ARB or $ALB) or transferring only a portion, the process may be greatly speeded.

V. MACRO BUFFER COMMANDS

The greatest power of the GRAPH BOX is its **Macro Buffer**. Using the 12K of storage for up to 256 macros, a GRAPH BOX operator can create his own library of logos, drawing sequences, graphing formats.

These commands control the Macro Buffer:

$QE+N+ - Enter data for macro N (1-256)

$QD+N - Delete macro N from buffer

$QR - Restore saved ($QS) drawing parameters

$QS - Save current drawing parameters

$QX+N - Execute macro N

$QZ - Zero the macro buffer (erase all macros)

$Q - Ends data entry to macro (specifying another $Q command will also end input)

Where N = 1-256 (name of macro)

Macros may contain any sequence of commands or text except $Q, including such control characters as <CR>, <LF>, and non-printing characters as well. The 12K buffer will hold up to 256 macros, numbered 1-256. Space is dynamically allocated.
THE NUMBER OF THE MACRO (N) MUST BE FOLLOWED WITH A (+)!!!

NOTE: THERE IS NO INDICATION WHEN THE LIBRARY IS FULL, ALWAYS TEST FOR PROPER EXECUTION OF MACRO

EXAMPLE:

```
10 PRINT  "$R$Z1$K$QE+1+";          'SETS GRAPH BOX, CLEARS
                                     SCREEN, PREPARES TO ENTER
                                     MACRO 1 DATA
20 PRINT "$0 25 256 $F55$V$C+HELLO" 'STORED IN MACRO 1
30 PRINT "$M 0 0 $FC$C+HELLO"       'STORED IN MACRO 1
40 PRINT "$Q"                       'ENDS DATA INPUT TO MACRO 1
50 PRINT "$QX+1"                    'EXECUTES MACRO 1
```

By saving the commands to disk, then feeding them in as MACRO entries, a library of subroutines may be developed, stored and shared.

VI. THE PRINTER PORT

A. THE HARDCOPY COMMANDS

The GB1 is supplied with a parallel printer port which has the resident firmware to drive an Epson FX 80/100 printer.

****ONLY THE FX SERIES CAN EXECUTE A 1 TO 1 SCREEN DUMP. ANY OTHER UNIT MAY CAUSE GRAPHICS DISTORTION!!!**

Printing may be initiated from either the software or the dump switch on the board front. This section will discuss the commands which deal with the printer and the GB1 firmware.

The printer commands are:

```
$TL+A+B+C+D+E+F  -    60 X 72 dots/inch (low density)
$TG+A+B+C+D+E+F  -    72 x 72 dots/inch (graphics mode)
$TH+A+B+C+D+E+F  -    120 x 144 dots/inch (high density)

$TC+A+<characters>-Transfers 'A' characters to printer

$TD+A...(data)   - Transfer 8 bit binary equivalent of
                   decimal data to printer

$TV              - Dumps screen in 72 x 72 format (Default
                   value of Dump button on GB)

$TT..            - Reassigns Dump button to:

        $TTL     -   60 x 72 dots/ inch (low density)
        $TTG     -   72 x 72 dots/ inch (graphics mode)
        $TTH     -  120 x 144 dots/ inch (high density)
```

NOTE:

A = Printing zoom factor (1-16)
B = Epson Columns to skip (PIXELS ON SCREEN)
C = First X Coordinate
D = Last X Coordinate
E = First Y Coordinate
F = Last Y Coordinate

Observe that with C,D,E,F the screen may be windowed.

THE DATA (A,B,C,D,E,F) MUST BE IN PROPER $R FORMAT (FREE OR SPECIFIED NUMBER OF INTEGERS)

ANY SCREEN DUMP WILL TERMINATE WHEN 'DUMP' BUTTON IS PUSHED

B. THE EPSON INTERFACE

HARDWARE DUMP

The GBI's parallel port is designed for the Epson FX80/100. The MX80 series will also provide screen dumps but due to their lower resolution, the 1 to 1 dump will distort. (The printer must have 72 x 72 dots/inch.) Simply plug in the printer and interface cables to install. Now push DUMP to generate hardcopy (72 x 72 dots/inch) of the graphics on the screen.

Although 72 x 72 resolution is the default value, the **$TT..** command may reassign the DUMP button default to either of the other resolutions.

SOFTWARE DUMP

The graphics dump may also be initiated by the software. There are three modes.

1. $TL 60 x 72 dots
2. $TG 72 x 72 dots
3. $TH 120 x 144 dots
4. $TV 72 x 72 dots (NO FURTHER COMMANDS)

These must be followed with the following values in the current data format, ($R0-9) EXCEPT $TV.

- A - The printing zoom factor (1-16)
- B - The number of Epson columns to skip before printing
- C - the first X coordinate to be printed
- D - the last X coordinate to be printed
- E - the first Y coordinate to be printed
- F - the last Y coordinate to be printed

The form is : $TH+A+B+C+D+E.
The values A-E must be in the most recently specified format ($R0-9).

This allows the user to "window" the output.

PRINT "$TG+2+100+100+200+100+200"

Initiates a graphics mode screen dump, showing only those pixels windowed between (100- 1st X, 200-last X;100-1st Y, 200-last Y).
It also offsets the print position by 100 pixels (approx 1/5 of page, 100/512) and zooms the output by 2.

Two other commands allow the user direct access to the printer. The **$TC** command will transfer characters directly to the printer. This can be used in labeling a chart even before you begin to run the graphics. the form is:

$TC+A+<characters>

Where A= the number of characters to be printed.

10 PRINT "$TC+5+ABCDEF"
would produce a printout of ABCDE since 5 characters were expected.

The other command **$TD** will transfer the 8 bit binary equivalent of decimal data to the printer even if it is turned off. This can be used to transmit control commands or characters which may cause difficulty with the terminal (escape sequences).

The form is:

10 PRINT "$TD+A...(DATA)"

Please also be aware that the parallel port may be used as a standard Centronics compatable output. By using the CF command from PDOS (Eyring Research), a file may be listed to a printer.

CF FILE,VMEBP (VMEBP is driver listed in APPENDIX G.

VII. MISCELLANEOUS COMMANDS

**** PLEASE NOTE SOME SYSTEMS DO NOT REQUIRE XON/XOFF PROTOCOL. ON THESE SYSTEMS SOME COMMANDS MAY EFFECT THE COMMUNICATIONS PORTS. TO ELIMINATE THE PROBLEM DISABLE THE XON/XOFF PROTOCOL ($E5)**

ALL OPTIONS ARE ENABLED UPON POWER UP OR RESET. TRY THE UNIT OUT BEFORE RESETTING ANY OPTIONS.

The hardware and firmware controlling the P2 are addressed by the E command. The optional parameters are as follows:

$E0,$E1,$E2,$E3,$E6,$E7	reserved codes
$E4	enable "link" XON/XOFF DEFAULT
$E5	disable "link" XON/XOFF
$E8	next decimal value is ASCII new "XON" character
$E9	next decimal value is ASCII new "XOFF" character
$EA	512 x 512 displayed DEFAULT
$EB	512 x 480 displayed (used for video tape or some monitors which cannot handle the resolution)
$EC	Clipping of offscreen images (image going off side of screen disappears)
$EW	Wraparound images (image going off one side of screen reappears on opposite side)

The form of the command is :

$En where n=0-9

NOTE: Options 8 and 9 require the decimal value in the format ($Rn) most recently used. Also note that upon power up or reset all options are enabled.

EXAMPLE:

```
10 PRINT "$R4$E8+0027"
```

This resets the XON character to ASCII character 27 (esc). We must use "0027" because the format was $R4.

Two other useful peripheral commands are:

$NB - Selects blank CRT screen for display
$NU - Selects unblank CRT for display

We hope this manual has helped acquaint you with the GRAPH BOX. If you find any inaccuracies or errors please advise us at the following address.

MICROGRAPHICS INC.
PO BOX 324
CUYAHOGA FALLS, OH 44221

APPENDIX A. Pin outs and connectors

These are the connections for the MGI cable if ordered.

Epson printer connector (AMPHENOL 57-40360)-

1-/STROBE	19-GROUND
2-DATA 1	20-GROUND
3-DATA 2	21-GROUND
4-DATA 3	22-GROUND
5-DATA 4	23-GROUND
6-DATA 5	24-GROUND
7-DATA 6	25-GROUND
8-DATA 7	26-GROUND
9-DATA 8	27-GROUND
10-/ACKNLG	28-GROUND
11-BUSY	29-GROUND
12-PE	30-GROUND
13-SLCT	31-/INIT
14-/AUTO FEED EXT	32-/ERROR
15-	33-
16-GROUND	34-
17-	35-
18-	36-GROUND

'Video' Connector (DB-15P)-

1 -		VIDEO
	9-	GROUND
2 -		
	10-	
3 -		VIDEO
	11-	GROUND
4 -		
	12-	
5 -		VIDEO
	13-	GROUND
6 -		
	14-	GROUND
7 -		VERTICAL SYNC
	15-	GROUND
8 -		HORIZONTAL SYNC

P2 Pinouts

Centronics parallel I/O		A	B	C		Centronics parallel I/O
1	strobe	1		1	GND	19
2	P0	2		2	GND	20
3	P1	3		3	GND	21
4	P2	4		4	GND	22
5	P3	5		5	GND	23
6	P4	6		6	GND	24
7	P5	7		7	GND	25
8	P6	8		8	GND	26
9	P7	9		9	GND	27
10		10		10	GND	28
11	busy	11		11	GND	29
12		12		12	GND	30
13		13		13		31
14		14		14		32
15		15		15		33
16		16		16		34
17	GND	17		17		35
18		18		18	GND	36
	GND	19		19	GND	
	GND	20		20	GND	
	video	21		21	GND	
	video	22		22	GND	
	video	23		23	GND	
	vert sync	24		24	GND	
	horiz sync	25		25	GND	
	/RST	26		26	GND	
	DUMP	27		27	GND	
	CLEAR	28		28	GND	
	CONT	29		29	GND	
		30		30		
	BUSY LED-	31		31	LED+	
	HOLD LED-	32		32	LED+	

APPENDIX B. Command list and definitions

$A.. Advanced Graphics functions

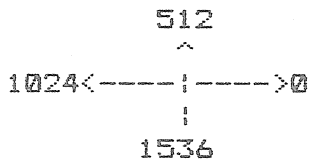
$AA - draws an arc centered at current position
E.G. $AA +A +B +C

$AC - draws a circle centered at current postion
E.G. $AC +A

$AP - draws a pie centered at current position
E.G. $AP +A +B +C

WHERE:
A = ARC/CIRCLE/PIE Radius in pixels
B = Beginning angle
C = Ending Angle

Angles are in units of 2048/360 degrees.



$AFB -Area fill to boundary using the area fill pattern loaded in the most recent $F sequence.

** NOTE- MOVE CURSOR BEFORE FILL. IF SITTING ON A VECTOR OF THE SAME COLOR, THE FILL WILL ABORT.

$AFR+ -Area fill a rectangular region defined as follows:
E.G. $AFR+ X0 Y0 X1 Y1

WHERE: X0,Y0 and X1 and Y1 specify the co-ordinates of the opposite corners of the rectangle to be filled

All reading or loading proceeds from the current position with increasing X and decreasing Y (left to right and top to bottom of screen).

Loading Data (from host system)

$AL -Load display memory by transferring a character/pixel from host for N pixels
$AL+N
E.G. $AL+N or $ALA+N

$ALB+ -Load specified number of display bytes (8 pixels per byte) beginning at current position (modulus 8)
I.E. Data follows "+" specifier as one 8 bit character per 8 pixels
E.G. $ALB+N

Reading Data (from GB1)

$AR -Read display memory GB sends a character/pixel from screen for the specified number of pixels(N) to the host and converts color code 0-1 to ASCII codes 0-1

$AR+N or $ARA+N

$ARB -Read display memory by transferring N bytes (8 pixels per byte) beginning at current position (Modulus 8). Data follows at 1 8-bit character per 8 pixels.
$ALB+N

$B BLOCK MODE PLOTTING
The $B command implements the block character drawing sequence. It draws the point mode character and replicates each dot in two dimensions as specified in the $F command (E.G. $F19V will write tall thin letters vertically with either $C or $B plotting). Character drawing proceeds as in the $C description.
FORMAT: $B+TEST prints "TEST" in block letters.

$BL LOAD PROGRAMMABLE CHARACTER SET
The $BL loads an 8 X 16 bit character design into RAM. It uses 16 8-bit words, plotting LSB first. The integer immediately following '$BL+___' is the ASCII code of the character which will be re-assigned.
In the format below, the new character is called to 65, the ASCII code for 'A'.

FORMAT: $BL+65+1+2+3+4+5+6+7+8+9+10+11+12+13+14+15+16
$BL+M + n0 +n1.....

After the above command whenever $BSA+A is struck the shape defined by the bit map of n0-n15 will be seen

$BS_ CHOOSE CHARACTER SET, BLOCK MODE

This command calls for character sets A (programmable), B (ROM set B), or C (ROM set C). Printing will be in block mode. Character drawing proceeds as described in $C description.

FORMAT: $BSA+abcd (Character set A, programmable RAM)
$BSB+abcd (Character set B, from ROM)
$BSC+abcd (Character set B, from ROM)
$BSR+abcd (Restore from set A to last-used ROM set)

$C_ CHARACTER MODE PLOTTING

This command initiates the drawing sequence:

1. The pen is lifted.
2. Search mode is initiated for a "+".
3. All characters after the "+" are drawn as specified by most recent $F command.
4. Any control character <CR>,<LF> terminates the $C command.
5. Pen is up at completion of command.
6. The A/N may be either points or vectors, determined by most recent command.

FORMAT: $C+TEST prints "TEST" in point or vector.

$CL LOAD POINT CHARACTER RAM WITH NEW CHARACTER

The $CL command is the same as $BL. It permits down-loading altered character sets, which are subject to $F modifiers just as are other typefaces and graphics. See also Section VI.D. Character Plotting. In the format below, the new character is called by the ASCII for 'A'.

FORMAT: $CL+65+1+2+3+4+5+6+7+8+9+10+11+12+13+14+15+16

$CS_ SELECT CHARACTER SET

As described under $BS_, selects either ROM set B or C, or alterable RAM set A. Prints in either point or vector mode, subject to $F formatting.

FORMAT: '$CSA+abcd' prints 'abcd' in set A characters
'$CSB+abcd' prints 'abcd' in set B characters
'$CSC+abcd' prints 'abcd' in set C characters
'$CSR+abcd' restores from set A to last-used ROM set.

$D PEN DOWN COMMAND

This lowers the "pen" and is necessary after lifting the pen or to initiate a plot. It is also necessary after certain advanced drawing sequences, since they lift the pen on completion.

$E MISCEL. SETUP
This allows setup of VIDEO and communication
The first character following $E specifies the function.

$E0,$E1,$E2,$E3,$E6,$E7 reserved codes
$E4- Enable "link" XON/XOFF (DEFAULT)
$E5- Disable "link" XON/XOFF
$E8- Next decimal value is new "XON" character
$E9- Next decimal value is new "XOFF" character
$EA- 512 x 480 video resolution (DEFAULT), 525 lines
$EB- 512 x 512 video resolution, 553 lines
$EC- 512 x 512 video resolution, 625 lines,(50.24 hz)
$ED- 512 x 480 video resolution, 533 lines,(59.96 Hz)
$EE- 512 x 512 video resolution, 627 lines,(50.08 Hz)
$EK- Clipping during drawing enabled
$EW- Clipping disabled (Wraparound)

**NOTE Options 8 and 9 require the decimal value to be in the format of the most recent $R

$F CHARACTER FORMAT SPECIFIER
This command modifies the $C, $AFB, $B, and $G drawing commands. Additional modifiers may follow $F (in any order). Each specifier remains in effect until the operator gives a new $F command or the GRAPH BOX is reset.

$FH Horizontal writing (DEFAULT)
$FV Vertical writing
$FG Lower case =Greek symbols
$FL Lower case
$F/ All zeroes slashed (DEFAULT)
$F\ All zeroes non-slashed
$FP Selects normal plotting size
$FB Selects Block size
$FC Complements bit plane
$FR Replaces bit plane for patterned vectors

Additionally, two digits (0-9) may be included. The first specifies the height (9 is largest), the second, the width. This applies to ALL Character and Symbol drawing.

The $F command also defines the vector pattern and fill pattern used when drawing. The first decimal number following the $F (in the proper format) is the 16 bit vector pattern used to draw. (LSB plotted first). The subsequent 16 numbers will specify the bit pattern used to fill areas. (LSB first proceeding down the screen.)

EXAMPLE:$F usage

```
10 PRINT "$F38VG/B"
```

When a $C,$B, or $G command is given this will print tall (8), thin (3) letters vertically (V), with Greek symbols (G) displayed when input is lower-case ASCII (as opposed to upper-case data entry), all zeroes slashed (/), and letters drawn in block size (B).

For area fills the same principle applies: 16 decimal numbers may be specified following the $F command . These will become the bit pattern for an area fill.

As before, once a format has been specified, it will be used until a change has been made ; hence it is not necessary to define the ($F)ormat unless a change must be made.

$G GRAPHICAL PLOT COMMAND

This provides 10 symbols which may be used as markers. The command does not change the current mode, but the symbols are always done in vector mode and the pen is always lifted on completion. The digits 0-9 indicate the symbol to be used. The $F command will affect the size/height/width. It also will affect the vectors used to draw the shape. The 'pen' does not advance after drawing one of these symbols as it would for printing of text.

$G0	-Diamond	$G5	-I
$G1	-Plus	$G6	-Right triangle
$G2	-Square	$G7	-Left triangle
$G3	-Tilted Square	$G8	-Down triangle
$G4	-Octagon	$G9	-Up triangle

$H HOLD

This command halts execution until the CONT button on the front panel is pushed.

$I Timed hold. Wait N x.1 sec intervals or continue if CONT button is pressed.
FORMAT: $I20 holds for 2 seconds.

$K CLEAR SCREEN

This commmand clears the monitor and reverses the color. Its function is identical to the CLEAR button. It may be used with the $Z command to clear to a color.
FORMAT: $K clears and reverses, $Z0$K clears to black.

This may also be used with the modifier R to clear a rectangular region in the same manner.
FORMAT: $KR+ X0 Y0 X1 Y1

WHERE X0,Y0 and X1,Y1= the opposite corners of the rectangle

$L MOVE RELATIVE TO CURRENT POSITION

This command moves the "pen" relative to current position. Data is accepted in the most recently specified format. (DX,DY,DX,DY etc.) The position would then be (X+DX,Y+DY) then (X=X+DX,Y=Y+DY etc.)
FORMAT: "$L 0 100 100 0 0 -100 -100 0" draws a box 100 pixels on a side, if GRAPH BOX is in free format ($R or $R0) and the 'pen' is down ($D).

$M MOVE RELATIVE TO ORIGIN

This command moves the "pen" relative to the current origin. Data is accepted in the most recently specified format (X,Y,X,Y etc.). The position would then be (XO+X, YO+Y etc.).
FORMAT: "$M 100 100 100 200 200 200 200 100 100 100" would draw a box 100 pixels on a side (from 100,100 to 100,200 to 200,200 to 200,100 to 100,100), if GRAPH BOX is in free format ($R or $R0) and the 'pen' is down ($D).

$N BLANKING OF SCREEN

Permits choice of blanking CRT or unblanking monitor.
$NB blanks the CRT.
$NU unblanks the CRT.

$O SET ORIGIN

Data following in the proper format defines the origin at those absolute coordinates. Since the scaling is 512 X 512 the middle of the screen would be (256, 256).
FORMAT: "$O02560256" in the $R4 format would set 256 X 256 to (0,0).

$P POINT MODE (DEFAULT)

This mode plots only the end points. The $C letters and characters are drawn in points rather than lines or vectors while GRAPH BOX is in point mode.

$Q MACRO BUFFER COMMAND

The macro command group permits user storage of drawing routines callable by 3 characters and a number. There are five commands altogether:

$QD+N - Deletes macro N from buffer.

$QE+N - Enters data and commands following as macro N. Subsequent input will all be stored, including <CR> and <LF>, until any $Q is encountered. This terminates macro input.

$QR - Restores saved drawing parameters

$QS - Saves current drawing parameters (origin, position, line and area fill pattern, input format, drawing color, drawing format, and pen position)

$QX+N - Executes macro N.

$QZ - Zeroes macro buffer (erases all macros).

**NOTE There are 12K buffer space, and 256 macros (numbered 1-256) available. A macro may contain any sequence of commands or text except $Q, including control characters, <CR>,LF and non-printing characters.

$R INITIATE GRAPHICS/SPECIFY DATA FORMAT

This resets the graphics functions of the Graph Box. It also specifies the form (INTEGER or FREE FORMAT) in which all data will be expected.

INTEGER FORMAT ($R1-$R9)

1. Immediately following a command, if the data is a "-", "+" or a digit, data is assumed to follow in the specified format until a <CR>, <LF>, or other control character is encountered. Zeroes, spaces, "+" and "-" are counted in the number of digits expected.

2. If the first character following a command is not one of the above, the GB enters a search mode. As soon as the character + is encountered all data is assumed to be in the specified format. Termination of the data by a <CR>,<LF> or other control character causes the GB to initiate the search mode again.

FREE FORMAT ($R or $R0) (DEFAULT)

1. Data are assumed to follow in a FREE FORMAT (i.e. data separated by a +, -, or space(s)) until terminated by <CR>,<LF> or other control character.
2. If the GRAPH BOX encounters an ASCII control character,it enters the search mode. When it again finds the+, the data is assumed to continue in FREE FORMAT.

NEW GRAPHICS COMMANDS ARE ALWAYS RECOGNIZED IN THE SEARCH MODE.

$S STOP PLOTTING

This turns off the GB.

$T TRANSFER DATA TO EPSON (PARALLEL PORT)
There are 7 commands which are part of this sequence.

1. $TL +A +B +C +D +E +F - Large plot (60 x 72 dots/in)

2. $TG +A +B +C +D +E +F - Graphics Mode (72 x 72 dots/in) (DEFAULT)

3. $TH +A +B +C +D +E +F - Hi Res (120 x 144 dots/in)

WHERE :

A = Printing zoom factor (1-16)
B = Epson columns to skip (SAME AS PIXELS)
C = First X coordinate
D = Last X coordinate
E = First Y coordinate
F = Last Y coordinate

NOTE** MODIFIERS C,D,E,F PERMIT SCREEN WINDOWING
A PERMITS PRINTING ZOOM

4. $TC+ A + <CHARACTERS> Transfers A succeeding characters

FORMAT: "$TC+4+ABCDEFG" prints "ABCD" since only 4 characters are transferred.

5. $TD+ A..(data) Transfers 8-bit binary equivalent of decimal data to printer

6. $TV Dumps screen as displayed (72 x 72 dots/inch) to EPSON FX80/100. Other printers may distort. (DEFAULT)

7. $TT.. Reassigns DUMP (FRONT PANEL) AS FOLLOWS:

$TTL	Dump 60 x 72 dots/inch
$TTG	Dump 72 x 72 dots/inch
$TTH	Dump 120 x 144 dots/inch

$U UP PEN
This raises the "pen" to enable moving from one point to another without leaving a trail or line.

$V VECTOR MODE
Draws a line between endpoints (data received in proper $R mode)

$X AUTO-INCREMENT X AXIS

This automatically increments the X axis by a specified value. Subsequent data are assumed to be the Y values, relative to origin. DATA MUST BE IN PROPER FORMAT.
FORMAT: "RX 5 10 15 25 10 35" plots a series of points (10,15,25,10,35) with GB incrementing 5 pixels along the X axis each time.

$Y AUTO-INCREMENT Y AXIS

This automatically increments the Y axis by a specified value. Subsequent data are assumed to be the X values, relative to origin. DATA MUST BE IN PROPER FORMAT.

```
THUS      XO+X,Y    THEN Y=Y+STEP VALUE
          XO+X,Y    etc.
```

$Z SETS VIDEO "COLOR"

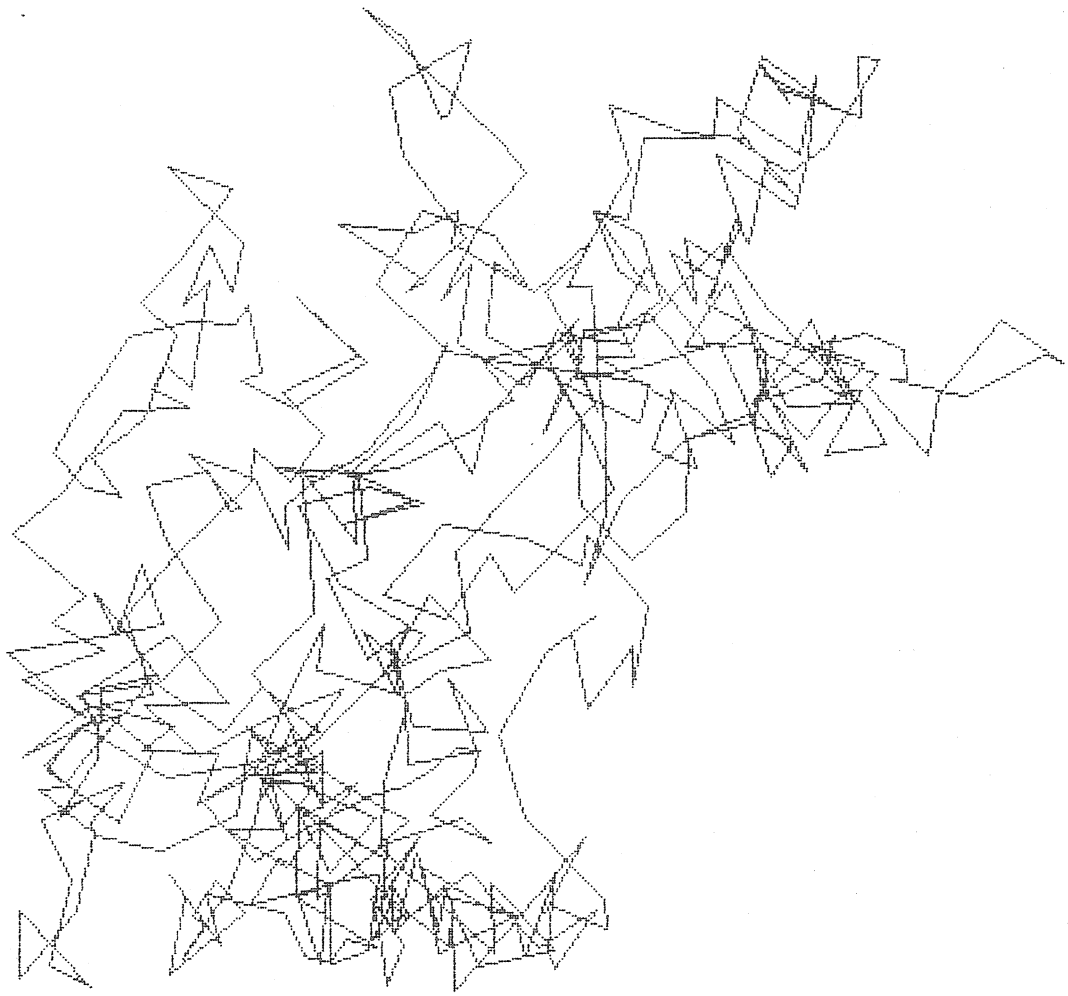
This sets the "color" of the drawing to:

1 = White
0 = Black

FORMAT: PRINT "$Z0$K" CLEARS SCREEN TO BLACK.

```
10  REM Unit 16 must be spooled to the Graphics Board via
20  REM the VGBG driver.
30  REM Use the PDOS command SU 16,VGBG  or
40  REM use the BASIC command SPOOL 16 TO "VGBG"
50  REM and then use UN(IT) 17 for output to VGBG and console)
60  REM
65  SPOOL 16 TO "VGBG"
70  REM **********
80  DIM C[2]
90  REM **********
100  P=P+1
101  X=256
102  Y=256
105  UNIT 1
106  PRINT "Random Walk  #";P
107  UNIT 16
110  PRINT "$R$V$Z0$K$Z1$0 0 0"
120  PRINT "$U$M 0 0$F22HL$B+Random Walk  # ";P
130  PRINT "$U$M 256 256$D"
10  FOR J=1 TO 500
160   GOSUB 500
170  NEXT J
180  PRINT "$I 10"
190  GOTO 100
200  REM **********
500  FOR I=1 TO 2
510   C[I]=1-2*RND
520  NEXT I
530  X=X+32*C[1]
540  IF X>480: X=450
550  IF X<32: X=62
560  Y=Y+32*C[2]
570  IF Y>480: Y=450
580  IF Y<32: Y=62
590  PRINT #"999";"$M";X;Y
700  RETURN
```

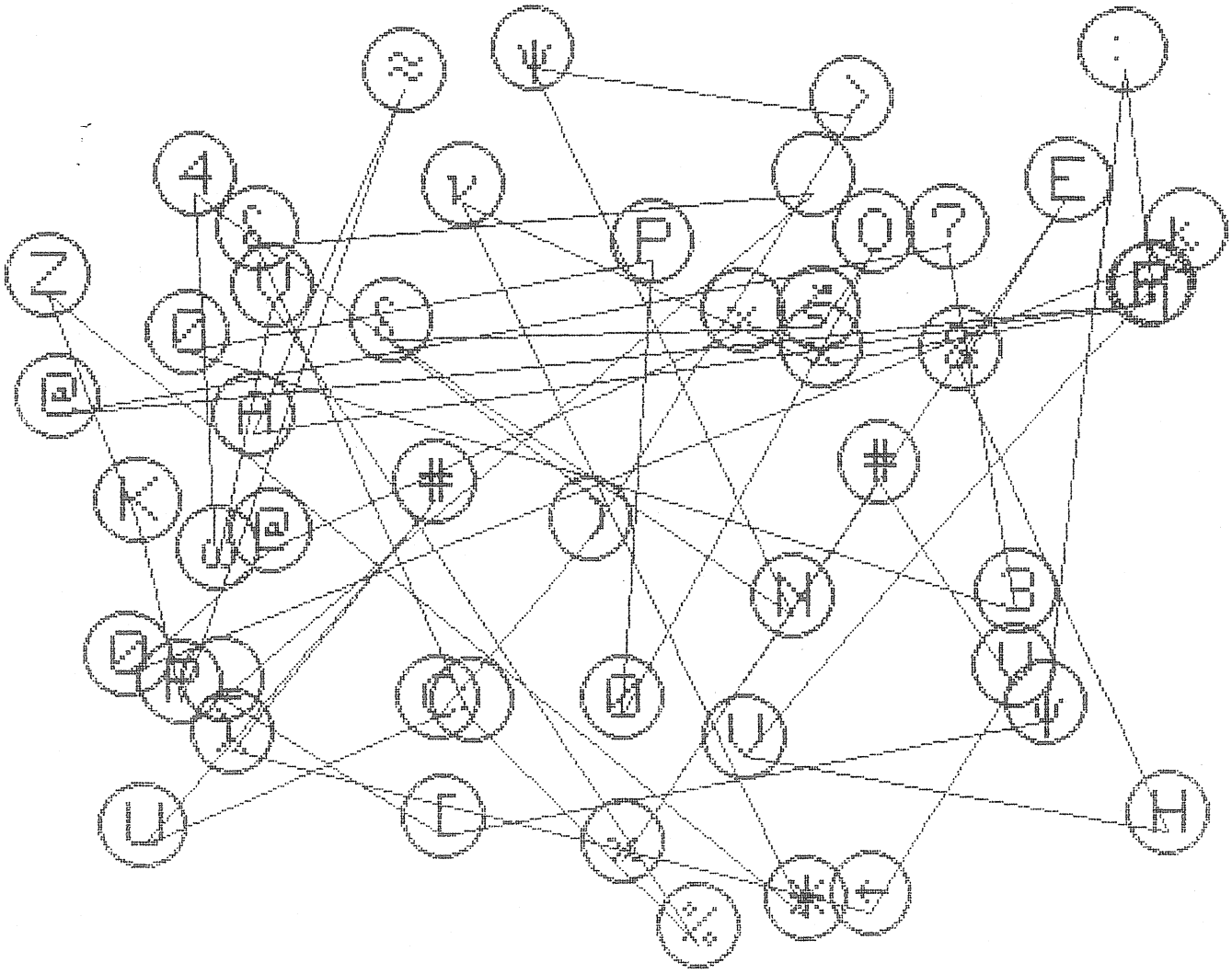
Random Walk Demo output is on following page



Random Walk # 2

```
10  REM Unit 16 must be spooled to the Graphics Board via
20  REM the VGBG driver.
30  REM Use the PDOS command SU 16,VGBG  or
40  REM use the BASIC command SPOOL 16 TO "VGBG"
50  REM and then use UN(IT) 17 for output to VGBG and console)
55  SPOOL 16 TO "VGBG"
56  UNIT 17
60  REM
70  REM **********
80  DIM C[3]
90  REM **********
100  P=P+1
110  PRINT "$R$V$Z0$K$Z1$O 0 0"
120  PRINT "$U$M 0 0$F22$B+PICTURE # ";P
130  PRINT "$U$M 256 256$D$F11HG"
150  FOR X=1 TO 50
160   GOSUB 500
170  NEXT X
180  PRINT "$I 20"
190  GOTO 100
200  REM **********
500  FOR I=1 TO 3
510   C[I]=RND
520  NEXT I
530  PRINT "$D$M";
540  PRINT #"E999";16+480*C[2];76+360*C[3]
550  $S=%32+96*C[1]
560  PRINT "$U$L-8 0$B+";$S
570  PRINT "$L-8 8$AC 16 17 $L 0-8"
590  RETURN
```

Picture Plot demo output on following page

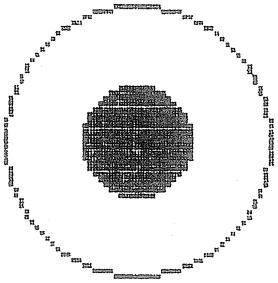


PICTURE # 1

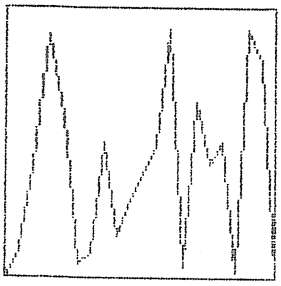
```
10  REM DEMO SOFTWARE FOR GB1
20  REM WRITTEN  10/15/86
30  SPOOL 16 TO "VGBG"  !SEND PRINT STATEMENT TO GB1
40  UNIT 17  !SHOW ON CONSUL TOO
50  PRINT "$R"  !ACTIVATE GB1 IN FREE FORMAT
60  PRINT "$Z0$K"  !CLEAR SCREEN TO BLACK ($Z0)
70  PRINT "$M 256 256 "  !MOVE TO CENTER SCREEN
80  PRINT "$V$AC 10 25 "  !VECTOR MODE/ CIRCLES OF 10&25
90  PRINT "$AFB"  !FILL CIRCLE TO BOUNDARY
100  PRINT "$TG+2+100+200+290+200+290 "
110  REM ABOVE PRINTS ONLY CENTER OF SCREEN ZOOMED
120  REM
130  REM
140  REM SEE EXAMPLE #1 FOR OUTPUT
150  PRINT "$Z0$K"  !CLEAR SCREEN TO BLACK
160  PRINT "$M 100 50 $D "  !MOVE TO LOWER LEFT QUADRAND/PEN DOWN
170  PRINT "$M 200 50 200 150 100 150 100 50 "  ! MAKE BOX
180  PRINT "$X 5 "  !SET AUTO INCREMENT FOR X AT 5 PIXELS
190  PRINT "+60 90 140 110 55 "  !Y DATA
200  PRINT "+59 100 65 79 88 "
210  PRINT "+99 142 54 115 92 "
220  PRINT "+100 52 142 130 50 "
230  PRINT "$U$M 100 25 "  ! RAISE PEN AND MOVE
240  PRINT "$FH00$CSB+Test Program"  !SET CHAR ATTRIBUTES ($FH00) & LABLE
250  PRINT "$TG+1+80+95+210+15+160"  !PRINT WINDOW AREA 1:1
260  REM
270  REM SEE EXAMPLE #2 FOR OUTPUT
280  REM
290  REM MACRO EXAMPLE
300  REM
310  PRINT "$R$QE+1+$Z0$K$O 0 0 $V$U";  !SET PARAMS/INIT MACRO #1
320  PRINT "$M 100 100 $D +200 100 200 200 100 200 100 100 ";
325  PRINT "$F18";  !SET SIZE OF CHARACTERS
330  PRINT "$U$M 110 110 $CSB+ERROR": PRINT   !WRITE MESSAGE
340  PRINT "$Q"  !END MACRO ENTRY
350  PRINT "$H"  ! HOLD PROCESSING UNTIL 'CONT' BUTTON PRESSED
360  PRINT "$QX+1"  !EXECUTE MACRO #1
370  RESET   !CLOSE DRIVER (MUST BE DONE TO RERUN PROGRAM)
380  END
```

Outputs on following Page

Example 1

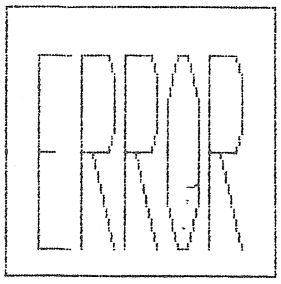


Example 2



Test Program

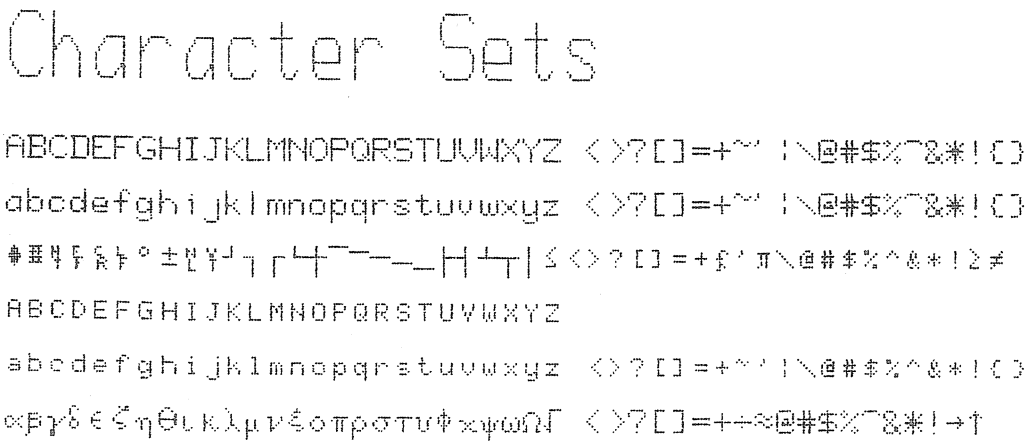
Example 3



Character generation program

```
 1520  PRINT "$Z0$K "
 1530  PRINT "$Z1$M 10 450 $F12$CSC+Character Sets "
 1540  PRINT "$M 10 420 $F00 $BSB+ABCDEFGHIJKLMNOPQRSTUVWXYZ <>?[]=+~'¦\@#$%^&*!
{} "
 1550  PRINT "$M 10 400 $BSB+abcdefghijklmnopqrstuvwxyz <>?[]=+~'¦\@#$%^&*!{}  "
 1560  PRINT "$m 10 380 $FG$BSC+abcdefghijklmnopqrstuvwxyz<>?[]=+~'¦\@#$%^&*!{}
"
 1570  PRINT "$M 10 360 $FL00$BSC+ABCDEFGHIJKLMNOPQRSTUVWXYZ "
 1580  PRINT "$M 10 340 $BSC+abcdefghijklmnopqrstuvwxyz <>?[]=+~'¦\@#$%^&*!{}  "
 1590  PRINT "$m 10 320 $FG$BSB+abcdefghijklmnopqrstuvwxyz <>?[]=+~`@#$%^&*!{}  "
```

Sample output from above



COMPONENTS FOR GB1 VME BOARD--- MICROGRAPHICS INC. 10/31/86

U1-U8	4164 (150 ns) SOCKETS
U9	LS112
U10	LS138
U11	LS00
U12	S260
U13	S10
U14	LS02
U15	LS02
U16	LS74
U17	LS244
U18	LS166
U19-22	74153
U23	S85
U24-25	LS138
U26	LS175
U27	LS54
U28	LS74
U29	LS163
U30	74279
U31	LS164
U32	68B45 SOCKET
U33	LS139
U34	S287 SOCKET
U35	68B09E SOCKET
U36	27128 Hitachi SOCKET
	HN4827128G-25 or equal
U37-39	LS374
U40	LS244
U41	LS374
U42	S04
U43	S74
U44	LS54
U45	LS259
U46	LS244
U47	LS374
U48	LS08
U49	7433
U50-53	8136
U54	LS244
U55	F245
U56	LS375

C1-C10	ALL .1 SPRAGUE
C13-14	(923CZ5U104M050B)
C16-47	
C49-55	

RN1	6 PIN 470
RN2	8 PIN 10K
RN3	6 PIN 10K
RN4	10 PIN 10K
RN5	8 PIN 10K
RN6	8 PIN 10K

C11	560pf (Disc)
C12	390pf (Disc)
C15	15pf (Disc)

C48,56	15 TANT 20V
C57,58	
C59,C60	
C61	

R1-2	47 1/4W 5%
R3	1K
R4-14	43
R15-16	560
R17	10K
R18	1K
R19	100
R20	100K
R21	220
R22	22K
R23	270
R24	100
R25-26	150

GB1 PARTS LIST CONT.

MISC PARTS

2 DIN CONNECTORS (ERNI 533602)
2 LEDS DIALITE (#550 0406)
4 SWITCHES ALCO TPB-11(BUTTONS #C 41 0)
3 DIODES (1N914 or 1N4148)
1 2N3643
1 XTAL 14.31818 MHZ (NYMPH NMP 143)
HEADERS FOR JUMPERS (SINGLE HIGH DBL ROW .1 CENTERS)
K2 NOT POPULATED (SAME AS ABOVE)

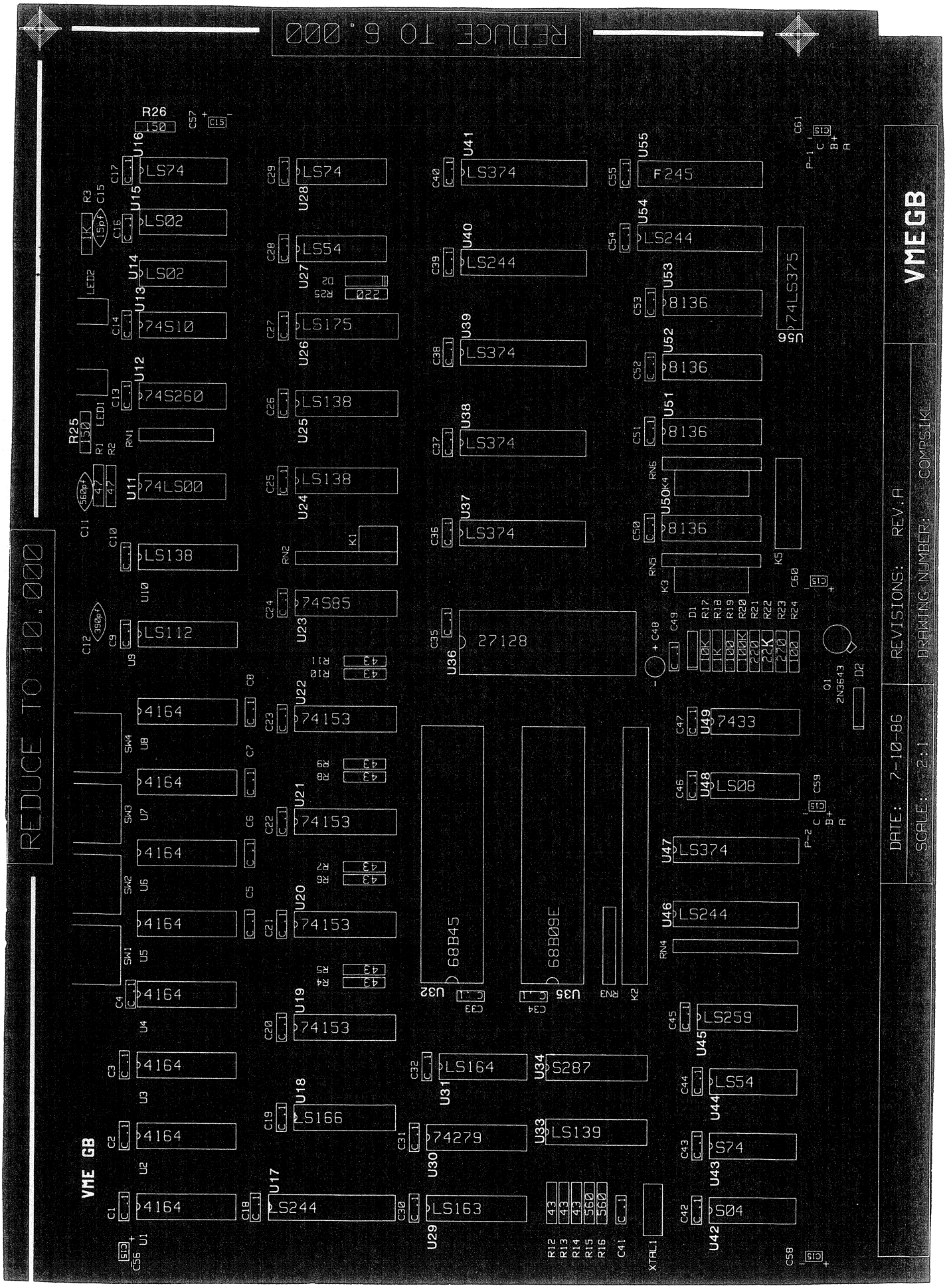
CAPACITORS' SUPPLIERS

C11 CENTRAL LAB #DD30561 OR EQUAVALENT
C12 " #DD30391 OR EQUAVALENT
C13 " #DD30150 OR EQUAVALENT

C48,56,57,58,89,60,61 KEMET #T362B156K050ASC
OR EQUAVALENT

ALL SOCKETS GOLD AND LOW PROFILE
ALL RESISTOR NETWORKS STANDARD LOW PROFILE SIPS

NOTE: THE 68B45 MUST NOT HAVE THE EXTENDED INSTRUCTION SET


REDUCE TO 6.000
REDUCE TO 10.000
VME GB
VMEGB
REVISIONS: REV.A
DRAWING NUMBER: COMPSIKL
DATE: 7-10-86
SCALE: 2:1
U1 74LS00
U2 74LS10
U3 74LS02
U4 74LS04
U5 74LS08
U6 74LS32
U7 74LS86
U8 74LS138
U9 74LS112
U10 74S85
U11 74LS138
U12 74S260
U13 74S10
U14 74LS02
U15 74LS02
U16 74LS74
U17 74LS244
U18 74LS166
U19 74153
U20 74153
U21 74153
U22 74153
U23 74S85
U24 74LS138
U25 74LS138
U26 74LS175
U27 74LS54
U28 74LS74
U29 74LS163
U30 74279
U31 74LS164
U32 68B45
U33 74LS139
U34 74S287
U35 68B09E
U36 27128
U37 74LS374
U38 74LS374
U39 74LS374
U40 74LS244
U41 74LS374
U42 74S04
U43 74S74
U44 74LS54
U45 74LS259
U46 74LS244
U47 74LS374
U48 74LS08
U49 74LS33
U50 8136
U51 8136
U52 8136
U53 8136
U54 74LS244
U55 74F245
U56 74LS375
4164
R26 150
R25 150
R1 47
R2 47
R3 1K
R4 43
R5 43
R6 43
R7 43
R8 43
R9 43
R10 43
R11 43
R12 43
R13 43
R14 43
R15 560
R16 560
R17 10K
R18 1K
R19 100
R20 100K
R21 220
R22 22K
R23 220
R24 100
R25 220
RN1
RN2
RN3
RN4
RN5
RN6 10K
C1 .1
C2 .1
C3 .1
C4 .1
C5 .1
C6 .1
C7 .1
C8 .1
C9 .1
C10 .1
C11 560pf
C12 390pf
C13 .1
C14 .1
C15 .1
C16 .1
C17 .1
C18 .1
C19 .1
C20 .1
C21 .1
C22 .1
C23 .1
C24 .1
C25 .1
C26 .1
C27 .1
C28 .1
C29 .1
C30 .1
C31 .1
C32 .1
C33 .1
C34 .1
C35 .1
C36 .1
C37 .1
C38 .1
C39 .1
C40 .1
C41 .1
C42 .1
C43 .1
C44 .1
C45 .1
C46 .1
C47 .1
C48 .1
C49 .1
C50 .1
C51 .1
C52 .1
C53 .1
C54 .1
C55 .1
C56 10
C57 15
C58 10
C59 10
C60 10
C61 10
LED1
LED2
D1
D2
Q1 2N3643
K1
K2
K3
K4
K5
XTAL1
SW1
SW2
SW3
SW4
P-1
P-2

```
*       VMEGBG:SR                  20-NOV-85
*****************************************************
*                                                   *
*       68K PDOS 'VGBG' DRIVER                      *
*       OUTPUT TO VMEGB-1 GRAPHICS BOARD            *
*              (GRAPHICS PORT)                      *
*                                                   *
*****************************************************
*                                                   *
*       DRIVER GENERATION                           *
*                                                   *
* ASSEMBLE:                                         *
*       MASM VMEGBG:SR,#VMEGBG:RB                   *
*                                                   *
* MAKE A SYSTEM FILE:                               *
*       MSYFL VMEGBG:RB,#VGBG                       *
*                                                   *
* SET DRIVER ATTRIBUTE:                             *
*       SA VGBG,DR                                  *
*                                                   *
*****************************************************
*
*       D5.L = CHARACTER COUNT (-1 = LINE)
*       D7.W = CHANNEL STATUS
*       (A2) = DRIVER BASE + 4
*       (A3) = MEMORY BUFFER
*       (A4) = FILE SLOT
*       (A5) = SYSRAM
*       (A6) = TASK TCB
*       (A7) = RETURN ADDRESS
*
        OPT     PDOS
*
        SECTION 0
DVGB    DC.W    $A55A           ;DRIVER ID
DROP    BRA.S   OPEN            ; 2 OPEN
DRCL    BRA.S   CLOSE           ; 4 CLOSE
DRRD    BRA.S   READ            ; 6 READ
DRWR    BRA.S   WRITE           ; 8 WRITE
DRPS    MOVEQ.L #80,D0          ;10 POSITION
*
*****************************************************
*       POSITION = ERROR 70
*
        RTS
*
*****************************************************
*       READ FROM DRIVER = ERROR 80
*
READ    MOVEQ.L #80,D0          ;ERROR 80
        RTS
*
*****************************************************
*       OPEN DRIVER
*
OPEN
*
*****************************************************
*       CLOSE DRIVER
*
```

```
CLOSE   CLR.W   D0              ;RETURN .EQ.
        RTS
*
*******************************************************
*       WRITE TO DRIVER
*
WRITE   MOVE.B  (A3)+,D2        ;GET CHARACTER, DONE ?
        BNE.S   @1              ;NO - SKIP
        TST.L   D5              ;YES - WRITE A LINE ?
        BMI.S   CLOSE           ;YES - FINISHED
*
@1      MOVEQ.L #100,D0         ;SET UP LOOP COUNTER
@2      BTST.B  #7,$FF0007      ;PORT READY ?
        BNE.S   @3              ;YES - SKIP
        DBNE    D0,@2           ;END OF LOOPING ?
        MOVEQ.L #100,D0         ;1 SECOND DELAY
        MOVE.L  #128,D1         ;LOCAL
        XDEV                    ;DELAY #128 FOR 1 SECOND
        BNE.S   @1              ;FULL
        XSUI                    ;NOW SUSPEND PROCESS
        BRA.S   @1              ;ON RETURN TRY AGAIN
*
@3      MOVE.B  D2,$FF0001      ;DUMP CHARACTER
        SUBQ.L  #1,D5           ;DONE ?
        BNE.S   WRITE           ;NO - LOOP
        BRA.S   CLOSE           ;YES - EXIT
*
*******************************************************
*       DRIVER LENGTH CHECK
*
        IFLT    256-(*-DVGB+4)
        FAIL    ** DRIVER LENGTH ERROR ! **
        ENDC
*
*******************************************************
*       END
*
        END     DVGB
```

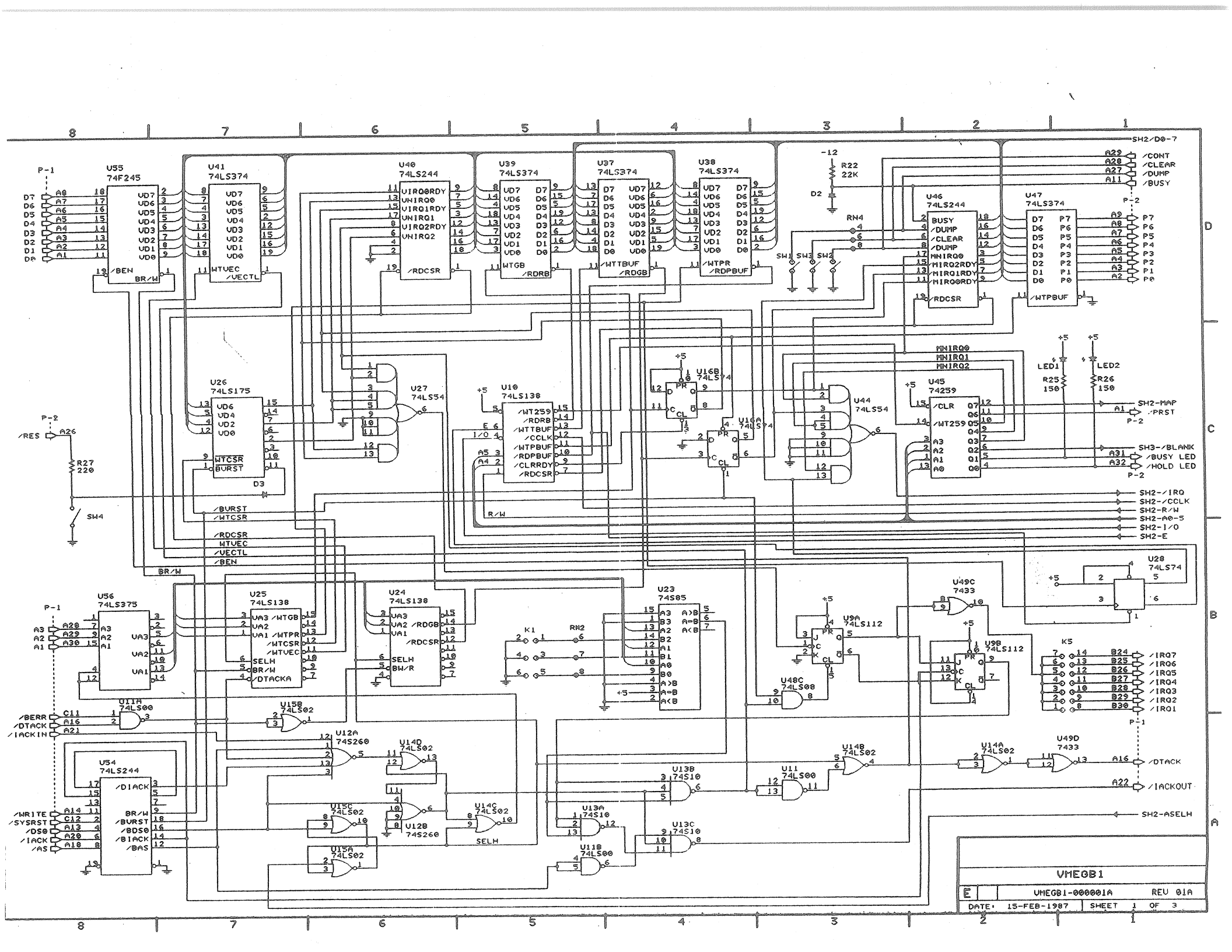
```
*       VMEGBP:SR                    20-NOV-85
*****************************************************
*                                                   *
*       68K PDOS 'VGBP' DRIVER                      *
*       OUTPUT TO VMEGB-1 GRAPHICS BOARD            *
*                (PRINTER PORT)                     *
*                                                   *
*****************************************************
*                                                   *
*       DRIVER GENERATION                           *
*                                                   *
* ASSEMBLE:                                         *
*       MASM VMEGBP:SR,#VMEGBP:RB                   *
*                                                   *
* MAKE A SYSTEM FILE:                               *
*       MSYFL VMEGBP:RB,#VGBP                       *
*                                                   *
* SET DRIVER ATTRIBUTE:                             *
*       SA VGBP,DR                                  *
*                                                   *
*****************************************************
*
*       D5.L = CHARACTER COUNT (-1 = LINE)
*       D7.W = CHANNEL STATUS
*       (A2) = DRIVER BASE + 4
*       (A3) = MEMORY BUFFER
*       (A4) = FILE SLOT
*       (A5) = SYSRAM
*       (A6) = TASK TCB
*       (A7) = RETURN ADDRESS
*
        OPT     PDOS
*
        SECTION 0
)VGP    DC.W    $A55A           ;DRIVER ID
)ROP    BRA.S   OPEN            ;  2 OPEN
)RCL    BRA.S   CLOSE           ;  4 CLOSE
)RRD    BRA.S   READ            ;  6 READ
)RWR    BRA.S   WRITE           ;  8 WRITE
)RPS    MOVEQ.L #70,D0          ;10 POSITION
*
*****************************************************
*       POSITION = ERROR 70
*
        RTS
*
*****************************************************
*       READ FROM DRIVER = ERROR 80
*
EAD     MOVEQ.L #80,D0          ;ERROR 80
        RTS
*
*****************************************************
*       OPEN DRIVER
*
)PEN
*
*****************************************************
*       CLOSE DRIVER
*
```

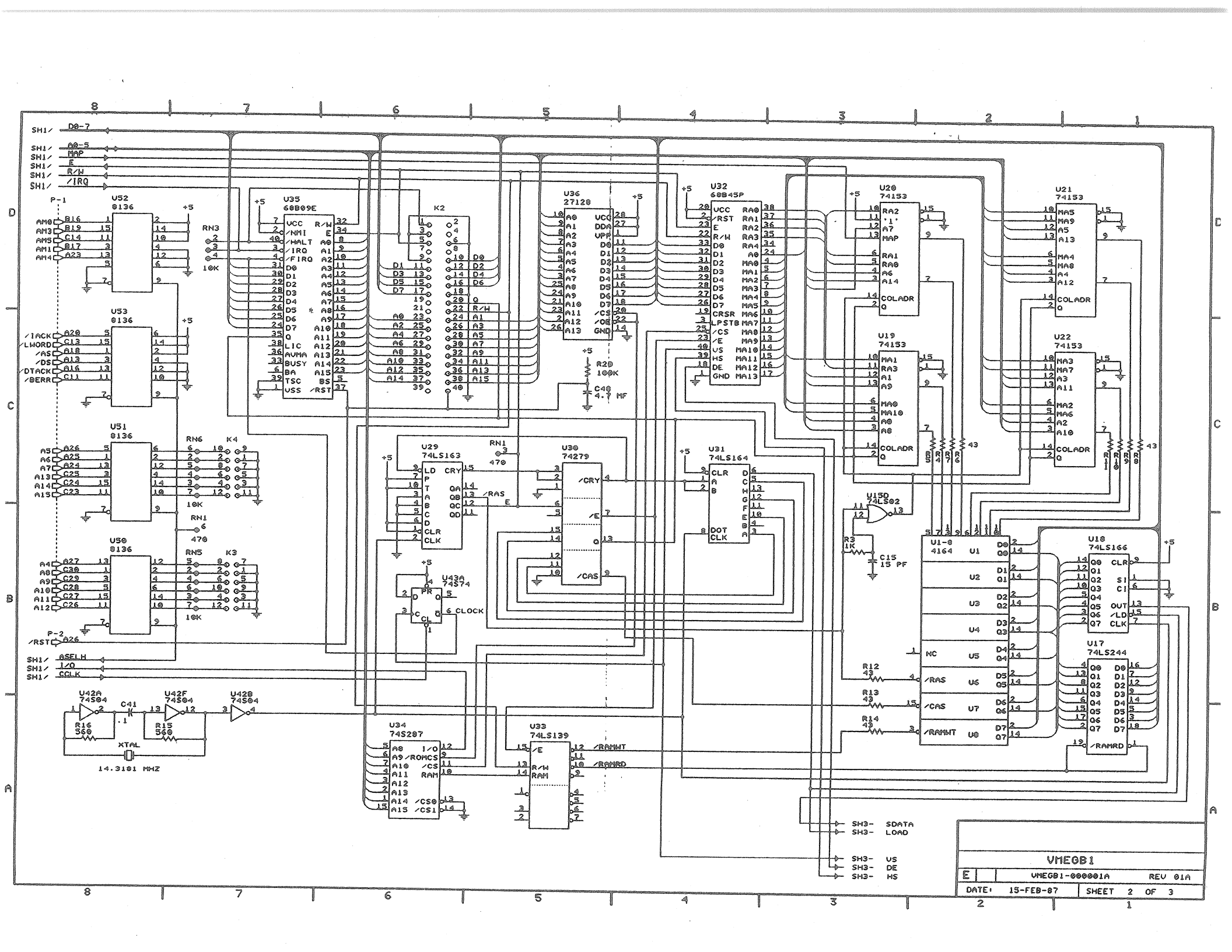
```
CLOSE   CLR.W   D0              ;RETURN .EQ.
        RTS
*
*****************************************************
*       WRITE TO DRIVER
*
WRITE   MOVE.B  (A3)+,D2        ;GET CHARACTER, DONE ?
        BNE.S   @1              ;NO - SKIP
        TST.L   D5              ;YES - WRITE A LINE ?
        BMI.S   CLOSE           ;YES - FINISHED
*
@1      CMPI.B  #$0A,D2         ;LF ?
        BEQ.S   @3              ;YES - SKIP
        CMPI.B  #$0D,D2         ;CR ?
        BNE.S   @2              ;NO - SKIP
        BSR.S   @4              ;WAIT FOR PORT TO BECOME READY
        MOVE.B  #$0A,$F0005     ;DUMP A LF
@2      BSR.S   @4              ;WAIT FOR PORT TO BECOME READY
        MOVE.B  D2,$FF0005      ;DUMP CHARACTER
@3      SUBQ.L  #1,D5           ;DONE ?
        BNE.S   WRITE           ;NO - LOOP
        BRA.S   CLOSE           ;ELSE FINISHED
*
@4      MOVEQ.L #100,D0         ;LOOP COUNTER
@5      BTST.B  #3,$FF0007      ;PORT READY ?
        BNE.S   @6              ;YES - EXIT
        DBNE    D0,@5           ;END OF LOOPING ?
        MOVEQ.L #100,D0         ;1 SECOND DELAY
        MOVE.L  #128,D1         ;LOCAL
        XDEV                    ;DELAY #128 FOR 1 SECOND
        BNE.S   @4              ;FULL
        XSUI                    ;NOW SUSPEND PROCESS
        BRA.S   @4              ;ON RETURN TRY AGAIN
@6      RTS                     ;EXIT
*
*****************************************************
*       DRIVER LENGTH CHECK
*
        IFLT    256-(*-DVGP+4)
        FAIL    ** DRIVER LENGTH ERROR ! **
        ENDC
*
*****************************************************
*       END
*
        END     DVGP
```

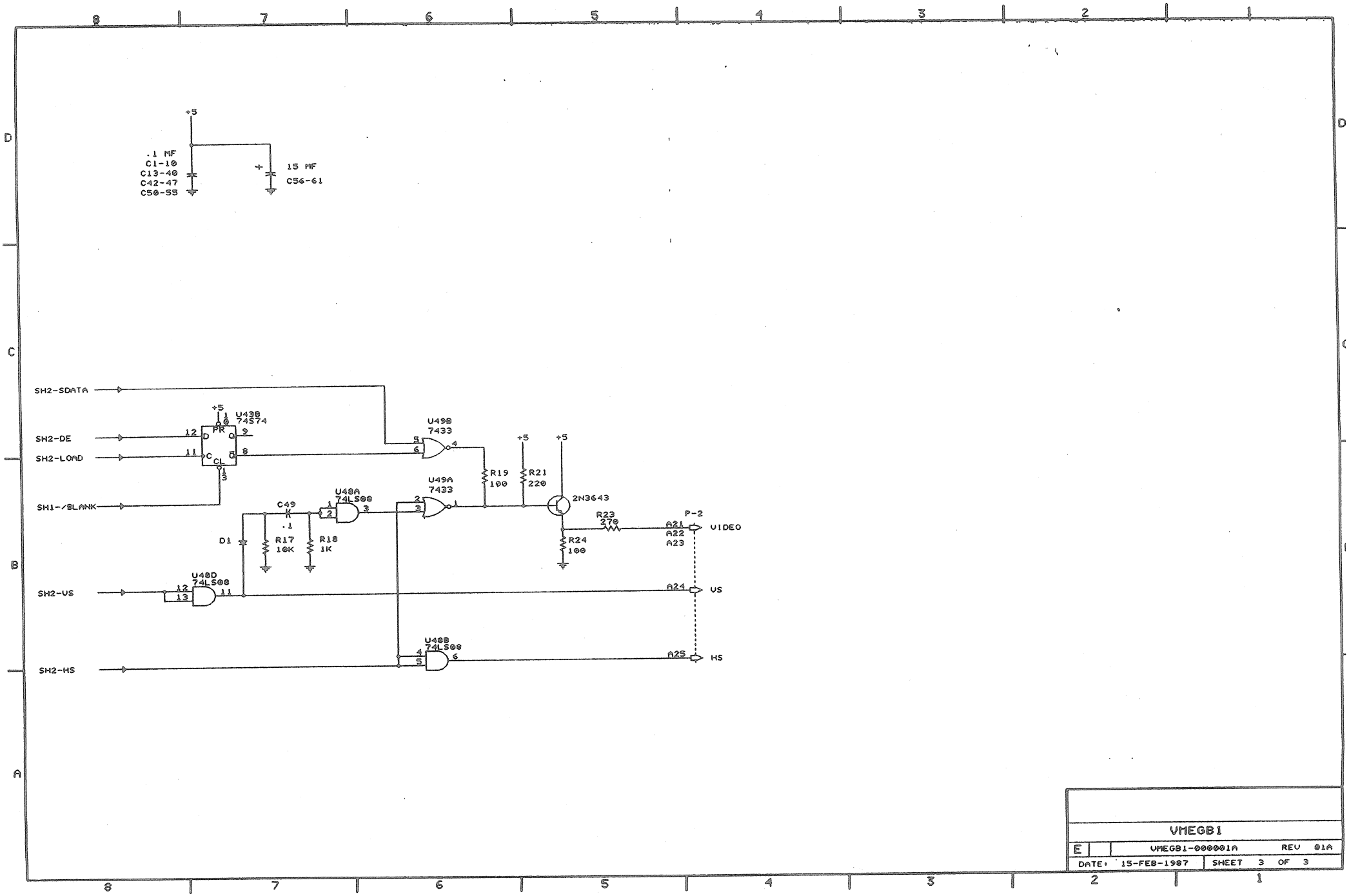
```
*       VMEGB:SR                12-OCT-85
*
*       GRAPHICS CONTROLLER TRANSFER TEST FOR VMEGB-1
*
VMEGB   EQU     $FF0000                 ;VMEGB ADDRESS
*
*
START:  XPMC    MES03                   ;PRINT STARTUP MESSAGE
        LEA     VMEGB,A0                ;VMEGB ADDRESS
*
*       1(A0)   IS DATA TO BOARD (WO)
*       3(A0)   IS DATA FROM BOARD (RO)
*       5(A0)   IS PRINTER DATA TO BOARD (WO)
*       7(A0)   IS BOARD STATUS (RW)
*       9(A0)   IS BOARD VECTOR (WO)
*
@1:     XGCC                            ;CHECK CHARACTER
        BLO.L   EXIT                    ;^C
        BLT.S   BRK1                    ;[ESC]
        BEQ.S   @3

        MOVE.B  D0,1(A0)                ;SEND CHARACTER TO VMEGB
        CMP.B   #13,D0                  ;<CR> ?
        BNE.S   @2
        XPCL                            ;<CR><LF>
        BRA.S   @3

@2:     XPCC                            ;ALL CHARACTERS EXCEPT <CR>

@3:     BTST    #5,7(A0)                ;DATA FROM VMEGB ?
        BEQ.S   @4                      ;NO - SKIP
        MOVE.B  3(A0),D0                ;GET CHARACTER
        ANDI.W  #$FF,D0                 ;ONE CHARACTER
        XPCC                            ;SEND CHARACTER TO SCREEN

@4:     BRA.S   @1                      ;GET ANOTHER

*
*       PRINTER PORT TRANSFER TEST FOR VMEGB-1
*
BRK1:   XPMC    MES01                   ;[ESC]
        XPMC    MES04                   ;PRINT STARTUP MESSAGE
        LEA     VMEGB,A0                ;VMEGB ADDRESS
*
*       1(A0)   IS DATA TO BOARD (WO)
*       3(A0)   IS DATA FROM BOARD (RO)
*       5(A0)   IS PRINTER DATA TO BOARD (WO)
*       7(A0)   IS BOARD STATUS (RW)
*       9(A0)   S BOARD VECTOR (WO)
*
@1:     XGCC                            ;CHECK CHARACTER
        BLO.L   EXIT                    ;^C
        BLT.S   BRK2                    ;[ESC]
        BEQ.S   @3

        MOVE.B  D0,5(A0)                ;SEND CHARACTER TO VMEGB
        CMP.B   #13,D0                  ;<CR> ?
        BNE.S   @2
        XPCL                            ;<CR><LF>
        MOVE.B  #10,5(A0)               ;<LF>
```

```
        BRA.S   @3

@2:     XPCC                            ;ALL CHARACTERS EXCEPT <CR>

@3:     BRA.S   @1

*
*       GRAPHICS CONTROLLER CONTINUOUS INPUT TEST
*
BRK2:   XPMC    MES01                   ;[ESC]
        XPMC    MES05                   ;PRINT MESSAGE
        LEA     VMEGB,A0                ;VMEGB ADDRESS
*
*       1(A0)   IS DATA TO BOARD (WO)
*       3(A0)   IS DATA FROM BOARD (RO)
*       5(A0)   IS PRINTER DATA TO BOARD (WO)
*       7(A0)   IS BOARD STATUS (RW)
*       9(A0)   IS BOARD VECTOR (WO)
*
@1:     XGCC                            ;TRY AN INPUT CHARACTER
        BLO.L   EXIT                    ;^C
        BLT.S   BRK3                    ;[ESC]

        BTST    #7,7(A0)                ;READY FOR A CHARACTER ?
        BEQ     @1                      ;NO - LOOP
        MOVE.B  #32,1(A0)               ;SEND A 'SPACE'
        BRA.S   @1                      ;AND LOOP

*
*       CONTINOUS CHARACTERS TO PRINTER
*
BRK3:   XPMC    MES01
        XPMC    MES06                   ;PRINT MESSAGE
        LEA     VMEGB,A0                ;VMEGB ADDRESS
*
*       1(A0)   IS DATA TO BOARD (WO)
*       3(A0)   IS DATA FROM BOARD (RO)
*       5(A0)   IS PRINTER DATA TO BOARD (WO)
*       7(A0)   IS BOARD STATUS (RW)
*       9(A0)   IS BOARD VECTOR (WO)
*
@1:     XGCC                            ;CHECK CHARACTER
        BLO.S   EXIT                    ;^C
        BLT.S   BRK4                    ;[ESC]

        BTST    #3,7(A0)                ;PRINTER READY ?
        BEQ.S   @1
        MOVE.B  #32,5(A0)               ;PRINT A SPACE
        BRA.S   @1

*
*
*
BRK4:   XPMC    MES01
        BRA.L   START

EXIT:   XPMC    MES02                   ;PRINT EXIT MESSAGE
        XEXT                            ;EXIT TO PDOS
*
*
```

```
MES01:  DC.B    $0A,$0D,'>>ESCAPE',0
MES02:  DC.B    $0A,$0D,'>>^C-EXIT',0
MES03:  DC.B    $0A,$0D,'VMEGB-1 Graphics Controller Transfer Test',$0A,$0D,0
MES04:  DC.B    $0A,$0D,'VMEGB-1 Printer Port Transfer Test',$0A,$0D,0
MES05:  DC.B    $0A,$0D,'VMEGB-1 Continuous Characters to GC',$0A,$0D,0
MES06:  DC.B    $0A,$0D,'VMEGB-1 Continous Characters to Printer',$0A,$0D,0
        EVEN
*
        END     START
```


VMEGB1
VMEGB1-000001A
REV 01A
DATE: 15-FEB-1987
SHEET 1 OF 3
U55 74F245
U41 74LS374
U40 74LS244
U39 74LS374
U37 74LS374
U38 74LS374
U46 74LS244
U47 74LS374
U26 74LS175
U27 74LS54
U10 74LS138
U16B 74LS74
U16A 74LS74
U44 74LS54
U45 74259
U28 74LS74
U56 74LS375
U25 74LS138
U24 74LS138
U23 74S85
U9A 74LS112
U9B 74LS112
U49C 7433
U49D 7433
U54 74LS244
U11A 74LS00
U15B 74LS02
U12A 74S260
U14D 74LS02
U15C 74LS02
U14C 74LS02
U12B 74S260
U15A 74LS02
U13B 74S10
U13A 74S10
U13C 74S10
U11B 74LS00
U11 74LS00
U14B 74LS02
U14A 74LS02
U48C 74LS08
SH2/D0-7
/CONT
/CLEAR
/DUMP
/BUSY
SH2-MAP
/PRST
SH3-/BLANK
/BUSY LED
/HOLD LED
SH2-/IRQ
SH2-/CCLK
SH2-R/W
SH2-A0-5
SH2-I/O
SH2-E
SH2-ASELH
/IRQ7
/IRQ6
/IRQ5
/IRQ4
/IRQ3
/IRQ2
/IRQ1
/DTACK
/IACKOUT
/BURST
/WTCSR
/RDCSR
WTVEC
/VECTL
/BEN
BR/W
SELH
R/W
MNIRQ0
MNIRQ1
MNIRQ2
/RES
/BERR
/DTACK
/IACKIN
/WRITE
/SYSRST
/DS0
/IACK
/AS


VMEGB1
VMEGB1-000001A
REV 01A
DATE: 15-FEB-87
SHEET 2 OF 3
SH1/ D0-7
SH1/ A0-5
SH1/ MAP
SH1/ E
SH1/ R/W
SH1/ /IRQ
SH1/ ASELH
SH1/ I/O
SH1/ CCLK
SH3- SDATA
SH3- LOAD
SH3- VS
SH3- DE
SH3- HS
U52 8136
U53 8136
U51 8136
U50 8136
U35 68B09E
U36 27128
U32 68B45P
U20 74153
U21 74153
U22 74153
U19 74153
U29 74LS163
U30 74279
U31 74LS164
U15D 74LS02
U1-8 4164
U18 74LS166
U17 74LS244
U34 74S287
U33 74LS139
U43A 74S74
U42A 74S04
U42F 74S04
U42B 74S04
K2
K3
K4
RN1 470
RN3 10K
RN5 10K
RN6 10K
R20 100K
C40 4.7 MF
R9 1K
C15 15 PF
R12 43
R13 43
R14 43
R15 560
R16 560
C41 .1
XTAL
14.3181 MHZ



	VMEGB1	
E	VMEGB1-000001A	REV 01A
DATE: 15-FEB-1987	SHEET 3 OF 3	