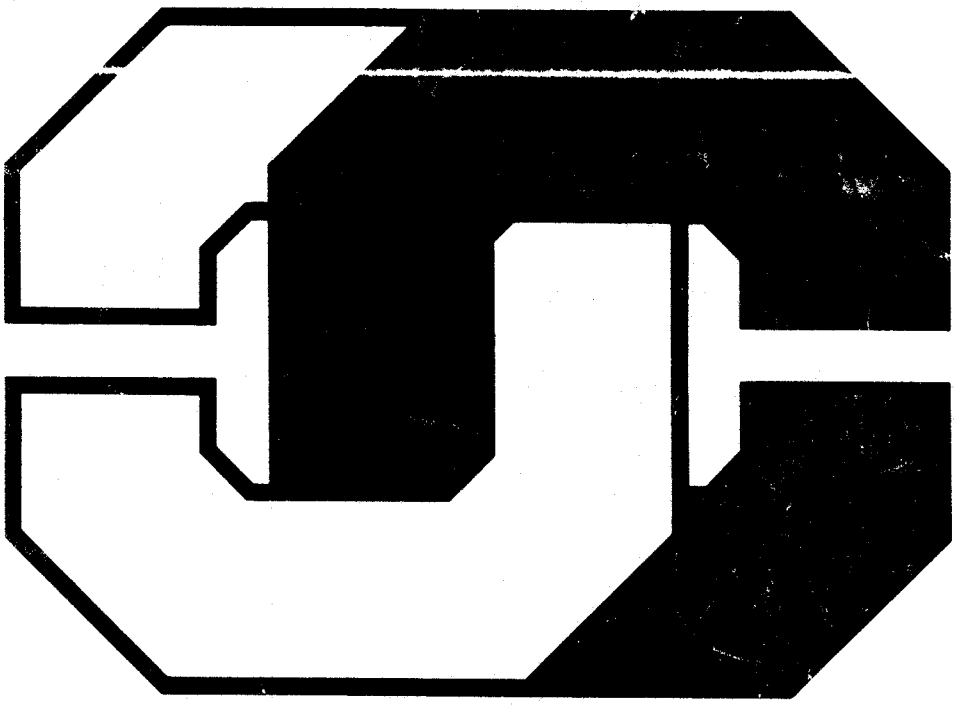


CLIMAX
COMPUTERS LTD

MICROVECTOR 256 OPERATING MANUAL

Climax Computers Ltd
17a Broad Street
South Molton
Devon EX36 3AQ

Telephone (07695) 2314
15/4/83
Issue B

1.0 INTRODUCTION

The Microvector 256 is a high performance colour graphics display interface for NASCOM and GEMINI computers. The circuit is based around the EF9365 graphic display processor with 32K bytes of memory to give a 256 x 256 x 16 colour pixel display. Two versions of the unit are available, the MV256A and MV256B. The B version is identical to the A version except for additional analogue colour monitor outputs. Full operating and programming instructions are given in the following sections. Extensive software is listed in the appendices.

Section 2 covers installation.

Section 3 describes the general hardware functions and options.

Section 4 describes the on-board I/O registers.

Section 5 covers low level device programming.

Sections 6 & 7 describe Assembly and Basic language programming.

Section 8 outlines the general principles of animation using the MV256.

2.0 COMMISSIONING

The MICROVECTOR 256 colour graphic display card consists of a single assembled unit. This should be removed from its polystyrene packing tray and inspected for any obvious mechanical damage. There should be no loose items.

The MV256 plugs into any spare edge connector socket on your systems back plane. If you are using a NASCOM system then you may require NASIO generation. This is selected by forming the appropriate wire link on the dil header near the edge connector. If your edge connector socket does not have a locating keyway (position 72) then you should check that the card is correctly orientated. Positions 1 & 78 are indicated on the PCB. Connection to your colour TV and/or video monitor should be made via suitable coaxial cable. The system can now be powered up.

It may be necessary to adjust the PAL encoder circuit in order to obtain optimum performance with your TV set. Details of how to do this are given in section 3.4.

Your unit has been tested before leaving the factory. If for some reason it fails to work on delivery then it should be returned at once to your supplier.

Please read this manual thoroughly.

3.0 GENERAL HARDWARE DOCUMENTATION

The MV256 unit consists of a single 8" x 8" assembled PCB which is electrically and mechanically NASBUS & 80-BUS compatible. The PCB is manufactured to BS9000 which requires electrical testing and visual inspection to set standards. Both sides of the PCB have solder resist. The track side has been hot air levelled and selectively solder coated. The components are flow soldered.

Figure 3.1 shows the main circuit areas of the MV256. The circuit design is based around the Thomson EF9365 graphic display processor. This IC performs most of the hardware functions, requiring only memory, control logic and a host computer interface to produce a high performance computer graphics system. Portions of the EF9365 data sheet are reproduced in appendix A.

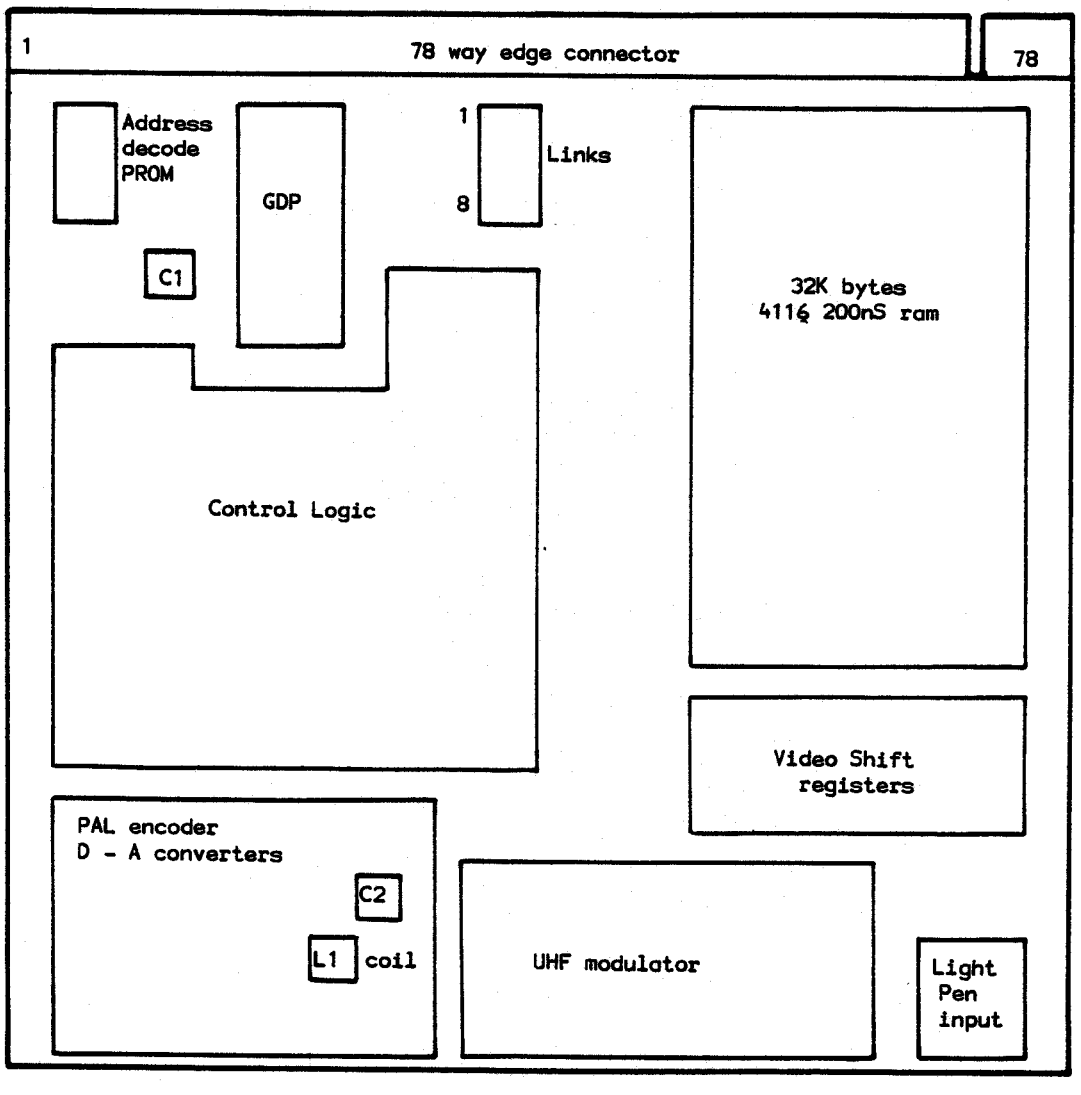


Fig 3.1. MV256 circuit board layout.

3.1 Memory architecture

32K bytes of 4116 ram are used to store the displayed video image. The memory is controlled by the EF9365 which performs all

read/write and refresh operations. Each pixel on the screen is represented by a 4-bit word in the memory. Thus any pixel can be set to 1 of 16 different values.

During display cycles data is read out of the memory 16-bits at a time into 4 shift registers which are clocked at 7MHz. The 4-bit pixel codes appear in parallel form at the shift register outputs which are connected to the D-A converters and PAL encoder.

The EF9365 controls the memory such that normal write operations can only occur during video blanking periods, thus insuring a flicker free display update facility. A special circuit mode enables the video output to be switched off (only sync. remains) and the display memory updated continuously. This enables fast picture generation.

3.2 Bus link options

Two special NASCOM signals are generated by the MV256 circuit, namely /DBDR and /NASIO. The former is permanently connected to the system bus while the latter is selected by forming wire link number 1 on the dil header near the 78-way edge connector. The dil header should be removed from its socket when soldering the link. /NASIO is not selected on supplied units.

3.3 I/O Decoding

Data transfer between the MV256 and host system is via 17 Z80 I/O ports which are mapped onto 13 on-board registers. The maximum data transfer rate is 1MHz which is equivalent to a Z80 system running at 4MHz. In order to insure flexibility and avoid possible I/O allocation clashes port decoding is performed by a bipolar PROM (256 4-bit words). The lower eight address lines are connected to the eight PROM address inputs. /IORQ is connected to the PROM /CE inputs to insure that only valid I/O addresses are decoded. The four PROM outputs are connected to various parts of the MV256 circuit.

The PROM is coded in the following way:

a) Select the 17 I/O ports which are to be assigned to the MV256 Card. It would be usual to select the lowest eight ports to decode a NASCOM 1 or 2 via /NASIO. Note that the MV256 does not require any external decoding.
b) The 256 locations in the PROM are directly mapped to the 256 available I/O ports. The selected locations in the PROM are loaded with the data which will activate the appropriate parts of the MV256 circuit when decoded. Table 1 shows the PROM code content for the MV256 as supplied.

The standard MV256 I/O port allocation ranges from C0H to D0H inclusive. If it is required to change this then a new PROM must be programmed. Although preferable, the 17 I/O ports need not be in consecutive locations but spread out in any configuration over the 256 available ports.

LOCATION	CONTENT (hex)	REGISTER FUNCTION
00	B	NASIO
01	B	NASIO
02	B	NASIO
03	B	NASIO
04	B	NASIO
05	B	NASIO
06	B	NASIO
07	B	NASIO
08	F	Not used
:	:	
BF	F	Not used
C0	6	STATUS/COMMAND
C1	6	CONTROL 1
C2	6	CONTROL 2
C3	6	CHARACTER SIZE
C4	6	Reserved
C5	6	DELTAX
C6	6	Reserved
C7	6	DELTAY
C8	6	X MSBs
C9	6	X LSBs
CA	6	Y MSBs
CB	6	Y LSBs
CC	6	XLP/reserved
CD	6	YLP/reserved
CE	6	Reserved
CF	6	Reserved
D0	C	GETCOL/PENCOL
D1	F	Not used
:	:	
FF	F	Not used

TABLE 1. Address decode PROM code content, as supplied.

Data is read from and written to the ports using the Z80 'in' and 'out' instructions respectively. There is no requirement for wait states or handshaking operations.

3.4 TV Interface

The TV interface consists of a UM1286 UHF modulator and enables the MV256 to be connected to any standard UK 625 line black & white or colour TV set. Connect the MV256 board to the TV set using 75ohm aerial cable. The UHF modulator accepts a standard phono type plug. Make sure that the copper screen of the cable is firmly connected to both the phono plug and the coaxial plug at the TV aerial input.

3.4.1 Tuning your TV

The modulator is preset to channel E36 which is commonly used for TV games and VTRs. When first powered up the MV256 will generate a black or 'cleared' display. Turn up the contrast control on your set. Set the volume control to the normal listening level. Select a spare channel and tune into the signal. When properly tuned in the display should be free from break up and random noise. The loud speaker should be quiet. If loud speaker hiss

is present then use the yellow pot marked 'fine tune' next to the modulator to retune the sound subcarrier to your set. DO NOT adjust any of the cores inside the modulator.

The contrast control should now be turned down until the display just becomes completely black under normal lighting conditions.

NOTE: If your host computer is using the system bus to 'poll' the status of a circuit card (the IVC for instance) at this time then lines may appear to traverse the screen. These can be eliminated by preventing your computer from polling (by running a programme for instance) or by tuning them out. The action of polling over the system bus causes video interference.

3.4.2 The PAL encoder

The encoding of the 4-bit pixel codes into PAL colour video is performed by a TEA1002 (Mullard) PAL encoder and video summer IC. Video synchronising and blanking signals are generated by the EF9365. The TEA1002 will get hot during normal operation since it typically dissipates 840mW.

Chrominance filtering is used to reduce unwanted patterning on the display. Since the filter causes a delay in the chorminance (colour) signal a delay line (Green box) is used to delay the luminance (brightness) signal by the appropriate amount. This insures that the colour and brightness information register correctly on the TV screen. The output of the TEA1002 is AC coupled to the UHF modulator.

The colour subcarrier oscillator is 'free running' i.e. is not locked to the video sync. oscillator. This causes fringing effects known as 'dot crawl' to occur at the boundries between different colours. In order to minimise the appearence of the dot crawl the colour subcarrier and sync. oscillators must be set such that their beat frequency is greater than 50Hz. When this is achieved motion at colour boundaries disappears. The dot crawl is still there but undetectible by the eye.

3.4.3 Setting up the PAL encoder

There are six controls which will affect the quality of the display on a colour TV set.

(i) TV Brightness control
(ii) TV Contrast control
(iii) TV Colour control
(iv) MV256 Cap. trimmer 1 (green)
(v) MV256 Cap. trimmer 2 (green)
(vi) MV256 Adjustable inductor (pink)

Refer to figure 3.1 for the relevant MV256 component positions. Allow 5 - 10 minutes for the two MV256 crystal oscillators to stabilise and then proceed:

(i) Leave the contrast control set as detailed in section 3.4.
(ii) Set the brightness control to the normal viewing level.

(iii) Set the colour control to half of its full swing.
(iv) Run the following BASIC programme:

```
 10 OUT(192),7                    ;RESET MV256
 20 W=SIN(60)*SIN(60)*SIN(60)     ;WAIT > 20mS
 30 OUT(193),3                    ;SELECT PEN UP
 40 OUT(201),15                   ;REPOSITION GRAPHIC
 50 OUT(203),90                   ;CURSOR
 60 OUT(195),128                  ;SELECT GRAPHIC BLOCK SIZE
 70 OUT(208),2                    ;SELECT GREEN PEN COLOUR
 80 OUT(192),12                   ;SET WHOLE SCREEN TO GREEN
 90 W=SIN(60)*SIN(60)*SIN(60)     ;WAIT > 20mS
100 FOR C=1 TO 7                  ;NEXT PEN COLOUR
110 OUT(208),C
120 OUT(192),11                   ;DRAW GRAPHIC BLOCK
130 NEXT C                        ;NEXT BLOCK
140 END                           ;FINISHED
```

The programme will generate a simple test picture.

(v) If the picture is black and white then adjust the MV256 cap. trimmer 2 until a colour display results. Now tweek the trimmer until the red, green & blue colours appear most saturated. Alternatively if horizontal lines are faintly visible in the yellow colour then tweek the trimmer until they are no longer distinguishable. The TV set has now locked on to the MV256 colour subcarrier oscillator.

(vi) Adjust the MV256 cap. trimmer 1 until the dot crawl at the colour boundaries becomes least noticible. This should be at the maximum beat frequency and not for a stationary beat pattern.

(vii) Adjust the TV set colour control to obtain the best colours without introducing noticible edge effects.

(viii) The MV256 adjustable coil should not need to be reset. However, if patterning occurs on the display (particularly in the blue colour) then the coil should be tweeked to reduce this effect. Take care not to wind the core to its maximum 'in' position since it may not be possible to get it out again.

After the initial setup process the MV256 adjustable coil and cap. trimmer 1 should need no further adjustment. The MV256 cap trimmer 2 may need tweeking from time to time. Some colour boundary effects will always be present.

NOTE: If you adjust the MV256 cap. trimmer 1 through its full range then at one point the oscillator frequency will be out of design spec. This will cause a disruption to normal display operation. No permanent damage can be done but the MV256 display memory may become corrupted.

3.4.4 Using sound

The MV256 UHF modulator enables sound generated externally to be fed through a TV set loudspeaker. The audio input is via a standard 3.5mm jack next to the modulator. A 'fine tune' pot enables the sound subcarrier to be tuned to individual TV sets. The audio input signal amplitude is 5v pk-pk maximum.

3.5 Composite B/W video output

This output enables the MV256 to be connected to most types of B/W video monitor. The signal is 1v pk-pk and has a 75ohm output impedance. Each of the 16 possible pixel codes generates a different grey level on the screen, ranging from black to white. The variation is linear.

The video signal is made available through a miniture coaxial type socket labled 'comp'. Extreme care should be exercised when inserting and removing miniture coaxial plugs.

3.6 RGB Video outputs (MV256B only)

The red, green & blue colour outputs enable the MV256B to be connected to most standard colour monitors. The signal levels are 1v pk-pk with a 75ohm output impedance. Composite sync. is mixed in on each signal. The RGB outputs should be connected to the corresponding inputs on the colour monitor. Each of the 16 possible pixel codes generates a different colour. The PAL and RGB colours are similar.

The RGB video signals are made available through three miniture coaxial type sockets labled 'red', 'green' & 'blue' respectively.

Suitable colour monitors.

The monitor must have analogue inputs, preferably with a 75ohm input impedance. TTL inputs are unsuitable. If a separate SYNC. signal is required then connect any ONE of the three colour signals to its respective colour input AND the sync. input.

3.7 Upgrading the MV256A to MV256B

The MV256A upgrade kit adds a colour monitor output capability to the MV256A circuit. There are no other differences between the MV256A & B circuits. Full instructions are supplied with the kit.

3.8 Light pen input

The light pen input consists of a single 5-pin 180^{o} DIN type socket labled 'light pen'. The socket is wired to accept the ARFON/TORCH type light pen with a -ve going STROBE output. No hardware modifications are required when using the light pen.

Pin function

N/C	1	- +12v
White	2	- 0v
Blue	3	- SWITCH/ENABLE (not used)
Red	4	- +5v
Green	5	- STROBE

Yellow = LED (N/C)

3.9 Interrupts

The interrupt signal generated by the MV256 is non-vectored. Wire link options enable the signal to be connected to one or more of five posible 80-BUS signal lines.

Wire link number -	Connects to this signal line	
2 :	/NMISW	
3 :	INT 0	Interrupt
4 :	INT 1	request
5 :	INT 2	lines
6 :	INT 3	

The wire links are formed on the dil header near the 78-way edge connector. The outputs are low power schottky TTL.

3.10 Power requirements

The MV256 requires three power supplies, all of which are on the 80-BUS and NASBUS.

+12volts @ 500mA typ.
+5volts @ 750mA typ.
-5volts neglible
and ground.

3.11 Bus signals used by the MV256

Standard:	A0-A7	address lines
	D0-D7	data lines
	/IQRQ	
	/RD	
	/WR	
	/DBDR	generated for nascom users
	/RESET	
Optional:	/NASIO	
	/NMISW	
	INT 0	
	INT 1	
	INT 2	
	INT 3	

4.0 REGISTER Description

4.1 I/O allocation

The MV256 is programmable via 13 on-board registers occupying 17 consecutive I/O ports within the host computers peripheral addressing space. Eleven registers are internal to the EF9365 while the GETCOL and PENCOL registers are separate devices. The I/O address of each register is given in Table 2.

Register I/O address		Register Function		Number of
Hex	Decimal	Read	Write	Bits
C0	192	STATUS	COMMAND	8
C1	193	CONTROL 1		7
C2	194	CONTROL 2		4
C3	195	CHARACTER SIZE		8
C4	196	Reserved		-
C5	197	DELTAX		8
C6	198	Reserved		-
C7	199	DELTAY		8
C8	200	X MSBs		8
C9	201	X LSBs		8
CA	202	Y MSBs		4
CB	203	Y LSBs		8
CC	204	XLP	Reserved	7
CD	205	YLP	Reserved	8
CE	206	Reserved		-
CF	207	Reserved		-
D0	208	GETCOL	PENCOL	4

TABLE 2. Register I/O addresses.

There are four types of register

1) COMMAND
2) STATUS
3) CONTROL
4) PARAMETER

4.2 COMMAND register C0H (192)D

There is a single 8-bit write only COMMAND register. Each write operation into this register causes a command to be executed by the MV256, upon completion of the time necessary to decode the command. Several types of command are available:

Vector plotting
Character plotting
Screen scanning
Screen erase
Screen memory read
Light pen sequence
Indirect modification of other MV256 registers.

The COMMAND register should not be modified when a command is currently executing.

4.3 STATUS register C0H (192)D

The single 8-bit read only STATUS register holds the current status of the MV256 circuit. The following status information is available at all times:

Light pen status
Vertical video blanking state
Command progress
Graphic cursor position w.r.t. display window
Interrupt status

Register bit functions

Bit 0: **0** = Light pen sequence underway
1 = Light pen circuit not in use

Bit 1: **0** = Pixel display
1 = Vertical video blanking (Top & bottom of video display)

Bit 2: **0** = Command currently executing
1 = Circuit ready for a new command

Bit 3: **0** = Graphic cursor within pixel display area
1 = Graphic cursor out of pixel display area

Bit 4: **1** = Interrupt generated by the completion of a light pen sequence.

Bit 5: **1** = Interrupt generated by the start of a vertical video blanking period (bottom of the pixel display)

Bit 6: **1** = Interrupt generated by the completion of execution of a command.

Bit 7: **1** = Interrupt has been generated

Bits 4,5,6 & 7 are reset by reading the STATUS register

4.4 CONTROL registers

There are two read-write control registers.

CONTROL 1 register C1 H (193)D

The 7-bit CONTROL 1 register enables the general MV256 circuit operating mode to be specified.

Register bit functions

Bit 0: **0** = Pen/eraser up selection
1 = Pen/eraser down selection

Bit 1: **0** = Eraser selection
1 = Pen selection

Bit 2: **0** = Normal display operation
1 = Pixel display switched off for high speed write

Bit 3: **0** = 4096 x 4096 logical picture area
1 = 256 x 256 logical picture area (cyclic screen)

Bit 4: **0** = Inhibit interrupt generated by the completion of a light pen sequence
1 = Enable interrupt generated by the completion of a light pen sequence

Bit 5: **0** = Inhibit interrupt generated by the start of vertical video blanking
1 = Enable interrupt generated by the start of vertical video blanking

Bit 6: **0** = Inhibit interrupt generated when the MV256 is ready for a new command
1 = Enable interrupt generated when the MV256 is ready for a new command

Bit 7: is not used and always read as '0'

CONTROL 2 register C2 H (194)D

The 4-bit CONTROL 2 register selects the type of vector or character to be plotted.

Register bit functions

Bit 0: These two bits define 4 types of vector.

Bit 1:

BIT 1	BIT 0	VECTOR TYPE
0	0	Continuous
0	1	Dotted
1	0	Dashed
1	1	Dotted-dashed

Bit 2: **0** = Straight characters
1 = Tilted characters

Bit 3: **0** = Write characters along the horizontal axis
1 = Write characters along the vertical axis

Bits 4,5,6 & 7 are not used and always read as '0000'

4.5 PARAMETER registers

There are nine parameter registers.

CHARACTER SIZE register C3 H (195)D

This 8-bit read-write register indicates the X,Y scaling factors used for character and symbol generation

MSB P Q LSB

The CHARACTER SIZE register

Each character or symbol can be increased in size by a factor of P(X), Q(Y) during the plotting process. These factors are independant integers in the range 0 - 15.

DELTAX and DELTAY registers
C5H (197) C7H (199)
These two 8-bit read-write registers indicate the X,Y projections of the next vector to be plotted. Such values are unsigned integers in the range 0 - 255.

X and Y registers
(200,201) C8,C9H CA,CBH (202,203)
These 12-bit read-write registers indicate the current position of the graphic cursor. The 2 x 12-bit write address covers a 4096 x 4096 pixel addressing range. Only the 8 LSBs correspond to displayed pixels since the display resolution is 256 x 256. The 4 MSBs are used to inhibit plotting where the actual display is regarded as a window within a 4096 x 4096 space. The above feature along with the relative mode description of pixels make it possible to solve the great majority of edge cut off problems (picture clipping).

In the cyclic screen mode the 4 MSBs are ignored.

XLP and YLP registers
CC (204) CD (205)
Both registers are read-only and of 7 and 8-bits respectively. Upon the completion of a light pen sequence they contain the screen address of the pixel detected by the light pen.

GETCOL register
D0 (208)
Upon completion of the MV256 'Get pixel colour' command this 4-bit read-only register contains the 4-bit code of the pixel at the current graphic cursor position.

PENCOL register
D0 (208)
This 4-bit write-only register is used to select the current pen plotting colour or grey level.

RESERVED registers

These have no function and are always read as FFH.

5.0 PROGRAMMING

The MV256 is programmed by writing data to the appropriate system I/O ports. Similarly the MV256 can be examined by reading those ports. There are no wait states or special handshaking operations to perform.

5.1 System initialisation

When first powered up the MV256 will initialise into a predefined state:

(i) Clear screen
(ii) Pen up selection
(iii) Black pen colour selection
(iv) Continuous line style selection
(v) Normal character selection
(vi) Horizontal character mode
(vii) Minimum character size (P=Q=1)
(viii) Normal display mode
(ix) All parameter registers reset

The STATUS register should be tested to see when the MV256 is ready for its first command.

5.2 Commands and parameters

Parameters and control codes are passed to and from the MV256 as 8-bit words. Unused bits are ignored. Commands are single byte codes which are executed when ever written into the COMMAND register. Executing commands use parameters previously sent to the MV256. Sending a new command before the previous one has finished executing will cause an unreported error, possibly corrupting both command sequences. Similarly, parameter and control registers used by a particular command should not be externally modified until the command has finished executing.

Register modification by commands

During the execution of some commands, parameter and control registers are modified in a predetermined manner. These modifications are described in the relavent sections.

5.3 System status cf (192)

The STATUS register contains the current operating status of the MV256. Reading this register will not affect the operation of the MV256 circuit. The required status information is extracted from the status byte by performing bit tests. See section 4.3

5.4 Pen/Eraser operation.

All drawing processes performed by the MV256 use the PEN or ERASER to modify the colour of displayed pixels.

5.4.1 Action

The pen is used to change the colour of displayed pixels i.e. to draw out figures etc. Using the pen, pixels can be set to 1 of 16 different states.
The eraser is used to delete displayed pixels i.e. to reset them to black (code 00H). Figures are deleted by using the eraser or the black pen colour.

5.4.2 Control

The pen/eraser function can be inhibited by selecting the UP condition. To activate the pen/eraser the DOWN condition must be selected.

The pen/eraser selection and the up/down mode is determined by the two LSBs of the CONTROL 1 register. Four MV256 commands are available for this purpose.

COMMAND	FUNCTION
00H	**Pen** selection (set bit 1 of CONTROL 1)
01H	**Eraser** selection (reset bit 1 of CONTROL 1)
10H	Pen/eraser **down** (set bit 0 of CONTROL 1)
11H	Pen/eraser **up** (reset bit 0 of CONTROL 1)

Alternatively, the CONTROL 1 register can be modified directly to achieve the same results.

5.4.3 Selecting a pen colour

The current pen colour is defined by the 4-bit binary code stored in the PENCOL register. The pen colour can be changed by writing the new pen colour code into this register, see table 3.

HUE	SATURATION 0.5	SATURATION 1.0
BLACK	00	08**
BLUE	01	09
GREEN	02	0A
CYAN	03	0B
RED	04	0C
MAGENTA	05	0D
YELLOW	06	0E
WHITE	07*	0F

TABLE 3. The 16 hex colour codes.

NOTE: * Appears GREY on RGB displays
and WHITE on PAL displays
** Appears GREY on RGB displays
and BLACK on PAL displays

When displayed on a black and white video monitor the colour codes produce grey levels of increasing brightness where code 00H = BLACK and 0FH = WHITE. The variation is linear.

5.5 Vector generation

The MV256 circuit has a hardwired processor capable of computing all the pixels which form the approximation of a straight line segment. The reader should refer to the extract from the EF9365 data sheet reproduced in appendix A for further information on the operation of this processor.

Vector description

All vectors are described by their X,Y starting co-ordinates and projections on the X and Y axes. Both absolute and relative vector plotting are possible.

Absolute: The start point and projections are specified for each new vector.

Relative: The start point of the next vector corresponds to the end point of the previous vector. Only the new vector projections need be specified.

The vector direction is described by a three bit code, according to figure 5.1.

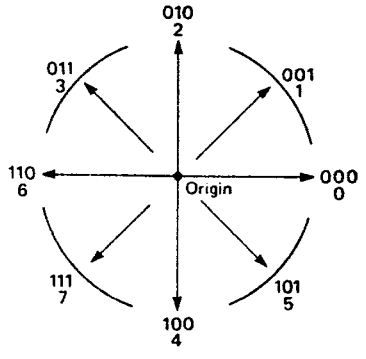


Fig 5.1. The three bit vector direction code.

The vector direction code is used to form part of the vector plot command. See later.

Two types of vector are possible:

(i) Long vectors
(ii) Short vectors

5.5.1 Long vectors

The starting point co-ordinates are defined by the graphic cursor X,Y position prior to the plotting operation. Projections on to the X & Y axes are defined as absolute values in the DELTAX and DELTAY registers, with the signs in the command code used to initiate the plotting process. During vector plotting the X,Y co-ordinates of the graphic cursor are incremented or decremented. On completion of vector plotting the graphic cursor is positioned at the end of the vector. This facility enables relative vector descriptions. Vectors with projections greater than 255 pixels must be divided into segments.

Long vectors may be plotted using one of four line styles, selected by the two LSBs of the CONTROL 2 register. See table 4.

BIT 1	BIT 0	VECTOR LINE STYLE	
0	0	Continuous	All pixels on
0	1	Dotted	2 pixels on - 2 pixels off
1	0	Dashed	4 pixels on - 4 pixels off
1	1	Dot-Dashed	10 pixels on - 2 pixels off
			-2 pixels on - 2 pixels off

TABLE 4. Using the two LSBs of the CONTROL 2 register to select the vector line style.

The vector plotting speed is the same for all line styles.

A long vector plotting sequence is initiated by writing the appropriate code to the COMMAND register. 16 commands are available.

Commands using both DELTAX and DELTAY

These four commands can be used to plot any vector. The 8-bit command code is constructed according to figure 5.2.

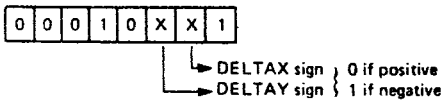


Fig 5.2

The vector direction is given by the three LSBs of the command code and corresponds to the diagonal lines of figure 5.1.

Commands ignoring either DELTAX or DELTAY

These four special commands are useful when drawing small rectangular boxes. The 8-bit command code is constructed according to figure 5.3.

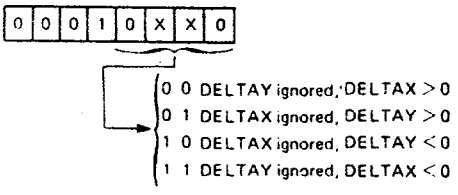


Fig 5.3

The vector direction is given by the three LSBs of the command code and corresponds to the horizontal and vertical directions of figure 5.1. The ignored register is considered to have a value of zero.

Commands which ignore the smaller of DELTAX and DELTAY

A further eight special commands allows the smallest of the two DELTAX and DELTAY registers to be ignored by considering it to be equal to the larger one. The 8-bit command code is constructed according to figure 5.4

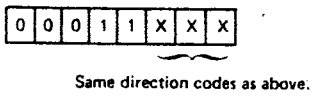


Fig 5.4

The vector direction is given by the three LSBs of the command code and corresponds to those given in figure 5.1. Vectors drawn using these commands will be parallel to the axes or diagonals.

Long vector drawing procedure:

a) Select line style
b) Select pen/eraser
c) Select pen colour if pen in use
d) Move graphic cursor to vector start (absolute vectors only)
e) Set DELTAX & DELTAY, to required values
f) Issue 'draw long vector' command to MV256

Steps a, b & c may be ignored if these parameters are the same as for previous vectors. Step d can be ignored for relative vectors.

5.5.2 Short vectors

The starting point co-ordinates are defined by the graphic cursor X,Y position prior to the plotting operation. Vector projections on to the axes are defined as absolute values within the command code itself, i.e. DELTAX and DELTAY are not used. Each projection can be of 0,1,2 or 3 pixels. The vector plotting operation is similar to that of the long vectors. The continuous line style should be used for short vectors.

The vector plotting sequence is initiated by writing the appropriate code to the COMMAND register. The 8-bit command code is constructed according to figure 5.5.

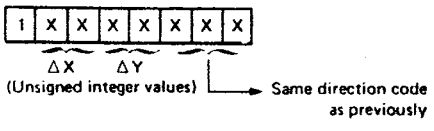


Fig 5.5 Short vector command code.

Both absolute and relative short vectors can be drawn.

Short vector drawing procedure:

a) Select continuous line style
b) Select pen/eraser
c) Select pen colour if pen in use
d) Move graphic cursor to vector start (absolute vectors only)
e) Issue 'draw short vector' command to MV256

Steps a,b and c may be ignored if these parameters are the same as for previous vectors. Step d can be ignored for relative vectors.

5.5.3 Vector drawing summary

Vectors are computed so that they may be erased by redrawing with the eraser or a different pen colour. Vectors can be drawn outside of the display window but within the 4096 x 4096 logical draw space. The drawing time remains the same where ever the vector occurs. The DELTAX, DELTAY, PENCOL and CONTROL register contents are not altered by the vector drawing process and can be repeatedly used.

It is plainly evident that relative vector drawing requires less information than absolute and that short vectors require less information than long vectors. The host computer graphics programmes should be organised to take advantage of this in order to reduce the time required to initialise a new vector drawing sequence and the storage space needed for vector descriptions. It is reasonable to assume that long vectors will be used for drawing geometrical shapes etc. and short vectors for fine detail (space invader shapes etc.) and approximations of curved surfaces.

5.6 Character generation

In addition to the vector processor the MV256 has a hardwired character generator. Its operation is similar to the vector processor and the reader should refer to appendix A for further details.

Character description

The character generator receives its parameters from the CHARACTER SIZE, CONTROL 2, PENCOL and COMMAND registers. The plotted characters are selected, according to the command code, out of 98 matrices (97 5x8 and 1 4x4 matrices) defined by an internal ROM. Two scaling factors may be applied to the characters. Characters can be tilted for italics and written on the horizontal or vertical axis. Both absolute and relative character plotting are possible.

Absolute: The start point is specified for each new character.

Relative: The start point of the next character corresponds to the end point of the previous character with a one pixel spacing. This enables lines of text to be written.

The ASCII code of the character to be plotted forms part of the character plot command. See later.

5.6.1 Character set

The complete 96 ASCII character set is shown in figure 5.6. Programmable characters are not possible but similar results can be achieved by using the short vector commands.

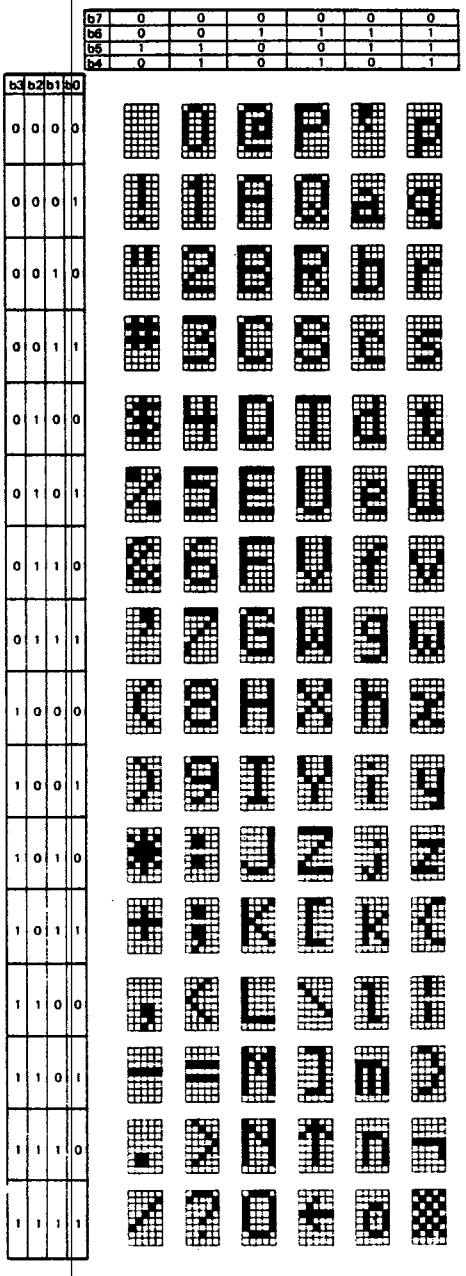


Fig 5.6. ASCII character set.

The start point co-ordinates are defined by the graphic cursor X,Y position prior to the plotting operation. The character plotting process is initiated by writing the character ASCII code (bit 7 reset) into the COMMAND register. During character plotting the graphic cursor X,Y co-ordinates are incremented or decremented. On completion of character plotting the graphic cursor is positioned for writing a further character next to the previous one with a one pixel spacing. Thus characters may be plotted in absolute terms (graphic cursor positioned each time) or relative terms (one pixel spacing).

5.6.2 Programmable character sizes

Each pixel in the 5 x 8 basic matrix may be replaced by a P x Q sized pixel block where:

P : X scaling factor
Q : Y scaling factor

The character size becomes 5P x 8Q. P and Q are independant and may take values from 1 through 16. They are defined by the CHARACTER SIZE register. Each value is encoded using 4 bits, value 16 being encoded as '0000', where P = 4 MSBs and Q = 4 LSBs. There are 256 different character sizes. Refer to section 4.5 for further details.

The one pixel spacing between characters is also scaled according to the X scaling factor. Character scaling is performed BEFORE character reorientation.

5.6.3 Programmable character orientations

Characters may be plotted on the horizontal or vertical axes using normal or italic styles. The selection is made using bits 2 and 3 of the CONTROL 2 register. See table 5.

BIT 3	BIT 2	CHARACTER TYPE
0	0	Straight character on horizontal axes
0	1	Italic character on horizontal axes
1	0	Straight character on vertical axis
1	1	Italic character on vertical axis

TABLE 5. Using the two MSBs of the CONTROL 2 register to select the character type.

Character drawing procedure:

a) Select the character type
b) Set P & Q scaling factors
c) Select pen/eraser
d) Select pen colour if pen in use
e) Move graphic cursor to character start
f) Issue ASCII code command to MV256

Steps a,b,c and d can be ignored if these parameters are the same as for previous characters. Step e can be ignored for relative characters.

5.6.4 Character drawing summary

Characters are computed so that they may be erased by redrawing with the eraser or a different pen colour. Characters can be drawn outside of the display window but inside of the 4096 x 4096 logical draw space. The draw time remains the same where ever the character occurs. The CHARACTER SIZE, CONTROL 2 and PENCOL register contents are not altered by the character drawing process and can be repeatedly used.

The spacing between tilted and enlarged characters is always arranged so that subsequent characters can be plotted alongside to create a line of text.

5.6.5 Graphic blocks

Two special symbols called graphic blocks are also defined by the internal character ROM. Both graphic blocks are plotted in the same fashion as characters, using parameters from the CHARACTER SIZE, CONTROL 2, PENCOL and COMMAND registers.

Command 0BH

Plots a 4P x 4Q graphic block. Upon completion the graphic cursor is positioned without spacing for the next symbol. Such a block makes it possible to fill uniform areas of the display.

Command 0AH

Plots a 5P x 8Q graphic block. Upon completion the graphic cursor is positioned with spacing for the next symbol. Such a block can be used to erase characters.

5.7 Screen operations

Two special commands enable the entire display to be cleared or set to a particular colour. Both commands cause the video output to be blanked for one video frame. This will cause a flash effect when the display is set to a colour other than black.

5.7.1 Clear screen - Command 04H

This command will reset every pixel in the display to the black level (code 00H). None of the MV256 parameter registers are affected by this command.

Clear screen procedure:

a) Issue command 04H to the MV256

5.7.2 Set screen - Command 0CH

This command will set every pixel in the display to the colour defined by the PENCOL register. As before none of the MV256 parameter registers are affected by this command.

Set screen procedure:

a) Select pen or eraser
b) Select pen colour if pen in use
c) Issue command 0CH to the MV256

Steps a and b can be ignored if already in the required state.

5.8 Get pixel colour operation. Command 0FH

This command will read the colour code of any pixel into the GETCOL register. The position of the pixel is defined by the graphic cursor X,Y co-ordinates prior to the read operation. Only the GETCOL register is affected by this command. The STATUS

register should be examined to determine when the pixel colour is available (ready for new command test). Points outside the display window will be black.

Get colour procedure:

a) Move graphic cursor to required pixel
b) Issue command 0FH to MV256

5.9 Graphic cursor control

The current graphic cursor position is given by the X and Y registers. In the normal display mode these cover a logical space of 4096 x 4096. In the cyclic display mode the logical space will be 256 x 256. If the graphic cursor is moved outside of its logical space then it will reappear at the opposite side to which it exited.

5.9.1 Moving the graphic cursor

The graphic cursor is moved by simply writing the new co-ordinates to the X and Y registers.

5.9.2 Commands which affect the graphic cursor

Plotting commands

These commands, such as vector and character plotting, will move the graphic cursor during the plotting operation. At the end of the plot, the graphic cursor will always be positioned at some known position (see previous sections and appendix A).

Graphic cursor reset commands

There are four commands which can be used to reset the graphic cursor X,Y co-ordinates to 0. They are:

COMMAND	:	FUNCTION
05H	:	Reset X and Y to 0
06H	:	Reset X and Y to 0 and clear screen
0DH	:	Reset X to 0
0EH	:	Reset Y to 0

None of the other parameter registers are affected by these commands.

5.10 Display reset. Command 07H

This command resets the entire MV256 circuit.

5.11 Display operating mode.

5.11.1 Logical Draw space size

The logical draw space size can be either 4096 x 4096 or 256 x 256. The larger size permits the graphic cursor to leave the display window and is normally used to insure automatic picture clipping. When the smaller space is used the graphic cursor will always be within the display window. Plotting operations which extend beyond the display window edges will appear at the opposite edge to that exited from.

The required space is selected by bit 3 of the CONTROL 1 register.

BIT 3	SIZE
0	4096 x 4096 (Normal)
1	256 x 256 (Cyclic screen)

5.11.2 High speed write

During normal display operation pixels can only be modified during video blanking periods (the black boarder around the pixel display). This limits the number of pixels which can be modified during any one video frame (20mS) to about 20000.

In the high speed write mode the pixel display is switched off. Pixel modification can now occur throughout the video frame (except during memory refresh). When a picture has been drawn the pixel display is switched back on again. In the off mode video synchronising pulses are still generated.

The on/off mode is selected by bit 2 of the CONTROL 1 register.

BIT 2	MODE
0	ON (Normal)
1	OFF (High speed write)

5.12 Light pen operation.

The MV256 has a light pen facility. This is fully documented in the light pen manual (available separately). A brief description is given here for completness.

A light pen enables the user to locate a position on the display on an interactive basis, i.e. by just positioning the pen at the desired point. The scanning electron beam is detected by the pens photodiode as it passes. In order to function the beam must correspond to a lighted pixel. The pen generates a clock pulse at this time which is used to sample the current display memory scan address into the XLP and YLP registers. These can be read by the host computer, corrected for offset errors and the pen position determined.

A light pen operating sequence lasts for one video frame (20mS). If no light pen clock pulse is generated during this time then this is indicated by a status bit in the LP registers, other wise these registers contain the pen co-ordinates. The light pen operating status is given by bit 0 of the STATUS register.

The light pen operating sequence is initiated by writing the command 09H to the COMMAND register. Only the XLP and YLP parameter registers are affected by this command.

5.13 Interrupt operation

Interupts in the host computer system allow an external device to interrupt the execution of a programme, forcing a branch to a specific software subroutine. The subroutine is then executed, after which the original programme is re-entered at the branch point.

Three types of non-vectored interrupts are generated by the MV256

a) MV256 ready for a new command
b) Vertical video blanking just started
c) Light pen sequence just finished

Interrupts are made available at a single output pin which may be connected to one or more of 5 possible 80-Bus signal lines. See section 3.11.

The STATUS register must be examined in order to determine which interrupt was generated. Each type of interrupt can be enabled or inhibited. See section 4.4.

5.14 Command execution time

5.14.1 Vector generation

The MV256 plots any vector at a rate of 1 pixel per 571nS clock period (1.75MHz). If the plotting process is interrupted by a display or refresh period, then the plot is postponed until the end of that period. The MV256 will take approximately 4 clock periods to decode the command.

Example: Plotting time for a vector with X,Y projections of 20, 30 respectively.

Vector length = $(20^2 + 30^2)^{1/2}$
= 36 pixels
Decode time = 4 x 571 nS
Plot time = 36 x 571 nS

Total time = 23uS

This assumes no display or refresh interruptions. The parameter programming time should be added. All vectors are plotted at the same rate regardless of line style.

5.14.2 Character generation.

The MV256 plots any character at a rate of 1 pixel per 571ns clock period. If the plotting process is interrupted by a display or refresh period then the plot is postponed until the end of that period.

Example: Plotting time for a character with P,Q scaling factors of 5,9 respectively.

Character size	=	6P x 8Q pixels (including character spacing)
	=	30 x 72 pixels
	=	2160 pixels
Decode time	=	4 x 571 nS
Plot time	=	2160 x 571 nS
Total time	=	1.23mS

This assumes no display or refresh interruptions. The plotting of large characters will almost certainly be interrupted. The parameter programming time should be added. All characters are plotted at the same rate.

NOTE: The spacing between characters is computed by the MV256 and must be taken into account.

5.14.3 Screen operations.

These include set screen and clear screen. Both are performed by the video display controller circuit rather than the vector or character processors.

Screen scan time	=	256 x 64 uS
	=	16.4mS

Since no other command can be executed during the screen scanning process, the entire video scan line period (64uS) must be used in the calculation rather than the line display time of 36.5uS. The screen scan process starts at the beginning of a video frame, so the time to complete the current video frame must be added to the screen scan time.

5.14.4 Get pixel colour operation.

Should the MV256 be engaged in a display or refresh period, then this operation is postponed until the end of that period. The command decode time is four clock periods.

Maximum waiting time	=	(64 + 4) x 571 nS
	=	39uS
Minimum waiting time	=	4 x 571 nS
	=	2.3uS

NOTE: There are 64 memory cycles per display line. Each cycle takes 571 nS.

5.14.5 Other commands.

Most other commands only require 2.3uS in which to perform their operation.

5.15 MV256 command summary

b7 b6 b5 b4 / b3 b2 b1 b0		0 0 0 0	0 0 0 1	0 0 1 0	0 0 1 1	0 1 0 0	0 1 0 1	0 1 1 0	0 1 1 1	1 0 0 0	1 0 0 1	1 0 1 0	1 0 1 1	1 1 0 0	1 1 0 1	1 1 1 0	1 1 1 1
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0 0 0 0	0	Set bit 1 of CTRL1 : Pen selection	Vector generation (for b2, b1, b0 see small vector definition)	SPACE	0	@	P	`	p								
0 0 0 1	1	Clear bit 1 of CTRL 1 : Eraser selection		!	1	A	Q	a	q								
0 0 1 0	2	Set bit 0 of CTRL1 : Pen/Eraser down selection		"	2	B	R	b	r								
0 0 1 1	3	Clear bit 0 of CTRL 1 : Pen/Eraser up selection		#	3	C	S	c	s								
0 1 0 0	4	Clear screen		$	4	D	T	d	t								
0 1 0 1	5	X and Y registers reset to 0		%	5	E	U	e	u								
0 1 1 0	6	X and Y reset to 0 and clear screen		&	6	F	V	f	v								
0 1 1 1	7	Clear screen, set CSIZE to code "minsize" All other registers reset to 0 (except XLP, YLP)		'	7	G	W	g	w								
1 0 0 0	8	Light-pen initialization (WHITE forced low)	Special direction vectors (for b2, b1, b0 see small vector definition)	(	8	H	X	h	x								
1 0 0 1	9	Light-pen initialization		)	9	I	Y	i	y								
1 0 1 0	A	5 x 8 block drawing (size according to CSIZE)		*	:	J	Z	j	z								
1 0 1 1	B	4 x 4 block drawing (size according to CSIZE)		+	;	K	[	k	{								
1 1 0 0	C	Screen scanning : Pen or Eraser as defined by CTRL1		,	<	L	\	l	¦								
1 1 0 1	D	X register reset to 0		-	=	M	]	m	}								
1 1 1 0	E	Y register reset to 0		.	>	N	↑	n	~								
1 1 1 1	F	Direct image memory access request for the next free cycle.		/	?	O	←	o	▒								

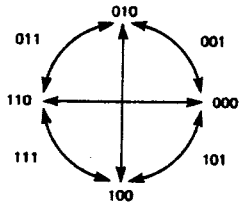
SMALL VECTOR DEFINITION :

b7	b6 b5	b4 b3	b2 b1 b0
1	|ΔX|	|ΔY|	Direction

Dimension

ΔX or ΔY		Vector length
0	0	0 step
0	1	1 step
1	0	2 steps
1	1	3 steps

Direction



6.0 ASSEMBLY LANGUAGE PROGRAMMING

In order to facilitate future programming of the MV256 a range of graphic primitive subroutines have been provided as Z80 source listings in appendix B. It is not recommended that they be typed in by hand. The source and object code are available on cassette or diskette.

6.1 Graphic primitive subroutines

These subroutines perform most of the low level functions required by any graphics programme. They are designed to be called directly from Microsofts 5.1 CP/M version of MBASIC.

6.1.2 Function operation

The graphic primitives are actually executed by the host computer system, usually by being called from a high level language. Most of the graphic primitives have arguments and take the form:

NAME(A1,A2,A3,...AN)

where N has a maximum value of 34 (10 when using compiling MBASIC).

The front end of each graphic primitive has an argument passing subroutine called GETPAR which interfaces to the high level language in use (in this case MBASIC). If programming in assembly language then this subroutine can be omitted and the arguments passed directly in the Z80 registers. On exit from the GETPAR subroutine:

HL contains the first argument
DE contains the second argument
BC contains the third argument

If there are more than three arguments then BC points to a data area containing the third and subsequent arguments (low byte first). The only special case exists for the text plotting graphic primitives which have been designed to accept strings from MBASIC. The programmer should write his/her own text plotting graphic primitives since they are fairly simple. The method of passing arguments from a MBASIC CALL statement is quite complex and is dealt with in section 7.3.

All arguments are passed as 16-bit signed integers using the standard twos complement format. There are no optional arguments. Limited error checking is performed with error messages being displayed on the MV256 screen. There are no fatal errors.

All the graphic primitives consider the logical draw space to be of size +or-2048 x +or-2048 so that the graphic cursor co-ordinates become 12-bit signed integers (bit 11 = sign). Co-ordinates outside of this range are automatically truncated.

Before using the graphic primitives the MV256 display operating mode should be set to NORMAL. This is most easily accomplished by the graphic primitive INIT which should be called at the beginning of a graphics programme.

6.1.3 Function description

A brief description of each graphic primitive is given here. The reader should refer to the source listings in appendix B for further details.

LINETO(X,Y) Line absolute

Draws a line from the graphic cursor (x,y) position to (X,Y) using the current line style and pen colour.

```
-2048 < X < 2048
-2048 < Y < 2048
```

On exit the graphic cursor is positioned at (x+X,y+Y).

LINEBY(DX,DY) Line relative

Draws a line from the current graphic cursor (x,y) position to (x+DX, y+DY) using the current line style and pen colour.

```
-32768 < DX < 32768
-32768 < DY < 32768
```

On exit the graphic cursor is positioned at (x+DX,y+DY).

LINEI(X,Y;X1,Y1) Line immediate

Draws a line from (X,Y) to (X1,Y1) using the current line style and pen colour.

```
-2048 < X < 2048
-2048 < Y < 2048
-2048 < X1 < 2048
-2048 < Y1 < 2048
```

on exit the graphic cursor is positioned at (X1,Y1).

MOVETO(X,Y) Move absolute

Moves the graphic cursor from its current (x,y) position to the point (X,Y).

```
-2048 < X < 2048
-2048 < Y < 2048
```

MOVEBY(DX,DY) Move relative

Moves the graphic cursor from its current (x,y) position to the point (x+DX,y+DY).

```
-32768 < DX < 32768
-32768 < DY < 32768
```

PLOTTO(X,Y) Plot absolute

Plots the point (X,Y) using the current pen colour.

-2048 < X < 2048
-2048 < Y < 2048

On exit the graphic cursor is positioned at (X,Y).

PLOTBY(DX,DY) Plot relative

Plots a point DX,DY from the current graphic cursor (x,y) position using the current pen colour.

-32768 < DX < 32768
-32768 < DY < 32768

On exit the graphic cursor is positioned at (x+DX,y+DY).

CHAR(C) Character relative

Plots a single ASCII character starting at the current graphic cursor (x,y) position using the current character size and style in the current pen colour.

On exit the graphic cursor is positioned at the end of the character.

TEXTTO(X,Y;TEXT) Text absolute

Plots TEXT starting at the point (X,Y) using the current character size and style in the current pen colour.

-2048 < X < 2048
-2048 < Y < 2048

TEXT is a string of ASCII characters. On exit the graphic cursor is positioned at the end of the TEXT string.

TEXTBY(DX,DY;TEXT) Text relative

Plots TEXT starting at a point DX,DY from the current graphic cursor (x,y) position using the current character size and style in the current pen colour.

-32768 < DX < 32768
-32768 < DY < 32768

TEXT is a string of ASCII characters. On exit the graphic cursor is positioned at the end of the TEXT string.

GETP(X,Y) Get colour of pixel absolute

Returns the colour code of the pixel at (X,Y) in the MV256 GETCOL register. This register must be read to determine the actual colour.

-2048 < X < 2048
-2048 < Y < 2048

The graphic cursor is not affected.

JOIN(N;X,Y;X1,Y1;....XN-1,YN-1) Join vertices

Joins N vertices starting at (X,Y) and ending at (XN-1,YN-1) using the current line style and pen colour. N equals the number of vertices, make sure it is correct.

```
    0 < N < 16
-2048 < X < 2048
-2048 < Y < 2048
  .       .
  .       .

-2048 < XN-1 < 2048
-2048 < YN-1 < 2048
```

On exit the graphic cursor is positioned at (XN-1,YN-1).

DBOX(X,Y;DX,DY) Draw Box

Draws a rectangular box starting at (X,Y) and of sides DX,DY using the current line style and pen colour.

```
 -2048 < X < 2048
 -2048 < Y < 2048
-32768 < DX < 32768
-32768 < DY < 32768
```

On exit the graphic cursor is positioned at (X,Y).

DCIR(X,Y;R) Draw circle

Draws a circle centre (X,Y) and of radius R using the continuous line style and current pen colour.

```
-2048 < X < 2048
-2048 < Y < 2048
    0 < R < 2048
```

Radius errors are detected and reported. On exit the graphic cursor is positioned at (X,Y).

DTRI(X,Y;X1,Y1;X2,Y2) Draw triangle

Draws a triangle formed by the vertices (X,Y;X1,Y1;X2,Y2) using the current line style and pen colour.

```
-2048 < X < 2048
-2048 < Y < 2048
-2048 < X1 < 2048
-2048 < Y1 < 2048
-2048 < X2 < 2048
-2048 < Y2 < 2048
```

On exit the graphic cursor is positioned at (X,Y).

DPOLY(N;X,Y;X1,Y1;....XN-1,YN-1) Draw polygon

Draws a polygon formed by the vertices (X,Y;X1,Y1;....XN-1,YN-1) using the current line style and pen colour. N equals the number of vertices, make sure it is correct.

```
    0 < N < 16
-2048 < X < 2048
-2048 < Y < 2048
   :         :
-2048 < XN-1 < 2048
-2048 < YN-1 < 2048
```

On exit the graphic cursor is positioned at (X,Y).

FBOX(X,Y;DX,DY) Fill Box

Fills any box drawn by the DBOX routine using the current line style and pen colour.

FCIR(X,Y;R) Fill circle

Fills any circle drawn by the DCIR routine using the current line style and pen colour.

FTRI(X,Y;X1,Y1;X2,Y2) Fill triangle

Fills any triangle drawn by the DTRI routine using the current line style and pen colour.

FPOLY(N;X,Y;X1,Y1;....XN-1,YN-1) Fill polygon

Fills any convex polygon drawn by the DPOLY routine using the current line style and pen colour. Attempting to fill polygons with concave parts may cause a filling error which is unreported.

CLEARS Clear screen

Clears the screen. The graphic cursor is not affected.

SCAN Set screen

Sets the entire screen to the current pen colour. The graphic cursor is not affected.

LPEN Light pen sequence

Waits for a pixel to be detected by the light pen. On exit the graphic cursor is positioned at the detected pixel.

PENCOL(C) Select pen colour

Sets the current pen colour to C

$$0 \leq C \leq 15$$

LINESTY(S) Select line style

$$0 \leq S \leq 3$$

CHARSTY(S) Select character style

$0 \leq S \leq 3$

CHARSIZ(P,Q) Select X,Y scaling factor

$1 \leq P \leq 16$
$1 \leq Q \leq 16$

Note: 1 = smallest size
16 = largest size

PENSEL Select pen mode

RUBSEL Select eraser mode

FLIP up/down

Inverts the current pen/eraser UP/DOWN status, i.e. if pen is up then FLIP will make pen down.

RESETGC Reset graphic cursor

Resets the graphic cursor co-ordinates to the point (0,0)

INIT Initialise display

Resets all parameter registers and clears the screen

6.2 General Machine code programming hints

Use the supplied subroutines when ever possible. Be very careful when modifying them. The most common error is failure to use the READY subroutine at the appropriate time. Always check that the MV256 is ready before modifying its registers or issuing a new command. All of the graphic primitives incorporate this checking procedure.

7.0 PROGRAMMING THE MV256 USING MBASIC

The graphic primitives detailed in section 6 can be called from Microsofts 5.1 CP/M version of MBASIC. To use with other high level languages will require modification of the argument passing procedure (GETPAR).

7.1 Calling procedure

A machine code graphic primitive is executed by using the basic CALL statement:

LINENUMBER CALL FUNCTIONNAME(ARGUMENTS)

Where FUNCTIONNAME is a variable whose value corresponds to the hex starting address of the appropriate graphic primitive. See table 6.

FUNCTION	START ADDRESS
LINETO(X,Y)	MVBASE + 0
LINEBY(DX,DY)	MVBASE + 3
LINEI(X,Y;X1,Y1)	MVBASE + 6
MOVETO(X,Y)	MVBASE + 9
MOVEBY(DX,DY)	MVBASE + 12
PLOTTO(X,Y)	MVBASE + 15
PLOTBY(DX,DY)	MVBASE + 18
CHAR(C)	MVBASE + 21
TEXTTO(X,Y;TEXT)	MVBASE + 24
TEXTBY(DX,DY;TEXT)	MVBASE + 27
GETP(X,Y)	MVBASE + 30
JOIN(N;X,Y;X1,Y1;....XN-1,YN-1)	MVBASE + 33
DBOX(X,Y;DX,DY)	MVBASE + 36
DCIR(X,Y;R)	MVBASE + 39
DTRI(X,Y;X1,Y1;X2,Y2)	MVBASE + 42
DPOLY(N;X,Y;X1,Y1;....XN-1,YN-1)	MVBASE + 45
FBOX(X,Y;DX,DY)	MVBASE + 48
FCIR(X,Y;R)	MVBASE + 51
FTRI(X,Y;X1,Y1;X2,Y2)	MVBASE + 54
FPOLY(N;X,Y;X1,Y1;....XN-1,YN-1)	MVBASE + 57
CLEARS	MVBASE + 60
SCAN	MVBASE + 63
LPEN	MVBASE + 66
PENCOL(C)	MVBASE + 69
LINESTY(S)	MVBASE + 72
CHARSTY(S)	MVBASE + 75
CHARSIZ(P,Q)	MVBASE + 78
PENSEL	MVBASE + 81
RUBSEL	MVBASE + 84
FLIP	MVBASE + 87
RESETGC	MVBASE + 90
INIT	MVBASE + 93

MVBASE -53502

TABLE 6. Graphic primitive relative start addresses

For example:

```
100 CALL MOVETO(X,Y)
```

The variable MOVETO will have a value of MVBASE+9 where MVBASE is the starting address of the graphic primitive subroutine package. Hence the variable MOVETO could be defined as follows:

```
10 MOVETO=MVBASE+9
```

7.2 Argument variables

All the graphic primitives expect the arguments to be 16-bit signed integers. Thus all variables used by the graphic primitives must be defined as integers within the basic programme. This should be done at the beginning of the basic programme by using the DEFINT statement. Failure to make the arguments integers will cause incorrect values to be passed to the graphic primitives. No fatal errors are possible in this respect. A typical basic programme line would be:

```
5 DEFINT X,Y,D,C,S,R,N
```

Remember that the DEFINT statement clears the defined variable(s) to zero. Make sure that you are not clearing previously defined CALL address variables as well.

7.3 Argument passing procedure

For each argument in the argument list the basic CALL statement will pass a 2 byte parameter to the graphic primitive. This applies for all types of variable. The parameters are pointers to the memory locations containing the actual argument. Thus on entry to the GETPAR subroutine:

```
HL contains the first  parameter
DE contains the second parameter
BC contains the third  parameter
```

If there are more than three arguments then BC points to a data area containing the third and subsequent parameters (low byte first). The GETPAR subroutine uses the passed parameters to get the arguments, which are then passed to the graphic primitives in the HL, DE and BC registers. For more than three arguments GETPAR will put the third and subsequent arguments into a 64 byte data area called ARGDATA. Both MBASIC and GETPAR store the arguments low byte first.

A special case exists for text strings. The passed parameter points to a three byte data area. The first byte contains the number of characters in the text string. The second and third bytes form a pointer to the actual text string which is stored in consecutive memory locations as standard ASCII characters.

The graphic primitive must know how many arguments to expect since MBASIC performs no checks to insure the correct number are present. Some of the graphic primitives perform limited checking on the arguments. Sending the incorrect number of arguments will cause an unreported error. No fatal errors are possible and control is always returned to MBASIC.

7.4 CP/M Programming example

The standard set of graphic primitives should be loaded into memory by executing MVLINK.COM (see appendix B for listing). The linking programme dynamically relocates the primitives immediately below CP/M's FBASE independant of system size and loads MBASIC ready for use with the MV256. The relocated code is protected by putting a jump into FBASE at the base of the code and changing the BDOS jump at location 5 to point at this. Provided MVLINK.COM has been run to load MBASIC the value of MVBASE can be obtained by PEEKing the appropriate memory locations. The following example programme sets the screen to red and plots a yellow filled in circle, centre (100,120) and of radius 80.

```
 5 DEFINT X,Y,R,C
10 MVBASE=(PEEK(6)+PEEK(7)*256+3)-65536!
15 FCIR=MVBASE+51
20 SCAN=MVBASE=63
25 PENCOL=MVBASE+69
30 INIT=MVBASE+93
35 CALL INIT
40 C=4:CALL PENCOL(C):CALL SCAN
45 C=6:CALL PENCOL(C)
50 X=100:Y=120:R=80:CALL FCIR(X,Y,R)
55 END
```

An alternative to using graphic primitives is to use the basic INP and OUT statements which read and write 8-bit integers to a specified I/O port. This method is used in the test programme of section 3.4.3.

7.5 Returned arguments

Some graphic primitives return arguments. These may be read into a basic variable by using the INP statement.

GETP(X,Y)

This primitive should be followed by:

```
C=INP(208)-240
```

The variable C now holds the pixel colour code.

LPEN and graphic cursor position

The LPEN graphic primitive moves the graphic cursor to the detected pixel. Both co-ordinates will be in the range 0-255.

```
X=INP(201)
Y=INP(203)
```

The colour of the pixel detected by the light pen is returned in the GETTCOL register and may be read into a basic variable as indicated for the GETP graphic primitive.

8.0 ANIMATED COMPUTER GRAPHICS

8.1 General principles of animation

An animated picture sequence consists of many similar individual picture frames viewed in rapid succession to give the impression of motion. The most familiar examples are film and television.

Film and television

In order to simulate smooth motion about 24 new picture frames are required per second. A disadvantage of this low picture frequency is that the picture flickers. In film this is checked by projecting every picture frame twice with the aid of a shutter (so that the repetition frequency of twice the picture frequency is achieved while no extra film is used). Television overcomes the problem by using the interlaced scanning technique in which the 625 lines forming a picture are transmitted as two groups of 312.5 lines. Each group is called a field. In this way a repetition frequency (field frequency) of twice the picture frequency is obtained with out increasing the video signal band width. Table 7 gives the various parameters for standard film and UK television.

PARAMETER	UK Television	Film
Picture frequency	25	28
Repetition frequency	50	46

TABLE 7. Picture and repetition frequencies.

Non-interlaced graphics display

In a standard display of this type (the MV256) the CRT screen is scanned once every 20mS to give a repetition frequency of 50Hz. Since one complete picture is displayed during each video frame the picture frequency is also 50Hz.

The rate at which a computer can up-date the displayed picture is referred to as the 'computed picture frequency'. For animation purposes the computed picture frequency needs to be greater than 24Hz.

8.2 Animation of a line drawing.

The procedure for animating a simple line drawing is outlined here. The reader should refer to a text book for a detailed mathematical explanation.

Computation time

Consider the problems of rotating a 'wire frame' cube structure at a reasonable rotation rate to produce an animated effect. The cube may be represented as 8 vertices joined by 12 lines. If we assume no hidden line removal then the vertex joining information will be the same whatever the cubes orientation in space. The vertices can be conveniently represented by a 8 x 4

matrix of homogeneous co-ordinates. The rotation (or any other transformation) can be accomplished by a 4 x 4 general transformation matrix. Equation 1 shows the general form for the transformation of a single vertex. To transform the entire cube the same operation is applied to each of the 8 vertices.

$$(X\ Y\ Z\ 1) \times \begin{matrix} a & b & c & m \\ d & e & f & o \\ g & h & i & p \\ j & k & l & q \end{matrix} = (X^*\ Y^*\ Z^*\ 1) \quad \text{EQU (1)}$$

Original vertex	transformation matrix	transformed vertex

The matrix multiplication for a single vertex requires 16 multiplications and 12 additions. If this is accomplished in software then the time is significiant. A typical Z80A type 8-bit microprocessor based computer requires 80uS to multiply and 12uS to add two 8-bit numbers. An 8-bit number representation is suitable for a 256 x 256 pixel display.

Thus the time to generate a single picture frame can be estimated (i.e. to compute one rotation of the cube):

Time to rotate 1 vertex = (16 x 80) + (12 x 12) uS
= 1424uS

Time to rotate 8 vertices = 8 x 1424 uS
= 11392uS

Hence the corresponding computed picture frequency

= $1/(11392 \times 10^{-6})$ Hz
= 88Hz

The calculation assumes that the execution time of other parts of the programme (erasing the previously drawn cube and incrementing the rotation angle for example) is negligible. Note also that the calculation does not account for the time required to actually plot the computed lines. In systems where no hardwired vector plot facility exists this will add considerably to the over all picture computation time.

The computed picture frequency of 88Hz indicates that the picture complexity could be increased considerably before this fell below 24Hz. In order to prevent jerky motion the movement between successive picture frames should be kept small. However, too small a movement will cause slow and boring animations.

8.3 Animation using the MV256

The intention here is to give the reader an idea of the 'animation ability' of the MV256.

8.3.1 Drawing method

Two drawing methods are considered

(i) Draw during any video blanking period.
(ii) Draw during vertical video blanking only.

The first method can be used to produce complex animated sequences. Vector plotting may take place during any video blanking period. This can lead to partially drawn pictures becoming visible during the animated sequence. The effect of this is to give the impression of actually seeing the picture being drawn.

While the second method produces better overall results it severely limits the picture complexity because of the short vertical video blanking period. Only complete line drawings are displayed during the animated sequence.

8.3.2 Drawing time

The amount of drawing time for each method has been determined:

Method 1 Drawing time = 10641uS per video frame
Method 2 Drawing time = 2739uS per video frame

8.3.3 Picture complexity

By making several reasonable assumptions it is possible to produce an estimate of the maximum picture complexity that can be animated using the MV256.

Assumptions: Average vector length = 65 pixels
Average vector programming time = 8uS
Average vector initialisation time = 2uS

The MV256 takes 571nS to plot each consecutive pixel within a vector. Hence the average total time to plot a vector can be estimated:

Average vector plot time = (65 x .571) + 8 + 2 uS
= 47uS

Hence the total number of vectors which can be drawn per video frame can be estimated:

Method 1. Number of vectors = 10641/47
= 226

Method 2. Number of vectors = 2739/47
= 58

Each picture of the animated sequence must be erased before the next one is drawn. The quickest way of doing this is to redraw using the eraser or in the background colour. Thus the actual picture can only contain half the total number of lines that may be drawn in each video frame (assuming consecutive pictures have the same number of lines).

Thus the maximum picture complexity for both methods becomes:

(1) 113 (2) 29

based on the assumptions outlined earlier.

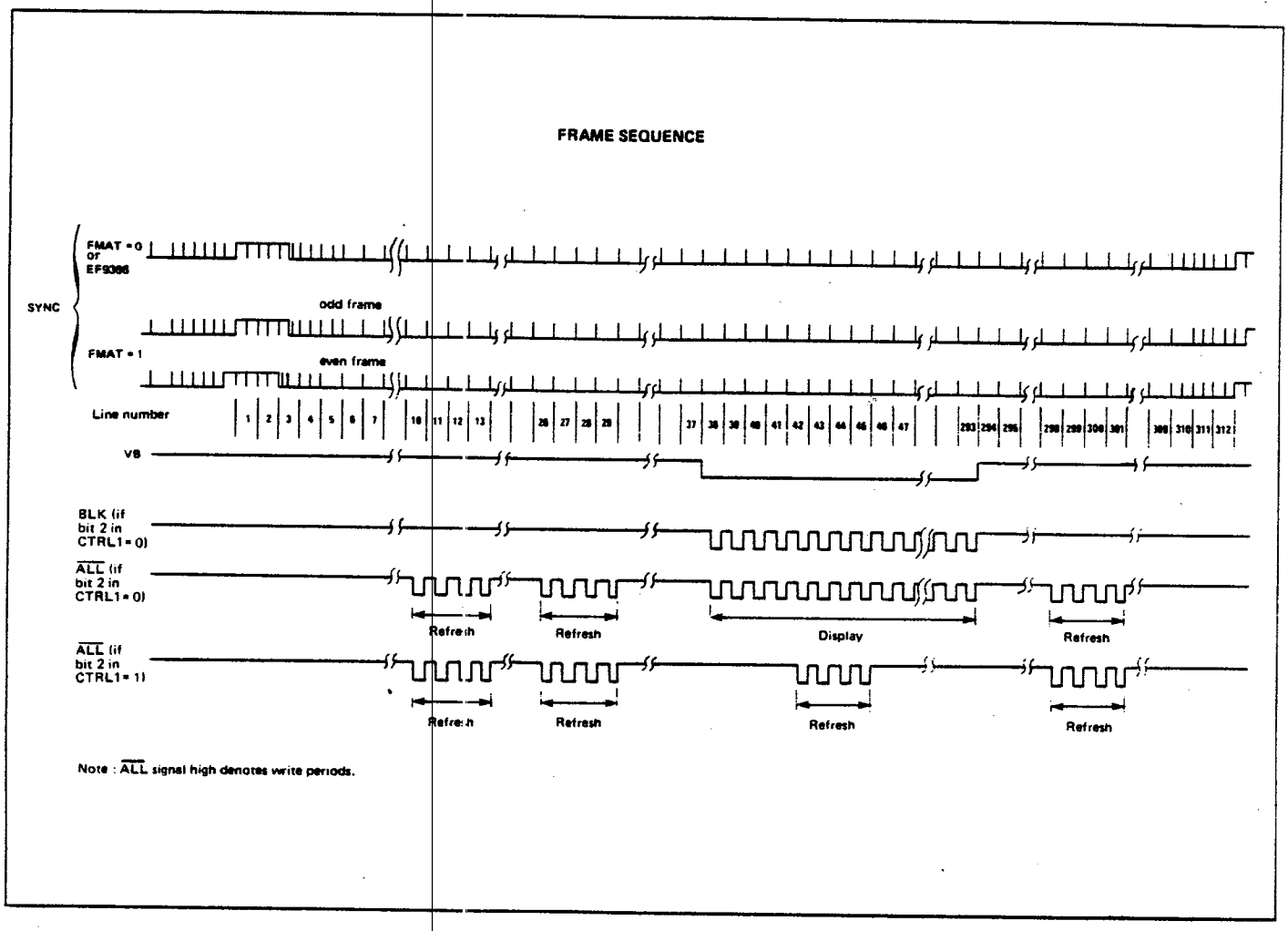
In both cases, it is likely that a new computed picture would be generated every other video frame.

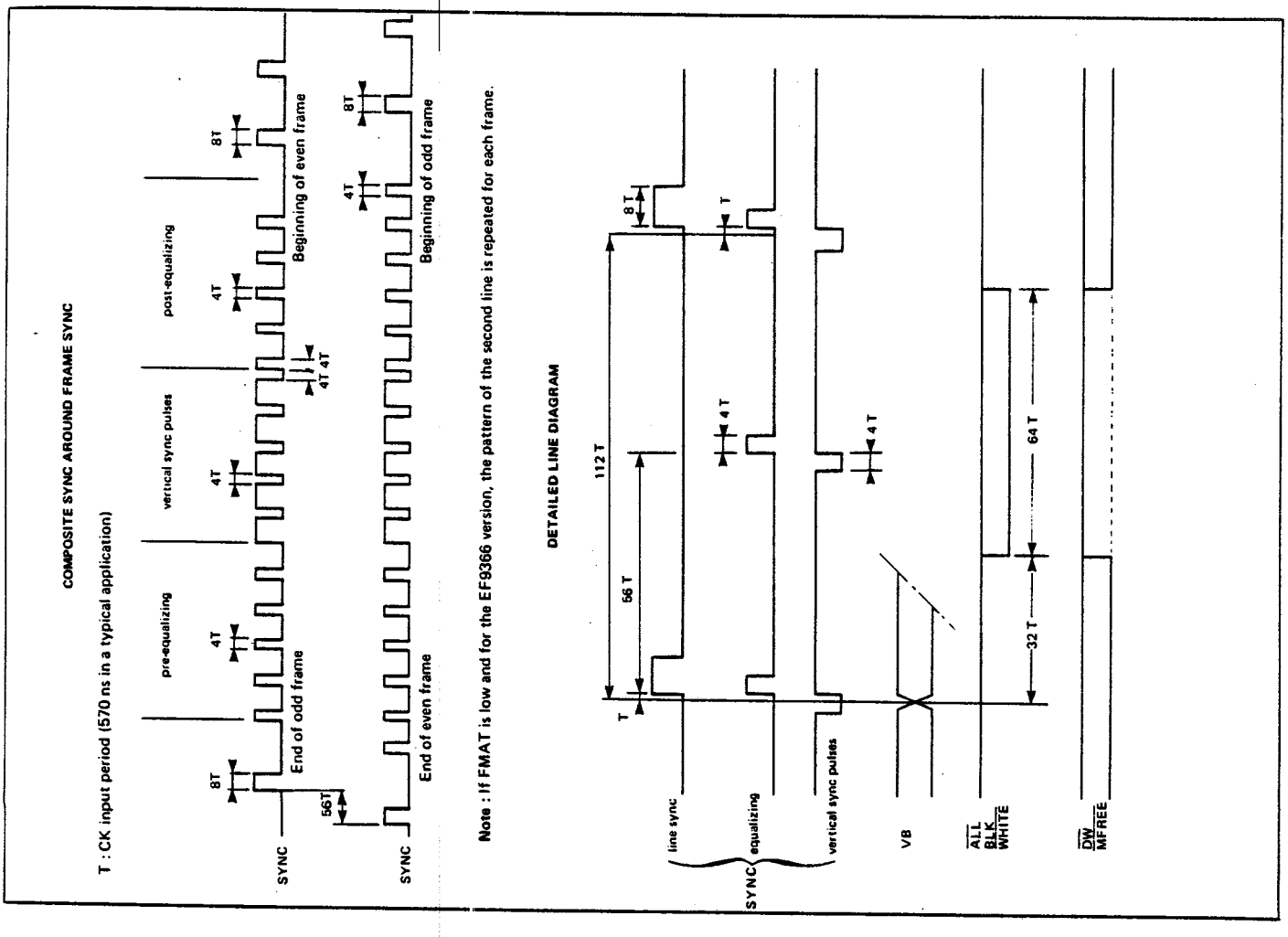
Comparing the above results with that of section 8.2 indicates that the MV256 is able to display animated sequences of greater complexity than any normal 8-bit microprocessor based computer could compute.

Appendix A. EF9365 data sheet extracts

Appendix B. Assembly language source

APPENDIX A


FRAME SEQUENCE
SYNC
FMAT = 0 or EF9366
odd frame
FMAT = 1
even frame
Line number
VB
BLK (if bit 2 in CTRL1 = 0)
ALL (if bit 2 in CTRL1 = 0)
ALL (if bit 2 in CTRL1 = 1)
Refresh
Refresh
Display
Refresh
Refresh
Refresh
Refresh
Refresh
Note : ALL signal high denotes write periods.


COMPOSITE SYNC AROUND FRAME SYNC
T : CK input period (570 ns in a typical application)
pre-equalizing
vertical sync pulses
post-equalizing
8T
4T
56T
SYNC
End of odd frame
End of even frame
Beginning of even frame
Beginning of odd frame
DETAILED LINE DIAGRAM
Note : If FMAT is low and for the EF9366 version, the pattern of the second line is repeated for each frame.
112 T
56 T
64 T
32 T
line sync
equalizing
vertical sync pulses
SYNC
VB
ALL BLK WHITE
DW MFREE

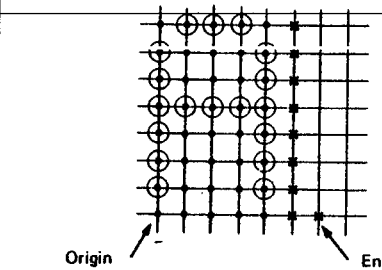
CHARACTER AND SYMBOL GENERATOR

The character generator operates in the same way as the vector generator, i.e. through incrementing or decrementing the X, Y registers, in conjunction with a $\overline{DW}$ output control.

It receives parameters from the CSIZE, CTRL2 and CMD registers. The characters plotted are selected, according to the CMD value, out of 98 matrices (97 8-dot high x 5-dot wide rectangular matrices, and one 4 dot x 4 dot matrix) defined in an internal ROM. Two scaling factors may be applied to the characters plotted using X and Y defined by the CSIZE register. The characters may be tilted, according to the content of register CTRL2.

Basic matrix

Upon completion of a character writing process, the X and Y registers are positioned for writing a further character next to the previous one, with a 1 dot spacing, i.e. Y is restored to its original value and X is incremented by 6.



• Unchanged
⊙ Altered dots
} if CMD = 41_{16} (in the ROM standard version)

× Computed dots, not defined into the ROM (not modifiable).

Scaling factors

Each individual dot in the 5 x 8 basic matrix may be replaced by a P x Q size block.

P : X co-ordinate scaling factor
Q : Y co-ordinate scaling factor

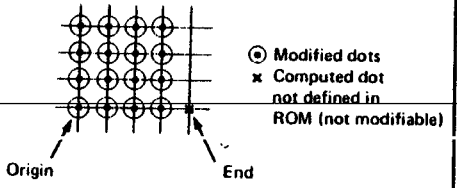
The character size becomes 5P x 8Q. Upon completion of the writing process, X is incremented by 6P. The CK clock cycle count required is 6P x 8Q.

P and Q may each take values from 1 through 16. They are defined by the CSIZE register. Each value is encoded on 4 bits, value 16 being encoded as 0_{16}.

In register CSIZE, P is encoded on the 4 MSBs and Q on the 4 LSBs.

Among the 97 rectangular matrices available in the standard ROM, 96 correspond to CMD values ranging from 20_{16} to $7F_{16}$, and the 97th matrix to $0A_{16}$. In the standard version, these values correspond to the 96 printable characters in the ASCII set. The 97th character is a 5P x 8Q block which may be used for deleting the other characters.

The 98th code (08_{16}) is used to plot a 4P x 4Q graphic block. It locates X, Y, without spacing for the next symbol. Such a block makes it possible to pad uniform areas on the screen.



⊙ Modified dots
× Computed dot not defined in ROM (not modifiable)

Tilted characters

All characters may be modified to produce tilted characters or to mark the vertical co-ordinate with straight or tilted type symbols. Such changes may be achieved using bits 2 and 3 in register CTRL2.

Note : Scaling factors P and Q are always applied within the co-ordinates of the character before conversion.

Character deletion

A character may be deleted using either the same command code or command code $0A_{16}$. In either case, bit 1 in register CTRL1 should be inverted, the origin should be the same as prior to a character plotting operation, as should the scaling factors.

Note : Vector generator and character generator operate in similar ways :

	VECTOR	CHARACTER
Dimensions	DELTAX, DELTAY	CSIZE, tilting
$\overline{DW}$ modulation	Type of line	Character code

USE OF LIGHT PEN CIRCUITRY

A rising edge on the LPCK input is used to sample the current display address in the XLP and YLP registers, provided that this edge is present in the frame immediately following loading of the 08_{16} or 09_{16} code into the CMD register.

Here, the frame origin is counted starting with the VB falling edge. With code 08_{16}, the $\overline{WHITE}$ output recopies the BLK signal from the frame origin up to the rising edge on the LPCK input, or when VB starts rising again, if the LPCK input remains low for the entire frame. With code 09_{16}, the $\overline{WHITE}$ output is not activated.

The YLP address is 8-bit coded since there are 256 display lines in each frame. The XLP address is 6-bit coded since there are 64 display cycles in each line.

These 6 bits are left justified in the XLP register. XLP and YLP register contents match the write address if FMAT is low (or for the EF9366), but should be multiplied by 2 if FMAT is high, so as to be able to match the write address.

The address sampled into XLP corresponds to the current memory cycle. Dots detected by the light pen were addressed in the memory during the previous cycle. Hence, 1 should be subtracted from bit 2 in XLP register where the light pen electronic circuitry does not produce any additional delay.

If the rising edge on input LPCK occurs while VB is low, then the LSB in XLP is set high. This bit acts as a status signal which is reset to the low state by reading register XLP or YLP.

The rising edge first received (LPCK or VB) sets bit 0 in STATUS register high. An interrupt is initiated if bit 4 in CTRL1 is high.

When commands 08_{16} or 09_{16} have been decoded, bit 2 of the status register goes high (circuit ready for any further command) and bit 0 goes low (light pen operating sequence underway).

SCREEN BLANKING COMMANDS

Three commands (04_{16}, 06_{16}, 07_{16}) will set the whole display memory to a status corresponding to a "black display screen" condition. Another command ($0C_{16}$) may be used to set the whole memory to a status other than black (this condition being determined by bit 1 in register CTRL1).

The 4 commands outlined above use the planned scanning of the memory addresses achieved by the display stage. The X and Y registers are not affected by commands 04_{16} and $0C_{16}$. Hence, the time required is that corresponding to one frame (EF9366 or FMAT low) or two frames (FMAT high). The time corresponding to the completion of the frame currently executing when the CMD register is loaded, should be added to the above time.

For the screen blanking process, the frame origin is counted starting with the VB falling edge.
The only signals affected here are the $\overline{DW}$ output, which remains low when VB is low, and the DIN output which is forced high where the 04_{16}, 06_{16} and 07_{16} commands are entered.

Such commands are activated without requiring action by WO input or bit 2 in register CTRL1. While these commands are executing, bit 2 in STATUS register remains low.

EXTERNAL REQUEST FOR DISPLAY MEMORY ACCESS ($\overline{MFREE}$ OUTPUT)

On writing code $0F_{16}$ into the CMD register, the $\overline{MFREE}$ output is set low by the circuitry, during the next free memory cycle.

Apart from the display and refresh periods, this cycle is the first complete cycle that occurs after input $\overline{E}$ is reset high.

During this cycle, those addresses output on DAD and MSL correspond to the X and Y register contents : $\overline{DW}$ is high, $\overline{ALL}$ is high.

Should the memory be engaged in a display or refresh operation, (which is the case when $\overline{ALL}$ is low), then this cycle is postponed to be executed after $\overline{ALL}$ is reset high. The maximum waiting time is thus 64 cycles.

The $\overline{MFREE}$ signal may be used e. g. for performing a read or write operation into a register located between the display memory and the microprocessor bus.

INTERRUPTS OPERATION

An interrupt may be initiated by three situations denoted by internal signals :

- Circuit ready for a further command
- Vertical blanking signal
- Light pen sequence completed.

These three signals appear in real time in the STATUS register (bits 0, 1, 2). Each signal is cross-referenced to a mask bit in the register CTRL1 (bits 4, 5, 6).

If the mask bit is high, the first rising edge that occurs on the interrupt initiating signal sets the related interrupt flip-flop circuit high.

The outputs from these three flip-flop circuits appear in the STATUS register (bits 4, 5, 6). If one flip-flop circuit is high, bit 7 in the STATUS register is high, and pin $\overline{IRQ}$ is forced low.

A read operation in the STATUS register resets its 4 MSBs low, after input $\overline{E}$ is reset high.

The three interrupt control flip-flops are duplicated to prevent the loss of an interrupt coming during a read cycle of the STATUS register.

The status of bits 4, 5 and 6 corresponds to the interrupt control flip-flop circuit output, before input $\overline{E}$ goes low.

An interrupt coming during a read cycle of the STATUS register does not appear in bits 4, 5 and 6 during this read sequence, but during the following one. However, it may appear in bits 0, 1, 2 or on pin $\overline{IRQ}$.

APPENDIX B

```
MV 256 Relocating Graphics Drivers    MACRO-80 3.43  27-Jul-81      PAGE   1-2
Relocation section

 0114   11 00CE              LD DE,MVBASE-LDBASE ; Offset to base MBASIC loader
 0117   B7                  OR A
 0118   ED 52               SBC HL,DE       ; Calc actual LDBASE and save
 011A   E5                  PUSH HL
 011B   11 01C9             LD DE,LDBASE    ; Calculate offset for relocator
 011E   B7                  OR A
 011F   ED 52               SBC HL,DE
 0121   22 0CA3             LD (REL),HL
 0124   E1                  POP HL          ; Restore relocated LDBASE
 0125   EB                  EX DE,HL        ; Source in [HL], destin in [DE]
 0126   01 0AD0             LD BC,MVEND-LDBASE ; length in [BC]
 0129   ED B0               LDIR            ; Move code
 012B   E1                  POP HL          ; Restore new FBASE+3
 012C   D1                  POP DE          ; Restore old FBASE
 012D   E5                  PUSH HL         ; Save new FBASE+3
 012E   2B                  DEC HL          ; Put jump into FDOS at base of
 012F   72                  LD (HL),D       ; graphics routines
 0130   2B                  DEC HL
 0131   73                  LD (HL),E
 0132   2B                  DEC HL
 0133   22 0006             LD (6),HL       ; Put new FBASE into location 6
 0136   E1                  POP HL          ; Restore relocated MVBASE
 0137   CD 0169             CALL DISADR     ; Convert to ASCII and print

                    ;*************************************************
                    ;******** RELOCATE ABSOLUTE ADDRESSES IN CODE *********
                    ;*************************************************

 013A   21 0D0C             LD HL,PADDR
 013D   5E          ALP:    LD E,(HL)       ; Get address from table
 013E   23                  INC HL
 013F   56                  LD D,(HL)
 0140   23                  INC HL
 0141   7A                  LD A,D          ; Test for end table
 0142   B3                  OR E
 0143   28 15               JR Z,AFIN
 0145   E5                  PUSH HL         ; Save pointer to table
 0146   2A 0CA3             LD HL,(REL)     ; Get offset
 0149   19                  ADD HL,DE
 014A   5E                  LD E,(HL)       ; Get address from code
 014B   23                  INC HL
 014C   56                  LD D,(HL)
 014D   E5                  PUSH HL         ; Save address
 014E   2A 0CA3             LD HL,(REL)     ; Add in offset
 0151   19                  ADD HL,DE
 0152   EB                  EX DE,HL
 0153   E1                  POP HL          ; Restore address
 0154   72                  LD (HL),D       ; Put back new code
 0155   2B                  DEC HL
 0156   73                  LD (HL),E
 0157   E1                  POP HL          ; Next position in table
 0158   18 E3               JR ALP

 015A   2A 0006     AFIN:   LD HL,(6)       ; Get MVBASE-3
 015D   11 00CB             LD DE,MVBASE-LDBASE-3 ; Calculate address
 0160   B7                  OR A            ; of LDBASE
 0161   ED 52               SBC HL,DE
 0163   E9                  JP (HL)
```

APPENDIX B

```
MV 256 Relocating Graphics Drivers      MACRO-80 3.43   27-Jul-81       PAGE    1-4
Relocation section

 01B6   19                  BXDEC3: ADD HL,DE       ; Restore to positive
 01B7   C6 30                       ADD A,30H       ; Convert to ASCII 0-9
 01B9   DD 77 00                    LD (IX),A       ; Store in buffer
 01BC   DD 23                       INC IX          ; Bump buffer pointer and point
 01BE   FD 23                       INC IY          ; to next power of ten
 01C0   FD 23                       INC IY
 01C2   7B                          LD A,E          ; Test for 5 digits
 01C3   FE 01                       CP 1
 01C5   C2 01A5                     JP NZ,BXDEC1    ; Outer loop
 01C8   C9                          RET

                            ;*************************************************
                            ;****** LOAD MBASIC FROM CURRENTLY LOGGED DRIVE *******
                            ;*************************************************
                            ;********** THIS ROUTINE IS MOVED BELOW THE ***********
                            ;************ NEW FBASE BEFORE EXECUTION **************
                            ;*************************************************

 01C9   11 0273             LDBASE: LD DE,FCB       ; Open file
                        +           R
 01CC   0E 0F                       LD C,OPEN
 01CE   CD 0005                     CALL BDOS
 01D1   FE FF                       CP 0FFH         ; Print error mess and
 01D3   20 0B                       JR NZ,LDBAS1    ; reboot if file not
 01D5   11 023B                     LD DE,ERRM      ; found
                        +           R
 01D8   0E 09                       LD C,PRTSTG
 01DA   CD 0005                     CALL BDOS
 01DD   C3 0000                     JP 0
 01E0   11 020B             LDBAS1: LD DE,BASM      ; Print loading basic mess
                        +           R
 01E3   0E 09                       LD C,PRTSTG
 01E5   CD 0005                     CALL BDOS
 01E8   21 0100                     LD HL,100H      ; Start TPA
 01EB   EB                  LDBAS2: EX DE,HL        ; Get new DMA address in DE
 01EC   D5                          PUSH DE         ; and save
 01ED   0E 1A                       LD C,SETDMA
 01EF   CD 0005                     CALL BDOS
 01F2   11 0273                     LD DE,FCB       ; Read next sector
                        +           R
 01F5   0E 14                       LD C,RDSEQ
 01F7   CD 0005                     CALL BDOS
 01FA   E1                          POP HL          ; Restore DMA address
 01FB   B7                          OR A            ; Check if EOF
 01FC   20 06                       JR NZ,LDBAS3
 01FE   11 0080                     LD DE,128       ; Bump DMA for next sector
 0201   19                          ADD HL,DE
 0202   18 E7                       JR LDBAS2       ; Load next sector
 0204   21 0000             LDBAS3: LD HL,0         ; Force return via warmboot
 0207   E5                          PUSH HL         ; and execute TPA
 0208   C3 0100                     JP 100H

 020B   0A 20 3C 3C         BASM:   DB LF," (((*** Loading MBASIC - please wait ***)))",CR,LF,LF,"$"
 020F   3C 2A 2A 2A
 0213   20 4C 6F 61
 0217   64 69 6E 67
 021B   20 4D 42 41
 021F   53 49 43 20
```

APPENDIX B

```
MV 256 Relocating Graphics Drivers     MACRO-80 3.43  27-Jul-81      PAGE   1-6
Graphics routines

                        ;*** CHAR(C)                          CHARACTER RELATIVE      0021
                        ;*** TXTTO(X,Y;TEXT)                  TEXT ABSOLUTE           0024
                        ;*** TXTBY(DX,DY;TEXT)                TEXT RELATIVE           0027
                        ;*** GETP(X,Y)                        GET PIXEL ABSOLUTE      0030
                        ;*** JOIN(N;X,Y;X1,Y1;..XN-1,YN-1)    JOIN POINTS             0033
                        ;*** DBOX(X,Y;DX,DY)                  DRAW RECTANGULAR BOX    0036
                        ;*** DCIR(X,Y;R)                      DRAW CIRCLE             0039
                        ;*** DTRI(X,Y;X1,Y1;X2,Y2)            DRAW TRIANGLE           0042
                        ;*** DPOLY(N;X,Y;X1,Y1;..XN-1,YN-1)   DRAW POLYGON            0045
                        ;*** FBOX(X,Y;DX,DY)                  FILL RECTANGULAR BOX    0048
                        ;*** FCIR(X,Y;R)                      FILL CIRCLE             0051
                        ;*** FTRI(X,Y;X1,Y1;X2,Y2)            FILL TRIANGLE           0054
                        ;*** FPOLY(N;X,Y;X1,Y1;..XN-1,YN-1)   FILL CONVEX POLYGON     0057
                        ;*** CLEARS                           CLEAR SCREEN            0060
                        ;*** SCAN                             SET SCREEN              0063
                        ;*** LPEN                             LIGHT PEN SEQUENCE      0066
                        ;*** PNCOL(C)                         SELECT PEN COLOUR       0069
                        ;*** LNSTY(S)                         SELECT LINE STYLE       0072
                        ;*** CHRSTY(S)                        SELECT CHARACTER STYLE  0075
                        ;*** CHRSIZ(P,Q)                      SET CHARACTER SIZE      0078
                        ;*** PENSEL                           SELECT PEN              0081
                        ;*** RUBSEL                           SELECT ERASER           0084
                        ;*** FLIP                             INVERT P/E UP/DWN STATE 0087
                        ;*** RESGC                            RESET GRAPHIC CURSOR    0090
                        ;*** INIT                             INITIALISE DISPLAY      0093
                        ;*** READY                            WAIT FOR MV256 READY    0096
                        ;*** SPARE JUMPS FOR EXPANSION                                0099-0126

                        ;************ MV256 HARDWARE REGISTERS

00C0                    STAT    EQU     0C0H    ;MV256 STATUS REGISTER (R)
00C0                    CMD     EQU     0C0H    ;MV256 COMMAND REGISTER (W)
00C1                    CNTL1   EQU     0C1H    ;MV256 CONTROL1 REGISTER (R/W)
00C2                    CNTL2   EQU     0C2H    ;MV256 CONTROL2 REGISTER (R/W)
00C3                    CSIZE   EQU     0C3H    ;MV256 CHARACTER SIZE REGISTER (R/W)
00C5                    DX      EQU     0C5H    ;MV256 DELTAX REGISTER (R/W)
00C7                    DY      EQU     0C7H    ;MV256 DELTAY REGISTER (R/W)
00C8                    XCM     EQU     0C8H    ;MV256 X MSBs REGISTER (R/W)
00C9                    XCL     EQU     0C9H    ;MV256 X LSBs REGISTER (R/W)
00CA                    YCM     EQU     0CAH    ;MV256 Y MSBs REGISTER (R/W)
00CB                    YCL     EQU     0CBH    ;MV256 Y LSBs REGISTER (R/W)
00CC                    XLP     EQU     0CCH    ;MV256 X LIGHT PEN REGISTER (R)
00CD                    YLP     EQU     0CDH    ;MV256 Y LIGHT PEN REGISTER (R)
00D0                    PENCOL  EQU     0D0H    ;MV256 PEN COLOUR REGISTER (W)
00D0                    GETCOL  EQU     0D0H    ;MV256 GET COLOUR REGISTER (R)


                        ;*** ACTUAL JUMP TABLE ENTRIES

0297    C3 0319                 JP      LNTO
                      +         R
029A    C3 034D                 JP      LNBY
                      +         R
029D    C3 03C2                 JP      LINEI
                      +         R
02A0    C3 03D8                 JP      MVTO
                      +         R
```

APPENDIX B

```
MV 256 Relocating Graphics Drivers      MACRO-80 3.43  27-Jul-81      PAGE   1-8
Graphics routines

  02FA   C3 0318                        JP      SPARE   ; 10 SPARE JUMPS FOR EXPANSION
                        +               R
  02FD   C3 0318                        JP      SPARE
                        +               R
  0300   C3 0318                        JP      SPARE
                        +               R
  0303   C3 0318                        JP      SPARE
                        +               R
  0306   C3 0318                        JP      SPARE
                        +               R
  0309   C3 0318                        JP      SPARE
                        +               R
  030C   C3 0318                        JP      SPARE
                        +               R
  030F   C3 0318                        JP      SPARE
                        +               R
  0312   C3 0318                        JP      SPARE
                        +               R
  0315   C3 0318                        JP      SPARE

                                ;*************************************************************************
                                ;*** RETURN POINTED TO BY EXPANSION JUMPS
  0318   C9                    SPARE:  RET

                                ;*************************************************************************;*** SUBROUTINE LNTO
X,Y) *** DRAW LINE ABSOLUTE ***
                                ;*** THIS SUBROUTINE DRAWS A LINE FROM THE CURRENT GRAPHIC CURSOR (x,y)
                                ;*** POSITION TO (X,Y) USING THE CURRENT LINE STYLE AND PEN COLOUR
                                ;*** ON ENTRY REGISTER HL POINTS TO X
                                ;*** ON ENTRY REGISTER DE POINTS TO Y
                                ;*** ON EXIT THE GRAPHIC CURSOR IS POSITIONED AT (X,Y)
  0319   37                    LNTO:   SCF             ;LESS THAN THREE ARGUMENTS
  031A   CD 0909                        CALL    GETPAR  ;GET ARGUMENTS
                        +               R
  031D   C5                     LNTOP:  PUSH    BC      ;SAVE REGISTERS
  031E   08                             EX      AF,AF'
  031F   F5                             PUSH    AF
  0320   CD 093B                        CALL    TRUN    ;TRUNCATE CO-ORDINATES
                        +               R
  0323   CD 08E0                        CALL    READY   ;WAIT FOR MV256 READY
                        +               R
  0326   DB C9                          IN      A,(XCL) ;GET CURRENT GRAPHIC CURSOR X CO-ORDINATE
  0328   4F                             LD      C,A     ;X LSBs
  0329   DB C8                          IN      A,(XCM) ;X MSBs
  032B   CB 5F                          BIT     3,A     ;TEST SIGN OF X
  032D   28 02                          JR      Z,LNTO1 ;JUMP IF +VE
  032F   F6 F0                          OR      0F0H    ;ELSE CONVERT TO FULL 16-BIT -VE NO
  0331   47                     LNTO1:  LD      B,A
  0332   B7                             OR      A       ;CLEAR CARRY
  0333   ED 42                          SBC     HL,BC   ;COMPUTE DX VECTOR PROJECTION
  0335   EB                             EX      DE,HL   ;DX TO DE,Y TO HL
  0336   DB CB                          IN      A,(YCL) ;GET CURRENT GRAPHIC CURSOR Y CO-ORDINATE
  0338   4F                             LD      C,A     ;Y LSBs
  0339   DB CA                          IN      A,(YCM) ;Y MSBs
  033B   CB 5F                          BIT     3,A     ;TEST SIGN OF Y
  033D   28 02                          JR      Z,LNTO2 ;JUMP IF +VE
  033F   F6 F0                          OR      0F0H    ;ELSE CONVERT TO FULL 16-BIT -VE NO
  0341   47                     LNTO2:  LD      B,A
  0342   B7                             OR      A       ;CLEAR CARRY
```

APPENDIX B

```
MV 256 Relocating Graphics Drivers     MACRO-80 3.43  27-Jul-81      PAGE   1-10
Graphics routines

038A  60                LNBY6:  LD      H,B     ;FRACTION TO H.INTEGER PART IN L
038B  79                        LD      A,C     ;SAVE SEGMENT COUNTER
038C  CB 39             LNBY7:  SRL     C       ;SEGMENT COUNTER/2
038E  38 08                     JR      C,LNBY8 ;TEST TO SEE IF LAST SEGMENT.JUMP IF SO
0390  CB 3A                     SRL     D       ;DIVIDE SMALLEST MSBs PROJECTION IN DE BY 2
0392  CB 1B                     RR      E
0394  CB 18                     RR      B       ;GENERATED FRACTION TO B (less than one screen unit)
0396  18 F4                     JR      LNBY7   ;DO UNTILL ALL SEGMENTS DONE
0398  50                LNBY8:  LD      D,B     ;FRACTION TO D.INTEGER PART IN E
0399  08                        EX      AF,AF'  ;GET BACK PROJECTION SWAP FLAG
039A  30 01                     JR      NC,LNBY9 ;IF NO SWAP TO DO THEN JUMP
039C  EB                        EX      DE,HL   ;DX TO DE,DY TO HL
039D  08                LNBY9:  EX      AF,AF'  ;SAVE MV256 COMMAND AND GET BACK SEGMENT COUNTER
039E  01 0000                   LD      BC,0000H ;CLEAR BC
03A1  F5                LNBYA:  PUSH    AF      ;SAVE VECTOR SEGMENT COUNTER
03A2  7A                        LD      A,D     ;GET DX FRACTION
03A3  80                        ADD     A,B     ;ADD TO DX CUMULATIVE ERROR
03A4  47                        LD      B,A     ;SAVE NEW CUMULATIVE DX ERROR
03A5  7B                        LD      A,E     ;INTEGER PART OF DX TO ACC
03A6  30 01                     JR      NC,LNBYB ;IF CUMULATIVE DX ERROR ( 1 PIXEL THEN JUMP
03A8  3C                        INC     A       ;OTHER WISE COMPENSATE FOR ERROR
03A9  F5                LNBYB:  PUSH    AF      ;SAVE DX
03AA  7C                        LD      A,H     ;GET DY FRACTION
03AB  81                        ADD     A,C     ;ADD TO CUMULATIVE DY ERROR
03AC  4F                        LD      C,A     ;SAVE NEW CUMULATIVE DY ERROR
03AD  7D                        LD      A,L     ;INTEGER PART OF DY TO ACC
03AE  30 01                     JR      NC,LNBYC ;IF CUMULATIVE DY ERROR ( 1 PIXEL THEN JUMP
03B0  3C                        INC     A       ;OTHER WISE COMPENSATE FOR ERROR
03B1  CD 08E0           LNBYC:  CALL    READY   ;WAIT FOR MV256 READY
                      +         R
03B4  D3 C7                     OUT     (DY),A  ;Y VECTOR PROJECTION TO MV256
03B6  F1                        POP     AF      ;GET BACK DX
03B7  D3 C5                     OUT     (DX),A  ;X VECTOR PROJECTION TO MV256
03B9  08                        EX      AF,AF'  ;GET BACK MV256 VECTOR COMMAND
03BA  D3 C0                     OUT     (CMD),A ;START DRAWING VECTOR
03BC  08                        EX      AF,AF'  ;SAVE MV256 VECTOR COMMAND
03BD  F1                        POP     AF      ;GET BACK VECTOR SEGMENT COUNTER
03BE  3D                        DEC     A       ;NEXT SEGMENT
03BF  20 E0                     JR      NZ,LNBYA ;DO FOR ALL VECTOR SEGMENTS
03C1  C9                        RET             ;FINISHED
                        ;*************************
                        ;*** SUBROUTINE LINEI(X,Y;X1,Y1) *** LINE IMMEDIATE ***
                        ;*** THIS SUBROUTINE DRAWS A LINE FROM (X,Y) TO (X1,Y1) USING THE CURRENT
                        ;*** LINE STYLE AND PEN COLOUR ***
                        ;*** ON ENTRY REGISTER HL POINTS TO X
                        ;*** ON ENTRY REGISTER DE POINTS TO Y
                        ;*** ON ENTRY REGISTER BC INDIRECTLY POINTS TO (X1,Y1)
                        ;*** ON EXIT THE GRAPHIC CURSOR IS POSITIONED AT (X1,Y1)
03C2  B7                LINEI:  OR      A       ;MORE THAN THREE ARGUMENTS
03C3  CD 0909                   CALL    GETPAR  ;GET ARGUMENTS
                      +         R
03C6  CD 03DC                   CALL    MVTOP   ;MOVE TO (X,Y)
                      +         R
03C9  0A                        LD      A,(BC)  ;GET X1
03CA  6F                        LD      L,A
03CB  03                        INC     BC
03CC  0A                        LD      A,(BC)
03CD  67                        LD      H,A
```

APPENDIX B

```
MV 256 Relocating Graphics Drivers     MACRO-80 3.43  27-Jul-81      PAGE   1-12
Graphics routines

 040F   F6 F0                    OR      0F0H    ;ELSE CONVERT TO FULL 16-BIT -VE NO
 0411   47              MVBY2:  LD      B,A     ;Y MSBs
 0412   DB CB                    IN      A,(YCL)
 0414   4F                      LD      C,A     ;Y LSBs
 0415   EB                      EX      DE,HL   ;DY TO HL
 0416   09                      ADD     HL,BC   ;COMPUTE NEW Y POSITION
 0417   7C                      LD      A,H     ;PUT TO MV256
 0418   D3 CA                   OUT     (YCM),A ;Y MSBs
 041A   7D                      LD      A,L
 041B   D3 CB                   OUT     (YCL),A ;Y LSBs
 041D   C9                      RET             ;FINISHED
                        ;************************
                        ;*** SUBROUTINE PLTTO(X,Y) *** PLOT ABSOLUTE ***
                        ;*** THIS SUBROUTINE PLOTS A SINGLE PIXEL IN THE CURRENT PEN COLOUR AT (X,Y) **
                        ;*** ON ENTRY REGISTER HL POINTS TO X
                        ;*** ON ENTRY REGISTER DE POINTS TO Y
                        ;*** ON EXIT THE GRAPHIC CURSOR IS POSITIONED AT (X,Y)
 041E   37              PLTTO:  SCF             ;LESS THAN THREE ARGUMENTS
 041F   CD 0909                 CALL    GETPAR  ;GET ARGUMENTS
                    +           R
 0422   CD 03DC        PLTTOP: CALL    MVTOP   ;MOVE THE GRAPHIC CURSOR TO (X,Y)
                    +           R
 0425   DB C2                   IN      A,(CNTL2) ;GET CURRENT LINE STYLE
 0427   F5                      PUSH    AF      ;SAVE IT
 0428   AF                      XOR     A       ;CONTINUOUS LINE STYLE
 0429   D3 C2                   OUT     (CNTL2),A
 042B   3E 80                   LD      A,80H   ;PLOT POINT COMMAND
 042D   D3 C0                   OUT     (CMD),A
 042F   F1                      POP     AF      ;GET BACK LINE STYLE
 0430   CD 08E0                 CALL    READY   ;WAIT FOR MV256 TO PLOT POINT
                    +           R
 0433   D3 C2                   OUT     (CNTL2),A ;RESTORE ORIGINAL LINE STYLE
 0435   C9                      RET             ;FINISHED
                        ;************************
                        ;*** SUBROUTINE PLTBY(DX,DY) *** PLOT RELATIVE ***
                        ;*** THIS SUBROUTINE PLOTS A SINGLE PIXEL IN THE CURRENT PEN COLOUR AT A
                        ;*** POSITION DX,DY FROM THE CURRENT GRAPHIC CURSOR (x,y) POSITION ***
                        ;*** ON ENTRY REGISTER HL POINTS TO DX
                        ;*** ON ENTRY REGISTER DE POINTS TO DY
                        ;*** ON EXIT THE GRAPHIC CURSOR IS POSITIONED AT (x+DX,y+DY)
 0436   37              PLTBY:  SCF             ;LESS THAN THREE ARGUMENTS
 0437   CD 0909                 CALL    GETPAR  ;GET ARGUMENTS
                    +           R
 043A   CD 03F3         PLTBYP: CALL    MVBYP   ;MOVE THE GRAPHIC CURSOR TO (x+DX,y+DY)
                    +           R
 043D   DB C2                   IN      A,(CNTL2) ;GET CURRENT LINE STYLE
 043F   F5                      PUSH    AF      ;SAVE IT
 0440   AF                      XOR     A       ;CONTINUOUS LINE STYLE
 0441   D3 C2                   OUT     (CNTL2),A
 0443   3E 80                   LD      A,80H   ;PLOT POINT COMMAND
 0445   D3 C0                   OUT     (CMD),A
 0447   F1                      POP     AF      ;GET BACK LINE STYLE
 0448   CD 08E0                 CALL    READY   ;WAIT FOR MV256 TO PLOT POINT
                    +           R
 044B   D3 C2                   OUT     (CNTL2),A ;RESTORE ORIGINAL LINE STYLE
 044D   C9                      RET             ;FINISHED
                        ;************************
                        ;*** SUBROUTINE CHAR(C) *** CHARACTER RELATIVE ***
```

APPENDIX B

```
MV 256 Relocating Graphics Drivers    MACRO-80 3.43  27-Jul-81      PAGE   1-14
Graphics routines

  048A   10 F7                        DJNZ    TXTBY1  ;DO FOR ALL CHARACTERS
  048C   C9                          RET             ;FINISHED
                              ;************************
                              ;*** SUBROUTINE GETP(X,Y) *** GET PIXEL ABSOLUTE ***
                              ;*** THIS SUBROUTINE GETS THE COLOUR CODE OF THE PIXEL AT THE POINT (X,Y)
                              ;*** ON ENTRY REGISTER HL POINTS TO X
                              ;*** ON ENTRY REGISTER DE POINTS TO Y
                              ;*** ON EXIT THE 'GETCOL' REGISTER HOLDS THE COLOUR CODE OF THE PIXEL AT (X,Y)
                              ;*** THE GRAPHIC CURSOR IS PRESERVED
  048D   37                   GETP:   SCF             ;LESS THAN THREE ARGUMENTS
  048E   CD 0909                      CALL    GETPAR  ;GET ARGUMENTS
                         +            R
  0491   CD 08E0                      CALL    READY   ;WAIT FOR MV256 READY
                         +            R
  0494   DB C8                       IN      A,(XCM) ;SAVE CURRENT GRAPHIC CURSOR
  0496   47                          LD      B,A     ;X MSBs
  0497   DB C9                       IN      A,(XCL)
  0499   4F                          LD      C,A     ;X LSBs
  049A   C5                          PUSH    BC      ;SAVE X
  049B   DB CA                       IN      A,(YCM)
  049D   47                          LD      B,A     ;Y MSBs
  049E   DB CB                       IN      A,(YCL)
  04A0   4F                          LD      C,A     ;Y LSBs
  04A1   CD 08C1                     CALL    RESGC   ;RESET GRAPHIC CURSOR
                         +            R
  04A4   CD 08E0                     CALL    READY   ;WAIT FOR MV256 TO DO THIS
                         +            R
  04A7   7D                          LD      A,L     ;POINT TO REQUIRED PIXEL.ONLY LSBs OF CO-ORD ARE VALID
  04A8   D3 C9                       OUT     (XCL),A ;X LSBs
  04AA   7B                          LD      A,E
  04AB   D3 CB                       OUT     (YCL),A ;Y LSBs
  04AD   3E 0F                       LD      A,0FH   ;'GET PIXEL CODE' COMMAND
  04AF   D3 C0                       OUT     (CMD),A
  04B1   CD 08E0                     CALL    READY   ;WAIT FOR MV256 TO GET PIXEL
                         +            R
  04B4   78                          LD      A,B     ;RESTORE ORIGINAL GRAPHIC CURSOR
  04B5   D3 CA                       OUT     (YCM),A ;Y MSBs
  04B7   79                          LD      A,C
  04B8   D3 CB                       OUT     (YCL),A ;Y LSBs
  04BA   C1                          POP     BC      ;UNSAVE X
  04BB   78                          LD      A,B
  04BC   D3 C8                       OUT     (XCM),A ;X MSBs
  04BE   79                          LD      A,C
  04BF   D3 C9                       OUT     (XCL),A ;X LSBs
  04C1   DB D0                       IN      A,(GETCOL) ;PIXEL COLOUR CODE TO ACC
  04C3   C9                          RET             ;FINISHED.PIXEL COLOUR CODE ALSO IN 'GETCOL' REGISTER
                              ;************************
                              ;*** SUBROUTINE JOIN(N;X,Y;X1,Y1;..     ...XN-1,YN-1) *** JOIN POINTS ***
                              ;*** THIS SUBROUTINE JOINS UP N VERTICES STARTING FROM (X,Y) USING THE CURRENT
                              ;*** LINE STYLE AND PEN COLOUR ***
                              ;*** ON ENTRY REGISTER HL POINTS TO N,THE NUMBER OF VERTICES (max 16)
                              ;*** ON ENTRY REGISTER DE POINTS TO X
                              ;*** ON ENTRY REGISTER BC INDIRECTLY POINTS TO (Y;X1,Y1;..      ...XN-1,YN-1)
                              ;*** ON EXIT THE GRAPHIC CURSOR IS POSITIONED AT (XN-1,YN-1)
  04C4   B7                   JOIN:   OR      A       ;MORE THAN THREE ARGUMENTS
  04C5   CD 0909                      CALL    GETPAR  ;GET ARGUMENTS
                         +            R
  04C8   7D                          LD      A,L     ;NUMBER OF VERTICES TO ACC
```

APPENDIX B

```
MV 256 Relocating Graphics Drivers     MACRO-80 3.43  27-Jul-81      PAGE   1-16
Graphics routines

 0505   03                             INC     BC
 0507   C5                             PUSH    BC
 0509   D9                             EXX             ;TO DY REGISTERS
 0509   C1                             POP     BC
 0509   0A                             LD      A,(BC)  ;GET DY
 050B   5F                             LD      E,A     ;DY LSBs
 050C   03                             INC     BC
 050D   0A                             LD      A,(BC)
 050E   57                             LD      D,A     ;DY MSBs
 050F   21 0000                        LD      HL,0000H ;SAY DX IS ZERO
 0512   D5                             PUSH    DE      ;SAVE DY
 0513   CD 0351                       CALL    LNBYP   ;DRAW FIRST SIDE
                        +               R
 0516   D1                             POP     DE      ;UNSAVE DY
 0517   D9                             EXX             ;TO DX REGISTERS
 0518   E5                             PUSH    HL      ;SAVE DX
 0519   11 0000                        LD      DE,0000H ;SAY DY IS ZERO
 051C   CD 0351                       CALL    LNBYP   ;DRAW SECOND SIDE
                        +               R
 051F   E1                             POP     HL      ;UNSAVE DX
 0520   D9                             EXX             ;TO DY REGISTERS
 0521   AF                             XOR     A       ;CLEAR CARRY
 0522   21 0000                        LD      HL,0000H ;CHANGE SIGN OF DY
 0525   ED 52                          SBC     HL,DE
 0527   EB                             EX      DE,HL
 0528   21 0000                        LD      HL,0000H ;SAY DX IS ZERO
 052B   CD 0351                       CALL    LNBYP   ;DRAW THIRD SIDE
                        +               R
 052E   D9                             EXX             ;TO DX REGISTERS
 052F   AF                             XOR     A       ;CLEAR CARRY
 0530   EB                             EX      DE,HL   ;CHANGE SIGN OF DX
 0531   21 0000                        LD      HL,0000H
 0534   ED 52                          SBC     HL,DE
 0536   11 0000                        LD      DE,0000H ;SAY DY IS ZERO
 0539   CD 0351                       CALL    LNBYP   ;DRAW FOURTH SIDE
                        +               R
 053C   C9                             RET             ;FINISHED
                        ;************************
                        ;*** SUBROUTINE DCIR(X,Y;R) *** DRAW CIRCLE ***
                        ;*** THIS SUBROUTINE DRAWS A CIRCLE,CENTRE (X,Y) AND OF RADIUS R USING THE
                        ;*** CONTINUOUS LINE STYLE AND CURRENT PEN COLOUR ***
                        ;*** ON ENTRY REGISTER HL POINTS TO X
                        ;*** ON ENTRY REGISTER DE POINTS TO Y
                        ;*** ON ENTRY REGISTER BC POINTS TO R
                        ;*** ON EXIT THE GRAPHIC CURSOR IS POSITIONED AT (X,Y)
 053D   37                      DCIR:   SCF             ;THREE ARGUMENTS
 053E   CD 0909                         CALL    GETPAR  ;GET ARGUMENTS
                        +               R
 0541   CB 78                          BIT     7,B     ;SEE IF -VE RADIUS
 0543   3E 01                          LD      A,01    ;MESSAGE NO
 0545   C4 0950                       CALL    NZ,EMESS ;IF SO THEN ERROR MESSAGE & RETURN
                        +               R
 0548   78                             LD      A,B
 0549   E6 F8                          AND     0F8H    ;SEE IF RADIUS TOO LARGE
 054B   3E 02                          LD      A,02H   ;MESSAGE NO
 054D   C4 0950                       CALL    NZ,EMESS ;IF SO THEN ERROR MESSAGE & RETURN
                        +               R
 0550   DB C2                          IN      A,(CNTL2) ;GET CURRENT LINE STYLE
```

APPENDIX B

```
MV 256 Relocating Graphics Drivers    MACRO-80 3.43  27-Jul-81      PAGE   1-18
Graphics routines


 05A1   CB 1B                         RR      E
 05A3   10 F4                         DJNZ    DCIR4
 05A5   B7                            OR      A       ;CLEAR CARRY
 05A6   ED 52                         SBC     HL,DE   ;COMPUTE Yn+1 LSBs
 05A8   D1                            POP     DE      ;UNSAVE Xn+1 LSBs
 05A9   EB                            EX      DE,HL   ;TO HL
 05AA   D9                            EXX
 05AB   ED 52                         SBC     HL,DE   ;COMPUTE Yn+1 MSBs
 05AD   D1                            POP     DE      ;UNSAVE Xn+1 LSBs
 05AE   EB                            EX      DE,HL   ;TO HL
                                      ;Xn+1 IN HL,HL',Yn+1 IN DE,DE'(32-bit no's)
 05AF   DD E5                         PUSH    IX      ;CIRCLE CENTRE X CO-ORDINATES TO BC
 05B1   C1                            POP     BC
 05B2   E5                            PUSH    HL      ;SAVE Xn+1 MSBs
 05B3   D5                            PUSH    DE      ;SAVE Yn+1 MSBs
 05B4   09                            ADD     HL,BC   ;COMPUTE X DISPLAY CO-ORDINATES
 05B5   EB                            EX      DE,HL   ;TO DE
 05B6   FD E5                         PUSH    IY      ;CIRCLE CENTRE Y CO-ORDINATES TO BC
 05B8   C1                            POP     BC
 05B9   09                            ADD     HL,BC   ;COMPUTE Y DISPLAY CO-ORDINATES
 05BA   EB                            EX      DE,HL   ;TO DE
 05BB   47                            LD      B,A     ;SAVE ERROR DIVISION FACTOR
 05BC   7C                            LD      A,H     ;SEE IF POINT OUT SIDE THE DISPLAY WINDOW
 05BD   B2                            OR      D
 05BE   20 0A                         JR      NZ,DCIR5 ;JUMP IF SO
 05C0   7D                            LD      A,L     ;ELSE PLOT THIS POINT ON CIRCLE EDGE
 05C1   D3 C9                         OUT     (XCL),A
 05C3   7B                            LD      A,E
 05C4   D3 CB                         OUT     (YCL),A
 05C6   3E 80                         LD      A,80H   ;MV256 PLOT POINT COMMAND
 05C8   D3 C0                         OUT     (CMD),A ;NO WAIT IS NECESSARY SINCE REST OF ROUTINE ) 64uS
 05CA   D1                  DCIR5:    POP     DE      ;UNSAVE
 05CB   E1                            POP     HL
 05CC   7C                            LD      A,H     ;SEE IF COMPUTED X IS -VE
 05CD   E6 08                         AND     08H
 05CF   78                            LD      A,B
 05D0   20 AB                         JR      NZ,DCIR2 ;JUMP TO NEXT DOT ON CIRCLE IF -VE
 05D2   7A                            LD      A,D     ;ELSE SEE IF COMPUTED Y IS ZERO
 05D3   B3                            OR      E
 05D4   78                            LD      A,B
 05D5   20 A6                         JR      NZ,DCIR2 ;JUMP TO NEXT DOT ON CIRCLE IF NOT
 05D7   DD E5                         PUSH    IX      ;REPOSITION GRAPHIC CURSOR
 05D9   E1                            POP     HL      ;AT SOME DEFINITE POINT BEFORE FINISHING
 05DA   FD E5                         PUSH    IY
 05DC   D1                            POP     DE
 05DD   F1                            POP     AF      ;UNSAVE ORIGINAL LINE STYLE
 05DE   D3 C2                         OUT     (CNTL2),A
 05E0   CD 03DC                       CALL    MVTOP   ;MOVE TO CENTRE OF CIRCLE
                          +           R
 05E3   C9                            RET             ;FINISHED
                                ;************************
                                ;*** SUBROUTINE DTRI(X,Y;X1,Y1;X2,Y2) *** DRAW TRIANGLE ***
                                ;*** THIS SUBROUTINE DRAWS A TRIANGLE FORMED BY THE VERTICES (X,Y;X1,Y1;X2,Y2)
                                ;*** USING THE CURRENT LINE STYLE AND PEN COLOUR ***
                                ;*** ON ENTRY REGISTER HL POINTS TO X
                                ;*** ON ENTRY REGISTER DE POINTS TO Y
                                ;*** ON ENTRY REGISTER BC INDIRECTLY POINTS TO (X1,Y1;X2,Y2)
                                ;*** ON EXIT THE GRAPHIC CURSOR IS POSITIONED AT (X,Y)
```

APPENDIX B

```
MV 256 Relocating Graphics Drivers     MACRO-80 3.43  27-Jul-81      PAGE   1-20
Graphics routines

 0623   0A                          LD      A,(BC)
 0624   57                          LD      D,A     ;Y MSBs
 0625   CD 030C                     CALL    MVTOP   ;MOVE TO (X,Y)
                    +                R
 0628   E5                          PUSH    HL      ;SAVE START CO-ORDINATES
 0629   D5                          PUSH    DE
 062A   08                          EX      AF,AF'  ;ADJUST NUMBER OF VERTICES
 062B   3D                          DEC     A
 062C   08                  DPOLY1: EX      AF,AF'  ;SAVE AS LOOP COUNTER
 062D   03                          INC     BC
 062E   0A                          LD      A,(BC)  ;NEXT X CO-ORDINATE
 062F   6F                          LD      L,A     ;LSBs
 0630   03                          INC     BC
 0631   0A                          LD      A,(BC)
 0632   67                          LD      H,A     ;MSBs
 0633   03                          INC     BC
 0634   0A                          LD      A,(BC)  ;NEXT Y CO-ORDINATE
 0635   5F                          LD      E,A     ;LSBs
 0636   03                          INC     BC
 0637   0A                          LD      A,(BC)
 0638   57                          LD      D,A     ;MSBs
 0639   CD 031D                     CALL    LNTOP   ;DRAW LINE
                    +                R
 063C   08                          EX      AF,AF'  ;SEE IF LAST BUT ONE LINE
 063D   3D                          DEC     A
 063E   20 EC                       JR      NZ,DPOLY1 ;IF NOT THEN NEXT LINE
 0640   D1                          POP     DE      ;UNSAVE Y START CO-ORDINATE
 0641   E1                          POP     HL      ;UNSAVE X START CO-ORDINATE
 0642   CD 031D                     CALL    LNTOP   ;DRAW LAST LINE OF POLYGON (JOIN XN-1,YN-1 TO X,Y)
                    +                R
 0645   C9                          RET             ;FINISHED
                        ;************************
                        ;*** SUBROUTINE FBOX(X,Y;DX,DY) *** FILL BOX ***
                        ;*** THIS SUBROUTINE FILLS A RECTANGULAR BOX OF SIDES DX & DY STARTING AT (X,Y)
                        ;*** USING THE CURRENT LINE STYLE AND PEN COLOUR ***
                        ;*** ON ENTRY REGISTER HL POINTS TO X
                        ;*** ON ENTRY REGISTER DE POINTS TO Y
                        ;*** ON ENTRY REGISTER BC INDIRECTLY POINTS TO (DX,DY)
                        ;*** ON EXIT THE GRAPHIC CURSOR IS POSITIONED AT (X,Y)
 0646   B7                  FBOX:   OR      A       ;MORE THAN THREE ARGUMENTS
 0647   CD 0909                     CALL    GETPAR  ;GET ARGUMENTS
                    +                R
 064A   CD 09F4                     CALL    ZEROES  ;CLEAR FILL WORKSPACE
                    +                R
 064D   CD 093B                     CALL    TRUN    ;TRUNCATE (X,Y) START CO-ORDINATES
                    +                R
 0650   C5                          PUSH    BC      ;DATA POINTER TO IX
 0651   DD E1                       POP     IX
 0653   CD 0656                     CALL    FBOX1   ;FIND WHERE WE ARE IN MEMORY
                    +                R
 0656   FD E1               FBOX1:  POP     IY      ;PC TO IY
 0658   D9                          EXX
 0659   DD 5E 02                    LD      E,(IX+2) ;DY TO DE
 065C   DD 56 03                    LD      D,(IX+3)
 065F   01 131B                     LD      BC,131BH ;PROGRAMME MOD CODES
 0662   CB 7A                       BIT     7,D     ;TEST SIGN OF DY
 0664   28 0A                       JR      Z,FBOX2 ;JUMP IF +VE
 0666   01 1B13                     LD      BC,1B13H ;PROGRAMME MOD CODES
```

APPENDIX B

```
MV 256 Relocating Graphics Drivers   MACRO-80 3.43  27-Jul-81      PAGE   1-22
Graphics routines

 06B6   FD E1                        POP     IY      ;Y
 06B8   60                           LD      H,B     :CIRCLE DRAWING START POINT (X+R,Y)
 06B9   69                           LD      L,C
 06BA   11 0000                      LD      DE,0000H
 06BD   D9                           EXX
 06BE   21 0000                      LD      HL,0000H :INITIALISE LSBs OF X
 06C1   11 0000                      LD      DE,0000H :INITIALISE LSBs OF Y
 06C4   D9                           EXX
 06C5   AF                           XOR     A       :CLEAR ACC
 06C6   08                           EX      AF,AF'
 06C7   08              FCIR1:       EX      AF,AF'
 06C8   3C                           INC     A       :WORK OUT ERROR DIVISION FACTOR,E=2^-n
                                                     :WHERE 2^n-1<R<2^n
 06C9   CB 38                        SRL     B       :DIVIDE R BY 2
 06CB   CB 19                        RR      C
 06CD   08                           EX      AF,AF'
 06CE   78                           LD      A,B
 06CF   B1                           OR      C       :SEE IF R IS ZERO YET:
 06D0   20 F5                        JR      NZ,FCIR1 ;IF NOT THEN DIVIDE AGAIN
 06D2   08                           EX      AF,AF'  ;ERROR DIVISION FACTOR IN A
 06D3   D5              FCIR2:       PUSH    DE      ;SAVE Yn MSBs
 06D4   D9                           EXX
 06D5   D5                           PUSH    DE      ;SAVE Yn LSBs
 06D6   47                           LD      B,A
 06D7   D9              FCIR3:       EXX
 06D8   CB 2A                        SRA     D       ;DIVIDE Yn BY 2
 06DA   CB 1B                        RR      E
 06DC   D9                           EXX
 06DD   CB 1A                        RR      D
 06DF   CB 1B                        RR      E
 06E1   10 F4                        DJNZ    FCIR3
 06E3   19                           ADD     HL,DE   :COMPUTE Xn+1 LSBs
 06E4   D1                           POP     DE      :UNSAVE Yn LSBs
 06E5   EB                           EX      DE,HL   ;TO HL
 06E6   D9                           EXX
 06E7   ED 5A                        ADC     HL,DE   :COMPUTE Xn+1 MSBs
 06E9   D1                           POP     DE      :UNSAVE Yn MSBs
 06EA   EB                           EX      DE,HL   ;TO HL
                                     :Xn+1 IN DE,DE'
 06EB   D5                           PUSH    DE      :SAVE Xn+1 MSBs
 06EC   D9                           EXX
 06ED   D5                           PUSH    DE      ;SAVE Xn+1 LSBs
 06EE   47                           LD      B,A
 06EF   D9              FCIR4:       EXX
 06F0   CB 2A                        SRA     D       :DIVIDE Xn+1 by 2
 06F2   CB 1B                        RR      E
 06F4   D9                           EXX
 06F5   CB 1A                        RR      D
 06F7   CB 1B                        RR      E
 06F9   10 F4                        DJNZ    FCIR4
 06FB   B7                           OR      A       :CLEAR CARRY
 06FC   ED 52                        SBC     HL,DE   :COMPUTE Yn+1 LSBs
 06FE   D1                           POP     DE      :UNSAVE Xn+1 LSBs
 06FF   EB                           EX      DE,HL   ;TO HL
 0700   D9                           EXX
 0701   ED 52                        SBC     HL,DE   :COMPUTE Yn+1 MSBs
 0703   D1                           POP     DE      ;UNSAVE Xn+1 LSBs
 0704   EB                           EX      DE,HL   ;TO HL
```

APPENDIX B

```
MV 256 Relocating Graphics Drivers      MACRO-80 3.43  27-Jul-81      PAGE   1-24
Graphics routines

                        ;*** (X,Y;X1,Y1;.. ...XN-1,YN-1) USING THE CURRENT LINE STYLE AND PEN COLOUR **
                        ;*** ON ENTRY REGISTER HL POINTS TO N,THE NUMBER OF VERTICES (max 16)
                        ;*** ON ENRTY REGISTER DE POINTS TO X
                        ;*** ON ENTRY REGISTER BC INDIRECTLY POINTS TO (Y;X1,Y1;..      ...XN-1,YN-1)
                        ;*** ON EXIT THE GRAPHIC CURSOR IS POSITIONED AT (X,Y)
0743   B7               FPOLY:  OR      A       ;MORE THAN THREE ARGUMENTS
0744   CD 0909                  CALL    GETPAR  ;GET ARGUMENTS
                    +           R
0747   CD 09F4                  CALL    ZEROES  ;CLEAR FILL WORKSPACE
                    +           R
074A   7D                       LD      A,L     ;NUMBER OF VERTICES TO ACC
074B   D6 02                    SUB     02H
074D   3E 03                    LD      A,03H   ;MESSAGE NUMBER
074F   CC 0950                  CALL    Z,EMESS ;ERROR MESSAGE AND RETURN IF N=2
                    +           R
0752   DC 0950                  CALL    C,EMESS ;ERROR MESSAGE AND RETURN IF N=0 OR 1
                    +           R
0755   CB 7C                    BIT     7,H
0757   C4 0950                  CALL    NZ,EMESS ;ERROR MESSAGE AND RETURN IF N IS -VE
                    +           R
075A   7D                       LD      A,L     ;NUMBER OF VERTICES TO ACC
075B   C5                       PUSH    BC      ;DATA POINTER TO IX
075C   DD E1                    POP     IX
075E   DD 6E 00                 LD      L,(IX)  ;GET Y START CO-ORDINATE
0761   DD 23                    INC     IX
0763   DD 66 00                 LD      H,(IX)
0766   DD 23                    INC     IX
0768   EB                       EX      DE,HL   ;X TO HL,Y TO DE
0769   D5                       PUSH    DE      ;SAVE POLYGON START CO-ORDINATES
076A   E5                       PUSH    HL
076B   E5                       PUSH    HL
076C   D9                       EXX
076D   D1                       POP     DE
076E   F5               FPOLY1: PUSH    AF      ;SAVE AS LOOP COUNTER
076F   DD 6E 00                 LD      L,(IX)  ;GET NEXT CO-ORDINATE
0772   DD 23                    INC     IX
0774   DD 66 00                 LD      H,(IX)  ;X MSBs
0777   D9                       EXX
0778   DD 23                    INC     IX
077A   DD 6E 00                 LD      L,(IX)  ;Y LSBs
077D   DD 23                    INC     IX
077F   DD 66 00                 LD      H,(IX)  ;Y MSBs
0782   DD 23                    INC     IX
0784   CD 093B          FPOLY2: CALL    TRUN    ;TRUNCATE CO-ORDINATES IN HL & DE REGISTERS
                    +           R
0787   AF                       XOR     A       ;CLEAR CARRY
0788   ED 52                    SBC     HL,DE   ;COMPUTE DY
078A   D5                       PUSH    DE      ;SAVE NEXT LINE Y START CO-ORDINATE
078B   3E 03                    LD      A,03H   ;VIGIN MODIFICTION CODE (Z80 INC INSTRUCTION)
078D   CB 7C                    BIT     7,H
078F   28 08                    JR      Z,FPOLY3 ;JUMP IF DY +VE
0791   F6 08                    OR      08H     ;FORM Z80 DEC INSTRUCTION
0793   EB                       EX      DE,HL   ;DO TWOS COMP ON DY
0794   21 0000                  LD      HL,0000H
0797   ED 52                    SBC     HL,DE
0799   08               FPOLY3: EX      AF,AF'  ;SAVE Z80 INSTRUCTION
079A   D9                       EXX             ;TO X REGISTERS
079B   CD 093B                  CALL    TRUN    ;TRUNCATE CO-ORDINATES IN HL & DE REGISTERS
```

APPENDIX B

```
MV 256 Relocating Graphics Drivers     MACRO-80 3.43  27-Jul-81      PAGE   1-26
Graphics routines

 07F4   19                          ADD     HL,DE   ;COMPUTE NEW ERROR TERM
 07F5   D9                          EXX             ;TO X,Y REGISTERS
 07F6                               DS      1       ;INC/DEC HL OR DE ACCORDING TO VECTOR DIRECTION
 07F7   18 EC                       JR      FPOLY7  ;NEXT DOT ON POLYGON EDGE
 07F9   F1                  FPOLY9: POP     AF      ;LINE LOOP COUNTER
 07FA   3D                          DEC     A
 07FB   28 0F                       JR      Z,FPOLYA ;JUMP TO FINISH IF LAST LINE JUST DONE
 07FD   FE 01                       CP      01H     ;SEE IF LAST BUT ONE LINE
 07FF   E5                          PUSH    HL      ;SAVE X
 0800   D9                          EXX
 0801   D1                          POP     DE
 0802   C2 076E                     JP      NZ,FPOLY1 ;IF NOT THEN NEXT LINE
                    +               R
 0805   E1                          POP     HL      ;POLYGON X START CO-ORDINATE
 0806   D9                          EXX
 0807   E1                          POP     HL      ;POLYGON Y START CO-ORDINATE
 0808   F5                          PUSH    AF      ;SAVE LINE LOOP COUNTER
 0809   C3 0784                     JP      FPOLY2  ;DRAW LAST LINE OF POLYGON
                    +               R
 080C   CD 03DC             FPOLYA: CALL    MVTOP   ;MOVE TO (X,Y)
                    +               R
 080F   C9                          RET             ;FINISHED
                            ;************************
                            ;*** SUBROUTINE CLEARS *** CLEAR SCREEN ***
                            ;*** THIS SUBROUTINE CLEARS THE SCREEN ***
                            ;*** ALL Z80 REGISTERS SAVED
 0810   F5                  CLEARS: PUSH    AF      ;SAVE REGISTERS
 0811   CD 08E0                     CALL    READY   ;WAIT FOR MV256 READY
                    +               R
 0814   3E 04                       LD      A,04H   ;MV256 COMMAND CODE
 0816   D3 C0                       OUT     (CMD),A
 0818   F1                          POP     AF      ;UNSAVE REGISTERS
 0819   C9                          RET             ;FINISHED
                            ;************************
                            ;*** SUBROUTINE SCAN *** SET SCREEN ***
                            ;*** THIS SUBROUTINE SETS THE WHOLE SCREEN TO THE CURRENT PEN COLOUR ***
                            ;*** ALL Z80 REGISTERS ARE SAVED
                            ;*** THE GRAPHIC CURSOR IS PRESERVED
 081A   F5                  SCAN:   PUSH    AF      ;SAVE REGISTERS
 081B   CD 08E0                     CALL    READY   ;WAIT FOR MV256 READY
                    +               R
 081E   3E 0C                       LD      A,0CH   ;MV256 COMMAND CODE
 0820   D3 C0                       OUT     (CMD),A
 0822   F1                          POP     AF      ;UNSAVE REGISTERS
 0823   C9                          RET             ;FINISHED
                            ;************************
                            ;*** SUBROUTINE LPEN *** LIGHT PEN SEQUENCE ***
                            ;*** THIS SUBROUTINE INITIATES LIGHT PEN SEQUENCES UNTILL A LIGHT PEN STROBE
                            ;*** SIGNAL IS GENERATED.THE COLOUR CODE OF THE PIXEL AT THE PENS POSITION
                            ;*** IS RETURNED IN THE 'GETCOL' REGISTER AND THE Z80 ACCUMULATOR. THE PENS
                            ;*** (X,Y) POSITION IS RETURNED IN HL & DE ***
                            ;*** ALL Z80 REGISTERS EXCEPT HL,DE & A ARE SAVED
                            ;*** ON EXIT THE GRAPHIC CURSOR IS POSITIONED AT THE LIGHT PENS SCREEN
                            ;*** CO-ORDINATES (errors compensated for)
 0824   CD 08C1             LPEN:   CALL    RESGC   ;RESET GRAPHIC CURSOR
                    +               R
 0827   3E 14                       LD      A,14H   ;WAIT FOR 20 VIDEO FRAMES
 0829   CD 08E9                     CALL    FWAIT   ;THIS PREVENTS MULTIPLE DATA ENTRIES
```

APPENDIX B

```
MV 256 Relocating Graphics Drivers     MACRO-80 3.43  27-Jul-81      PAGE   1-28
Graphics routines

 0876   C9                          RET             ;FINISHED
                        ;************************
                        ;*** SUBROUTINE CHRSTY(S) *** SELECT CHARACTER STYLE ***
                        ;*** THIS SUBROUTINE SELECTS A NEW CHARACTER DRAWING STYLE ***
                        ;*** ON ENTRY REGISTER HL POINTS TO THE REQUIRED STYLE CODE
                                ;FOR STRAIGHT CHARACTER ON HORIZONTAL AXIS CODE=0000H
                                ;FOR TILTED CHARACTER ON HORIZONTAL AXIS   CODE=0001H
                                ;FOR STRAIGHT CHARACTER ON VERTICAL AXIS   CODE=0002H
                                ;FOR TILTED CHARACTER ON VERTICAL AXIS     CODE=0003H
 0877   37                  CHRSTY: SCF     ;LESS THAN THREE ARGUMENTS
 0878   CD 0909                     CALL    GETPAR  ;GET ARGUMENT
                    +               R
 087B   7D                          LD      A,L     ;CLEAR ANY UNWANTED BITS
 087C   E6 03                       AND     03H
 087E   17                          RLA             ;FORM CHARACTER STYLE CODE FOR MV256
 087F   17                          RLA
 0880   6F                          LD      L,A
 0881   CD 08E0                     CALL    READY   ;WAIT FOR MV256 READY
                    +               R
 0884   DB C2                       IN      A,(CNTL2) ;GET PREVIOUS CHARACTER STYLE FROM MV256
 0886   E6 F3                       AND     0F3H    ;CLEAR IT
 0888   B5                          OR      L       ;NEW CHARACTER STYLE
 0889   D3 C2                       OUT     (CNTL2),A
 088B   C9                          RET             ;FINISHED
                        ;************************
                        ;*** SUBROUTINE CHRSIZ(P,Q) *** SELECT CHARACTER SIZE ***
                        ;*** THIS SUBROUTINE SELECTS NEW X & Y SCALING FACTORS FOR CHRARACTERS ***
                        ;*** ON ENTRY REGISTER L HOLDS P,THE X SCALING FACTOR
                        ;*** ON ENTRY REGISTER E HOLDS Q,THE Y SCALING FACTOR
                                ;MIN X SCALE FACTOR CODE=0001H (X SCALE MULTIPLIER=1)
                                ;MAX X SCALE FACTOR CODE=0010H (X SCALE MULTIPLIER=16)
                                ;MIN Y SCALE FACTOR CODE=0001H (Y SCALE MULTIPLIER=1)
                                ;MAX Y SCALE FACTOR CODE=0010H (Y SCALE MULTIPLIER=16)
                                ;SCALE FACTOR CODES 0000H & 0010H ARE EQUIVALENT
 088C   37                  CHRSIZ: SCF             ;LESS THAN THREE ARGUMENTS
 088D   CD 0909                     CALL    GETPAR  ;GET ARGUMENTS
                    +               R
 0890   CB 25                       SLA     L       ;CONVERT TO REQUIRED MV256 CODE
 0892   CB 25                       SLA     L
 0894   CB 25                       SLA     L
 0896   CB 25                       SLA     L
 0898   7B                          LD      A,E     ;GET Y SCALE FACTOR
 0899   E6 0F                       AND     0FH     ;CLEAR ANY MSBs
 089B   B5                          OR      L       ;ADD X SCALE FACTOR TO MSBs OF CODE
 089C   CD 08E0                     CALL    READY   ;WAIT FOR MV256 READY
                    +               R
 089F   D3 C3                       OUT     (CSIZE),A ;PUT NEW CHARACTER SIZE TO MV256
 08A1   C9                          RET             ;FINISHED
                        ;************************
                        ;*** SUBROUTINE PENSEL *** SELECT MV256 PEN ***
                        ;*** THIS SUBROUTINE SELECTS THE MV256 PEN ***
                        ;*** ALL Z80 REGISTERS ARE SAVED
 08A2   F5                  PENSEL: PUSH    AF      ;SAVE REGISTERS
 08A3   CD 08E0                     CALL    READY   ;WAIT FOR MV256 READY
                    +               R
 08A6   AF                          XOR     A       ;SELECT PEN
 08A7   D3 C0                       OUT     (CMD),A
 08A9   F1                          POP     AF      ;UNSAVE REGISTERS
```

APPENDIX B

```
MV 256 Relocatimo Graonics Drivers     MACRO-80 3.43  27-Jul-81      PAGE    1-30
Graonics routines

                        ;*** ALL Z80 REGISTERS SAVED
08E0    F5              READY:  PUSH    AF      ;SAVE REGISTERS
08E1    DB C0           READY1: IN      A,(STAT) ;GET MV256 STATUS
08E3    E6 04                   AND     04H     ;SEE IF READY (BIT 2)
08E5    28 FA                   JR      Z,READY1 ;IF NOT THEN TEST AGAIN
08E7    F1                      POP     AF      ;UNSAVE REGISTERS
08E8    C9                      RET             ;FINISHED
                        ;************************
                        ;*** SUBROUTINE FWAIT(N) *** FRAME WAIT ***
                        ;*** THIS SUBROUTINE WAITS FOR N VIDEO FRAMES ***
                        ;*** ON ENTRY REGISTER A HOLDS N
                        ;*** ALL Z80 REGISTERS SAVED
08E9    F5              FWAIT:  PUSH    AF      ;SAVE REGISTERS
08EA    C5                      PUSH    BC
08EB    B7                      OR      A       ;TEST TO SEE IF N=0
08EC    28 0F                   JR      Z,FWAIT3 ;JUMP IF N=0
08EE    47                      LD      B,A
08EF    DB C0           FWAIT1: IN      A,(STAT) ;GET MV256 STATUS
08F1    E6 02                   AND     02H     ;SEE IF VERTICAL VIDEO BLANKING (BIT 1)
08F3    28 FA                   JR      Z,FWAIT1 ;IF NOT THEN TEST AGAIN
08F5    DB C0           FWAIT2: IN      A,(STAT) ;GET MV256 STATUS
08F7    E6 02                   AND     02H     ;SEE IF VERTICAL VIDEO BLANKING (BIT 1)
08F9    20 FA                   JR      NZ,FWAIT2 ;IF SO THEN TEST AGAIN
08FB    10 F2                   DJNZ    FWAIT1  ;DO FOR N FRAMES
08FD    C1              FWAIT3: POP     BC      ;UNSAVE REGISTERS
08FE    F1                      POP     AF
08FF    C9                      RET             ;FINISHED
                        ;************************
                        ;*** SUBROUTINE VBLANK *** VIDEO BLANKING ***
                        ;*** THIS SUBROUTINE WAITS FOR VERTICAL VIDEO BLANKING TO START ***
                        ;*** ALL Z80 REGISTERS SAVED
0900    F5              VBLANK: PUSH    AF      ;SAVE REGISTERS
0901    DB C0           VBLNK1: IN      A,(STAT) ;GET MV256 STATUS
0903    E6 02                   AND     02H     ;TEST FOR VERTICAL VIDEO BLANKING (BIT 1)
0905    28 FA                   JR      Z,VBLNK1 ;IF NOT THEN TEST AGAIN
0907    F1                      POP     AF      ;UNSAVE REGISTERS
0908    C9                      RET             ;FINISHED
                        ;************************
                        ;*** SUBROUTINE GETPAR *** GET PARAMETERS ***
                        ;*** THIS SUBROUTINE GETS THE ARGUMENTS PASSED BY THE CALL STATEMENT OF
                        ;*** MICROSOFT BASIC.WHEN PROGRAMMING IN ASSEMBLY LANGUAGE THIS SUBROUTINE
                        ;*** CAN BE OMITTED AND ARGUMENTS PASSED IN THE REGISTERS.WHEN USING OTHER
                        ;*** HIGH LEVEL LANGUAGES TO CALL GRAPHIC PRIMITIVES THIS SUBROUTINE MUST BE
                        ;*** MODIFIED TO SUIT THE ARGUMENT TRANSFER SYSTEM ***
                        ;*** ON ENTRY HL REGISTER POINTS TO FIRST ARGUMENT
                        ;*** ON ENTRY DE REGISTER POINTS TO SECOND ARGUMENT
                        ;*** ON ENTRY BC REGISTER POINTS TO THIRD ARGUMENT IF TOTAL OF THREE ARGUMENTS
                        ;*** ELSE POINTS DATA AREA CONTAINING POINTERS TO THIRD AND SUBSEQUENT ARGUMENT
                        ;*** ON ENTRY CARRY FLAG SET IF THREE OR LESS ARGUMENTS
                        ;*** ON ENTRY CARRY FLAG RESET IF MORE THAN THREE ARGUMENTS
                        ;*** ON EXIT HL REGISTER CONTAINS THE FIRST ARGUMENT
                        ;*** ON EXIT DE REGISTER CONTAINS THE SECOND ARGUMENT (IF ANY)
                        ;*** ON EXIT BC REGISTER CONTAINS THE THIRD ARGUMENT (IF ANY) IF TOTAL OF
                        ;*** THREE ARGUMENTS ELSE REGISTER POINTS TO THIRD AND SUBSEQUENT ARGUMENTS.
                        ;*** CALLED BY ALL PRIMITIVES WITH ARGUMENTS
0909    D5              GETPAR: PUSH    DE      ;SAVE SECOND ARGUMENT POINTER
090A    C5                      PUSH    BC      ;IY POINTS TO PARAMETERS(POINTERS) PASSED BY BASIC CALL
090B    FD E1                   POP     IY      ;IF MORE THAN THREE ARGUMENTS.
```

APPENDIX B

```
MV 256 Relocatimo Graphics Drivers     MACRO-80 3.43  27-Jul-81      PAGE   1-32
Graphics routines

                          +            R
0955    CD 08E0                        CALL    READY
                          +            R
0958    3E 06                          LD      A,06H   ;SELECT CHARACTER SIZE
095A    D3 C3                          OUT     (CSIZE),A
095C    3E 00                          LD      A,00H   ;SELECT CHARACTER STYLE
095E    D3 C2                          OUT     (CNTL2),A
0960    3E 06                          LD      A,6     ;SELECT YELLOW PEN COLOUR
0962    D3 D0                          OUT     (PENCOL),A
0964    3E 0B                          LD      A,0BH
0966    06 04                          LD      B,04H
0968    D3 C0                EMESS1:   OUT     (CMD),A
096A    CD 08E0                        CALL    READY
                          +            R
096D    10 F9                          DJNZ    EMESS1
096F    3E 00                          LD      A,00H   ;SELECT BLACK PEN COLOUR
0971    D3 D0                          OUT     (PENCOL),A
0973    3E 12                          LD      A,12H   ;SELECT CHARACTER SIZE
0975    D3 C3                          OUT     (CSIZE),A
0977    08                             EX      AF,AF'  ;UNSAVE ERROR MESSAGE NUMBER
0978    06 20                          LD      B,32    ;MESSAGE LENGTH
097A    11 0996                        LD      DE,ERROR1 ;POINT TO MESSAGE START
                          +            R
097D    FE 01                          CP      01H
097F    28 0E                          JR      Z,EMESS2
0981    06 24                          LD      B,36
0983    11 09B6                        LD      DE,ERROR2
                          +            R
0986    FE 02                          CP      02H
0988    28 05                          JR      Z,EMESS2
098A    06 1A                          LD      B,26
098C    11 09DA                        LD      DE,ERROR3
                          +            R
098F    CD 08C1              EMESS2:   CALL    RESGC   ;RESET GRAPHIC CURSOR TO (0,0)
                          +            R
0992    CD 0469                        CALL    TXTTO1  ;PLOT ERROR MESSAGE
                          +            R
0995    C9                             RET             ;RETURN TO MAIN CALLING ROUTINE (BASIC IF BEING USED)
0996    20 45 52 52          ERROR1:   DB      " ERROR 1: NEGITIVE CIRCLE RADIUS"
099A    4F 52 20 31
099E    3A 20 4E 45
09A2    47 49 54 49
09A6    56 45 20 43
09AA    49 52 43 4C
09AE    45 20 52 41
09B2    44 49 55 53
09B6    20 45 52 52          ERROR2:   DB      " ERROR 2: CIRCLE RADIUS OUT OF RANGE"
09BA    4F 52 20 32
09BE    3A 20 43 49
09C2    52 43 4C 45
09C6    20 52 41 44
09CA    49 55 53 20
09CE    4F 55 54 20
09D2    4F 46 20 52
09D6    41 4E 47 45
09DA    20 45 52 52          ERROR3:   DB      " ERROR 3: TOO FEW VERTICES"
09DE    4F 52 20 33
09E2    3A 20 54 4F
```

APPENDIX B

```
MV 256 Relocating Graphics Drivers     MACRO-80 3.43  27-Jul-81      PAGE   1-34
Graphics routines

 0A40   30 04                          JR      NC,FILL3 ;IF DX +VE THEN JUMP
 0A42   ED 44                          NEG             ;TWOS COMP ON DX
 0A44   0E 16                          LD      C,16H   ;MV256 COMMAND CODE FOR -VE DX
 0A46   CD 08E0             FILL3:     CALL    READY   ;WAIT FOR MV256 READY
                      +                R
 0A49   D3 C5                          OUT     (DX),A  ;VECTOR LENGTH TO MV256 DELTAX REGISTER
 0A4B   78                             LD      A,B
 0A4C   D3 C9                          OUT     (XCL),A
 0A4E   7B                             LD      A,E
 0A4F   D3 CB                          OUT     (YCL),A
 0A51   79                             LD      A,C     ;DRAW VECTOR COMMAND CODE
 0A52   D3 C0                          OUT     (CMD),A
 0A54   FD 70 01            FILL4:     LD      (IY+1),B ;NEW POLYGON EDGE START/FINISH X VALUE
 0A57   C1                  FILL5:     POP     BC      ;UNSAVE REGISTERS
 0A58   C9                             RET             ;NEXT DOT ON SHAPE
                           ;*************************************************************************
 0A59                      WORKSP:    DS      512     ;512 BYTES WORK SPACE FOR FILL SUBROUTINE
                           ;************************
                           ;************************
 0C59                      ARGDAT:    DS      64      ;64 BYTES ARGUMENT BUFFER FOR CALL ROUTINE
                           ;*************************************************************************
 0C99                               MVEND   EQU $
                                    SUBTTL Worksoace
                           ;*****************************************************
                           :************ WORKSPACE FOR LINK ROUTINES *************
                           ;*****************************************************

 0C99   2710 03E8           P10TAB:    DW 10000,1000,100,10,1
 0C9D   0064 000A
 0CA1   0001

 0CA3   0000                REL:       DW 0

 0CA5   0D 0A 0A 20         SIGNON:    DB CR,LF,LF," *** MV256 graphics primitive "
 0CA9   2A 2A 2A 20
 0CAD   4D 56 32 35
 0CB1   36 20 67 72
 0CB5   61 70 68 69
 0CB9   63 73 20 70
 0CBD   72 69 6D 69
 0CC1   74 69 76 65
 0CC5   20
 0CC6   6C 69 6E 6B                 DB "linker V0.1 ***",CR,LF,LF,"$"
 0CCA   65 72 20 56
 0CCE   30 2E 31 20
 0CD2   2A 2A 2A 0D
 0CD6   0A 0A 24

 0CD9   42 61 73 65         ADDRM:     DB "Base address of routines = "
 0CDD   20 61 64 64
 0CE1   72 65 73 73
 0CE5   20 6F 66 20
 0CE9   72 6F 75 74
 0CED   69 6E 65 73
 0CF1   20 3D 20
 0CF4   30 30 30 30         HADDR:     DB "0000 Hex -- "
 0CF8   20 48 65 78
 0CFC   20 2D 2D 20
```

APPENDIX B

```
MV 256 Relocating Graphics Drivers    MACRO-80 3.43   27-Jul-81      PAGE    5
Workspace

Macros:
.LAB            .TAB            R

Symbols:
0000    .R              01CA    .R1             02A7    .R10
0606    .R100           060B    .R101           0613    .R102
0616    .R103           061B    .R104           0626    .R105
063A    .R106           0643    .R107           0648    .R108
064B    .R109           02AA    .R11            064E    .R110
0654    .R111           067B    .R112           068E    .R113
069B    .R114           06A2    .R115           06AA    .R116
06AD    .R117           06B0    .R118           0715    .R119
02AD    .R12            072D    .R120           0732    .R121
0735    .R122           0745    .R123           0748    .R124
0750    .R125           0753    .R126           0758    .R127
0785    .R128           079C    .R129           02B0    .R13
07C5    .R130           07E6    .R131           0803    .R132
080A    .R133           080D    .R134           0812    .R135
081C    .R136           0825    .R137           082A    .R138
0831    .R139           02B3    .R14            0854    .R140
085B    .R141           085E    .R142           0866    .R143
086D    .R144           0879    .R145           0882    .R146
088E    .R147           089D    .R148           08A4    .R149
02B6    .R15            08AD    .R150           08B7    .R151
08C3    .R152           08CD    .R153           08D4    .R154
0922    .R155           0938    .R156           0953    .R157
0956    .R158           096B    .R159           02B9    .R16
097B    .R160           0984    .R161           098D    .R162
0990    .R163           0993    .R164           09F5    .R165
09FB    .R166           0A12    .R167           0A47    .R168
02BC    .R17            02BF    .R18            02C2    .R19
01D6    .R2             02C5    .R20            02C8    .R21
02CB    .R22            02CE    .R23            02D1    .R24
02D4    .R25            02D7    .R26            02DA    .R27
02DD    .R28            02E0    .R29            01E1    .R3
02E3    .R30            02E6    .R31            02E9    .R32
02EC    .R33            02EF    .R34            02F2    .R35
02F5    .R36            02F8    .R37            02FB    .R38
02FE    .R39            01F3    .R4             0301    .R40
0304    .R41            0307    .R42            030A    .R43
030D    .R44            0310    .R45            0313    .R46
031B    .R47            0321    .R48            0324    .R49
0298    .R5             0347    .R50            034F    .R51
0382    .R52            03C4    .R53            03C7    .R54
03D5    .R55            03DA    .R56            03DD    .R57
03E0    .R58            03F1    .R59            029B    .R6
03F4    .R60            0420    .R61            0423    .R62
0431    .R63            0438    .R64            043B    .R65
0449    .R66            0454    .R67            045B    .R68
045E    .R69            029E    .R7             046C    .R70
0475    .R71            0478    .R72            0486    .R73
048F    .R74            0492    .R75            04A2    .R76
04A5    .R77            04B2    .R78            04C6    .R79
02A1    .R8             04CE    .R80            04D1    .R81
04D6    .R82            04E1    .R83            04F3    .R84
04FC    .R85            04FF    .R86            0514    .R87
051D    .R88            052C    .R89            02A4    .R9
053A    .R90            053F    .R91            0546    .R92
054E    .R93            0557    .R94            055A    .R95
05E1    .R96            05E6    .R97            05EB    .R98
```