CFX-II Video Modes

Bill Brendling
17 May 2020

Introduction

The VGA output from the CFX-II board is produced by a Parallax Propeller microcontroller chip. This supports a number of different operating modes:

- 80 x 24 text modes with 160 x 96 graphics, largely similar to the Memotech 80 column card:
  - 16 colour mode compatible with the Memotech card using colour output.
  - Monochrome (green) mode compatible with the Memotech card using monochrome output.
  - Enhanced 64 colour mode.
- 320 x 240 graphics mode with 40 x 24 text:
  - 2 out of 64 colours per character cell.
- MTX VDP emulation mode:
  - Text mode: 40 x 24 characters.
  - Graphics mode I: 256 x 192 graphics, 2 out of 16 colours per 16 character cells.
  - Graphics mode II: 256 x 192 graphics, 2 out of 16 colours per character cell.
  - Multi-colour mode: 80 x 48 independent colour tiles.

The VDP emulation modes shadow the hardware interface of the VDP, reproducing what is displayed on the composite video output. The 80 column modes are software compatible with the original Memotech card, but with a different hardware interface.

BASIC Support

By default, when in Basic mode, the CFX VGA output echoes the output produced by the original VDP output. However the latest CFX ROM allows use of some of the other capabilities of the VGA display from within Basic. It does this by introducing two new screen types that can be used with the CRVS command:

- **Type 3:**
  - 80 columns x 24 rows of cells
  - Each cell is 8 x 20 VGA pixels
  - Each character cell contains 2 x 4 plot points (each plot point being 4 x 5 VGA pixels). A total of 160 x 96 plot points on the screen.
  - A cell may contain a character, or plot points, but not both.
  - Each cell has two colours, foreground and background.
- **Type 2:**
  - 40 columns x 24 rows of cells
  - Each cell is 16 x 20 VGA pixels, but individual pixels cannot be controlled
  - Each cell contains 8 x 10 plot points (each plot point being 2 x 2 VGA pixels). A total of 320 x 240 plot points on the screen.
  - Plotting may go over characters.
  - Each cell has two colours, foreground and background.

As with the original type 0 (text) and type 1 (graphics) screens, the CRVS command can be used to define multiple virtual screens with different sizes and positions within a screen type. It should be noted that while multiple virtual screens may be shown at the same time, they must all be of the same screen type. This is not a new restriction, the original Memotech cannot display a type 0 (text) and a type 1 (graphics) screen at the same time.

It should be noted that the CRVS command also has the effect of selecting the newly created virtual screen. This is not new, it has always been the case for Memotech Basic, but less obvious with only type 0 and type 1 screens. When a type 2 or type 3 screen is selected, the VGA monitor will flicker for a few seconds as the Propeller changes out of VDP emulation mode and into the new mode.

A number of the Basic commands behave differently with the new screen types. There are also a few new USER commands to support these screen types. The remainder of this section outlines the differences.

Sprites

Neither type 2 nor type 3 screens support sprites. Therefore any of the sprite commands will produce an error if used with these screen types.

Plotting

Both type 2 and type 3 screens support plotting, and all the plotting commands may be used on either screen type.

For a type 3 screen the resolution is low, and plotting over a character replaces the character at that location.

Type 2 screens have a higher resolution, and plotting at a character position will draw over the character. The existing plotting commands allow a maximum x-coordinate of 255, whereas a type 2 VS may be up to 320 plot-points wide (maximum x-coordinate of 319). There are two ways of working around this limitation. The plot coordinates are measured from the bottom left hand corner of the virtual screen, so by adjusting the position of this it is possible to plot at any position on the display.

Secondly, there are two new USER commands:

    USER PLOT x, y

    USER LINE x1, y1, x2, y2

which can be used for type 2 screens. They work the same way as the standard PLOT and LINE commands, except that they permit x-coordinates up to 319.

It should also be noted that each character cell may only have two colours, one foreground and one background. So plotting at a location may change the colour of other points in the same character cell. The original Memotech type 1 screen has a similar limitation but it may be less obvious as it applies to single rows of 8 pixels.

ATTR command

The four attributes work, as far as possible, the same way on type 2 and type 3 screens as they do on type 1.

Attribute 1: Inverse print – Characters are printed in the paper colour on a background of the ink colour.

Attribute 2: Over print – On type 2 screens new characters are printed on top of any existing character. This is not possible on type 3 screens so this attribute has no effect on type 3 screens.

Attribute 3: Unplot – Removes plotted points, resetting them to the paper colour,

Attribute 4: Overplot – Reverses the state of plotted points, removing those previously plotted (so they become paper colour) or plotting points not previously plotted (so they become ink colour).

Note that turning on both attributes 3 and 4 results in plotting having no effect.

COLOUR command

This works as for type 1 screens, except that there is no border, so setting the border colour has no effect.

GENPAT command

Characters on a type 3 screen occupy 8 x 20 pixels, therefore it needs 20 bytes to define their shape. The GENPAT command only specifies 8 bytes, therefore three GENPAT commands are needed to completely redefine the shape of a character. The format of the command is:

GENPAT p,n,d1,d2,d3,d4,d5,d6,d7,d8

In this command p=0 to redefine rows 1 to 8 of a character, p=1 to redefine rows 9-16 of a character and p=2 to redefine rows 17-20 of the character.

n is the ASCII code of the character to redefine (0-255).

d1 to d8 are the bytes defining successive rows. For p=2, d5 to d8 are ignored, but must be present.

Type 2 screens use the same font as type 3, but only uses every other row to give an 8 x 10 character.

Redefining a character lasts for as long as a type 2 or type 3 screen is selected. Switching to a type 0 or 1 screen looses all the user definitions and characters will have their default shape when a type 2 or 3 screen is selected again.

GR$ function

The function GR$(x,y,n) returns a character whose ASCII value is made up of a column of n (1 to 8) plot-points, starting at the coordinate (x,y).

For type 2 screens, the plot-points may be the result of a character printed at that location, or of a point plotted there.

For type 3 screens, any location containing a printed character is regarded as having no plotted points.

SPK$ function

This function returns the character under the cursor, and advances the cursor.

For type 3 screens it returns the character at that location regardless of any attributes (such as inverse printing) that may have been applied. Any character location containing plotting will return an ASCII NULL (zero) character.

For type 2 screens it works by recognising the plot pattern at that location. It will therefore fail (and return ASCII NULL) if the character has been over-plotted, is inverse, or contains the underline cursor. It will however work for user-defined character shapes providing they are clear.

USER COLOUR command

Memotech Basic supports 16 colours, as generated by the VDP. However the Propeller VGA display is capable of generating 64 colours. By default 16 of these 64 colours which approximate to the VDP are used. However the USER COLOUR command provides the ability to change these. The format of this command is:

USER COLOUR c, r, g, b

Where:

c = Colour to redefine (0 to 15)

r = Red value for this colour (0 to 3)

g = Green value for this colour (0 to 3)

b = Blue value for this colour (0 to 3)

Note: This command will not change the colour of any text that is already displayed. The correct sequence is:

- Use the USER COLOUR command to to define the RGB values for a colour or colours.
- Use the INK and / or PAPER commands to select the newly defined colours.
- Print text.

By repeating this sequence all 64 colours may be displayed at the same time.

USER VGA command

This command enables you to use an 80 column screen for editing programs. The format of this command is:

USER VGA mode

where mode = 1 for 80 column editing, and mode = 0 to return to 40 column editing. The effect of selecting 80 column mode is to change the definitions of virtual screens 0, 1, 5 and 7 to be type 3 screens. As a result program listing and editing uses the 80 column screen type.

The default graphics virtual screen (VS 4) is changed to a type 2 screen. This does mean that sprites cannot be used without redefining the screen.

If 80 column mode is selected, it will also be used for PANEL, although that does not make any use of the extra screen space.

If you have two monitors attached to the Memotech, one on the original composite VDP output and a second on the CFX-II VGA output, then there are two further modes (mode =2 or mode = 3) that may be usefully selected. If these modes are selected, then the CFX-II display does not switch back to VDP echo mode when virtual screens of types 0 or 1 are selected. Instead it continues to display what was written to the type 2 or type 3 virtual screens. Thus it is possible to make use of both monitors by having some virtual screens of type 0 or type 1, which will be displayed on the composite video monitor, and other virtual screens of type 2 or 3 will be displayed on the VGA monitor.

The difference between VGA modes 2 and 3 is that in mode 2, virtual screens 0, 1, 5 and 7 default to type 0, so Basic editing and PANEL will be on the composite monitor, whereas in VGA mode 3 these virtual screens default to type 3, so that Basic editing and PANEL will be in 80 columns on the VGA monitor.

Summary of Control and Escape Codes

The following tables summarises the use of control and escape codes from within MTX Basic, the CP/M driver for the original Memotech 80 column card, and the CFX-II 80 column display. Note that when using the 80 column display from MTX Basic as described in the previous section the MTX ROM processes all the control and escape codes, so the first column of the tables is relevant. The third column is only applicable for CP/M, or bypassing the MTX ROM by writing to port 96 (60 hex).

Control Codes

Hex	Letter	MTX Basic	Memotech 80 Col	CFX-II VGA
0x01	^A	Plots point	Plots point	Plots point
0x02	^B	Plots line	Plots line	Plots line
0x03	^C	Position cursor	Position cursor	Position cursor
0x04	^D	Sets background colour	Sets background colour	Sets background colour
0x05	^E	Erase to end of line	Erase to end of line	Erase to end of line
0x06	^F	Sets foreground colour	Sets colours /attributes	Sets foreground colour
0x07	^G	Sounds bell	Sounds bell	
0x08	^H	Backspace	Backspace	Backspace
0x09	^I	Tab	Tab	Tab
0x0A	^J	Line feed	Line feed	Line feed
0x0B	^K	Cursor up	Cursor up	Cursor up
0x0C	^L	Clear screen	Clear screen	Clear screen
0x0D	^M	Carriage return	Carriage return	Carriage return
0x0E	^N	CTLSPR	Blink on	Blink on
0x0F	^O	GENPAT	Blink off	Blink off
0x10	^P	COLOUR	Black foreground	Black foreground
0x11	^Q	ADJSPR	Red foreground	Red foreground
0x12	^R	SPRITE	Green foreground	Green foreground
0x13	^S	MOVSPR	Yellow foreground	Yellow foreground
0x14	^T	VIEW	Blue foreground	Blue foreground
0x15	^U	Insert key	Magenta foreground	Magenta foreground
0x16	^V	Delete key	Cyan foreground	Cyan foreground
0x17	^W	Tab back	White foreground	White foreground
0x18	^X	White text on black	Initialise configuration	Initialise configuration
0x19	^Y	Cursor right	Cursor right	Cursor right
0x1A	^Z	Home cursor	Home cursor	Home cursor
0x1B	^[	Escape	Escape	Escape
0x1C	^\	Scroll mode	Scroll mode	Scroll mode
0x1D	^]	Page mode	Page mode	Page mode
0x1E	^^	Show cursor	Show cursor	Show cursor
0x1F	^_	Hide cursor	Hide cursor	Hide cursor

Escape Codes

Letter	MTX Basic	Memotech 80 Col	CFX-II VGA
A	ATTR	Select alternate font	Select alternate alpha font
B	Select language	Set bit of both attributes	Set bit of both attributes
C	GR$	Scroll mode	Scroll mode
D	Invalid	Page mode	Page mode
E	Invalid	Show cursor	Show cursor
F	Invalid	Hide cursor	Hide cursor
G	Invalid	Select graphics font	Select lower graphics font
H	Invalid		Delete character under cursor
I	Insert blank line	Insert blank line	Insert blank line
J	Delete the current line	Delete the current line	Delete the current line
K	Duplicates a line		Duplicates a line
L	Read character at cursor		
M	Invalid		Redefine char.
N	Invalid	Set bit of non-printing attrib.	Set bit of non-printing attrib.
O	Invalid		Select virtual screen
P	Toggle Page / Scroll	Set bit of printing attributes	Set bit of printing attributes
Q	Invalid		Input 8-bit characters
R	Set print colour, clear attributes		Input raw buffer data
S	Invalid	Select standard font	Select standard alpha font
T	Invalid	Set printing attributes	Set printing attributes
U	Reset screen	Set non-printing attributes	Set non-printing attributes
V	BASIC setup	Set both attributes	Set both attributes
W	PANEL Setup	Set write mask	Set write mask
X	Simulate control code	Simulate control code	Simulate control code
Y	CRVS		Define virtual screen
Z	VS		Reboot
[			
\			
]			
^			Copy characters
_			Space characters

CP/M Display

In CP/M mode character sequences for display are sent to the Propeller video generator. Processing of control and escape sequences is performed by the Propeller firmware on the CFX-II card rather than by a CP/M driver such as that for the original Memotech 80 column card.

Mode Selection

The CFX-II starts in 80 column, 16 colour mode compatible with the original Memotech 80 column driver. The following character sequences (escape codes) are used to switch between modes:

Character Sequence	**Mode Selected**
0x1B, 0x9B	VDP emulation
0x1B, 0x9C	Compatible: 24 row x 80 column text, 16 colours, 160 x 96 graphics
0x1B, 0x9D	Monochrome: 24 row x 80 column text, monochrome green, 160 x 96 graphics
0x1B, 0x9E	Enhanced: 24 row x 80 column text, 64 colours, 160 x 96 graphics
0x1B, 0x9F	Graphics: 24 row x 40 column text, 64 colours, 320 x 240 graphics

These escape sequences do nothing on the original MTX, and have been chosen as unlikely to be generated accidentally. Note that the second character has to be exactly as given, none of the other characters with the same 5 lsb work (unlike most escape codes on the MTX).

Entering VDP emulation mode takes a few seconds and video generation is interrupted during the transition. Once the emulation has started the emulated VDP is still not configured with the MTX font, and shows a blank (red) screen. It then needs to be configured by writes to ports 0x01 and 0x02, as per the VDP. Once in VDP mode, this mode can only be exited by resetting the Propeller by a command to port 0x61 (see later), again causing an interruption to video generation.

Printable Characters (0x20 – 0xFF)

Typically a printable character will be displayed upon the screen, using the current printing attributes, and the cursor advanced one space. This may be affected by the write mask (Esc “W”). If this is set, the character may be updated, without changing the existing on-screen attributes, or the attributes may be updated without changing the displayed character.

If the cursor goes beyond the bottom of the screen, either the display will scroll (scroll mode) or the cursor will return to the top of the screen (page mode). There is nothing in the driver to pause and wait for a key press at the end of a page. This functionality must be at a higher level.

The CFX-II video has 256 character glyphs, illustrated in the figure below and an additional 256 plotting glyphs (the same as the Memotech 80 column card). Assuming that a character is output, the glyph displayed depends upon the font selected, and whether or not the graphics mode bit is set in the attribute byte.

If the graphics mode bit is clear, then the following table gives the glyph displayed:

Character Code	Standard Font	Alternate Font	Special Graphics Font
0x20 - 0x3F	Standard numerals (glyphs 0x20-0x3F)	Alternate numerals (glyphs 0xA0-0xBF)	Standard numerals (glyphs 0x20-0x3F)
0x40 - 0x5F	Standard upper case (glyphs 0x40-0x5F)	Alternate upper case (glyphs 0xC0-0xDF)	Special graphics (lower) (glyphs (0x00-0x1F)
0x60 - 0x7F	Standard lower case (glyphs 0x60-0x7F)	Alternate lower case (glyphs 0xE0-0xFF)	Special graphics (upper) (glyphs 0x80-0x9F)
0x80 - 0x9F	Special graphics (upper) (glyphs 0x80-0x9F)	Special graphics (upper) (glyphs 0x80-0x9F)	Special graphics (lower) (glyphs 0x00-0x1F)
0xA0 - 0xBF	Alternate numerals (glyphs 0xA0-0xBF)	Alternate numerals (glyphs 0xA0-0xBF)	Standard numerals (glyphs 0x20-0x3F)
0xC0 - 0xDF	Alternate upper case (glyphs 0xC0-0xDF)	Alternate upper case (glyphs 0xC0-0xDF)	Special graphics (lower) (glyphs 0x00-0x1F)
0xE0 - 0xFF	Alternate lower case (glyphs 0xE0-0xFF)	Alternate lower case (glyphs 0xE0-0xFF)	Special graphics (upper) (glyphs 0x80-0x9F)

It should be noted that character code 0x7F is printable. It does not delete the character under the cursor.

If the graphics mode attribute is set, then the corresponding plotting glyph is displayed. Each of the plotting glyphs consists of a 4x2 array of plot points as below, with the point filled if the corresponding bit is set, or clear if the bit is not set.

Bit 0 (lsb)	Bit 1
Bit 2	Bit 3
Bit 4	Bit 5
Bit 6	Bit 7 (msb)

Control Codes (0x00 – 0x1F)

Control codes may be followed by one or more data bytes update the display in various ways, as detailed in the following table. Any screen updates are affected by the settings of the write mask, as for the printable characters. Some of the details depend upon the currently selected mode (C=Compatible, M=Monochrome, E=Enhanced, G=Graphics).

Mode	Control Code	Data Bytes	Action
CMEG	^@ (0x00)		Does nothing
CME	^A (0x01)	m, n	Plot a point at x = m – 32, y = n – 32 If x < 0, x > 159, y < 0 or y > 95 do nothing.
G	^A (0x01)	xl, xh, y	Plots a point at x = 256 * xh + xl, y If x < 0, x > 319, y < 0 or y > 239 do nothing.
CME	^B (0x02)	m1, n1, m2, n2	Draws a line from x1 = m1 -32, y1 = n1 – 32 to x2 = m2 – 32, y2 = n2 - 32
G	^B (0x02)	x1l, x1h, y1, x2l, x2h, y2	Draws a line from x1 = 256 * x1h + x1l, y1 to x2 = 256 * x2h + x2l, y2
CMEG	^C (0x03)	m, n	Sets cursor position: row = m – 32, column = n - 32
C	^D (0x04)	m	Set background colour (printing and non-printing): Bit 0 = Red, Bit 1 = Green, Bit 2 = Blue
M	^D (0x04)	m	Set the following attributes (printing and non-printing): Bit 1 = Inverse video Bit 2 = Shaded background
EG	^D (0x04)	m	Set background colour (printing and non-printing): Bits 0 & 1 = Red, Bits 2 & 3 = Green, Bits 4 & 5 = Blue
CMEG	^E (0x05)		Erase to end of line. Fill from cursor position to end of line with space character and non-printing attribute.
C	^F (0x06)	m	Set colours and attributes (printing and non-printing): Foreground: Bit 0 = Red, Bit 1 = Green, Bit 2 = Blue Background: Bit 3 = Red, Bit 4 = Green, Bit 5 = Blue Attributes: Bit 6 = Blink, Bit 7 = Plotting

Mode	Control Code	Data Bytes	Action
M	^F (0x06)	m	Set attributes (printing and non-printing): Bit 0 = Underline, Bit 2 = Bright, Bit 4 = Inverse video, Bit 5 = Shaded background, Bit 6 = Blink, Bit 7 = Plotting
EG	^F (0x06)	m	Set foreground colour (printing and non-printing): Bits 0 & 1 = Red, Bits 2 & 3 = Green, Bits 4 & 5 = Blue
CMEG	^G (0x07)		Does nothing
CMEG	^H (0x08)		Moves the cursor back one space. If at the beginning of a line, moves back to the end of the previous line. If at top of screen, does nothing. Does not erase the character.
CMEG	^I (0x09)		Tab. Moves the cursor forward to the next multiple of 8 columns. If in the last 7 columns of a line moves to the start of the next line. If in last 7 characters of screen, either scrolls a line, or moves to top of screen, depending upon scroll / page mode.
CMEG	^J (0x0A)		Cursor down. Moves the cursor down one line. If on last line of screen, either scrolls a line, or moves to top of screen, depending upon scroll / page mode.
CMEG	^K (0x0B)		Move cursor up one line. If already at top of screen, do nothing.
CMEG	^L (0x0C)		Clear screen. Fill entire video RAM with space character and non-printing attribute (depending upon write mask). Sets top of screen to top of video RAM.
CMEG	^M (0x0D)		Carriage return. Sets cursor position to beginning of line.
CMEG	^N (0x0E)		Blink on. Sets the blink bit (bit 6) in the printing attributes byte only.
CMEG	^O (0x0F)		Blink off. Clears the blink bit (bit 6) in the printing attributes byte only.
CEG	^P (0x10)		Set print foreground colour to black.
M	^P (0x10)		Print attributes: Underline and bright off.
CEG	^Q (0x11)		Set print foreground colour to red.
M	^Q (0x11)		Print attributes: Underline on, bright off.
CEG	^R (0x12)		Set print foreground colour to green.
M	^R (0x12)		Print attributes: Underline and bright off
CEG	^S (0x13)		Set print foreground colour to yellow.
M	^S (0x13)		Print attributes: Underline on, bright off.
CEG	^T (0x14)		Set print foreground colour to blue.
M	^T (0x14)		Print attributes: Bright on, underline off.
CEG	^U (0x15)		Set print foreground colour to magenta.
M	^U (0x15)		Print attributes: Bright and underline on.

Mode	Control Code	Data Bytes	Action
CEG	^V (0x16)		Set print foreground colour to cyan.
M	^V (0x16)		Print attributes: Bright on, underline off.
CEG	^W (0x17)		Set print foreground colour to white.
M	^W (0x17)		Print attributes: Bright and underline on.
CMEG	^X (0x18)		Initialise display: Turns on scroll mode. Sets both printing and non-printing colours to green foreground on black background. Turns on the cursor. Enables both character and attribute writes. Performs a carriage return and line feed. Selects the standard font.
CMEG	^Y (0x19)		Cursor forward. If in the last column of a line moves to the start of the next line. If in last character of screen, either scrolls a line, or moves to top of screen, depending upon scroll / page mode.
CMEG	^Z (0x1A)		Home cursor. Moves cursor position to first character of top row.
CMEG	^[(0x1B)		Start of an escape sequence. See next section.
CMEG	^\ (0x1C)		Sets scroll mode. If cursor flows off the bottom of the screen, then all text is moved up one line, bottom line is cleared (using space character and non-printing attribute, depending upon write mask), and cursor is positioned on the new bottom line.
CMEG	^] (0x1D)		Sets page mode. If cursor flows off bottom of screen it re-appears at the top of the screen. None of the screen is cleared. There is no code in the driver to wait for a key press.
CMEG	^^ (0x1E)		Turn on display of the cursor position.
CMEG	^_ (0x1F)		Turn off display of the cursor position.

Escape Sequences

Escape sequences are similar to control codes. They consist of the ESC character (0x1B), followed by the command character and then possibly one or more data bytes.

In the Memotech implementation, if the character following the ESC is in the range 0x00 – 0x1F then the sequence is terminated and does nothing. Otherwise the 5 lsb of the command character are used to select the command to be executed. Therefore the sequence (ESC, 0x01) does nothing while each of the sequences (ESC, “!”), (ESC, “a”), (ESC, 0x81), (ESC, 0xA1), (ESC, 0xC1) and (ESC, 0xE1) all do the same as (ESC, “A”). In the table below, the standard upper case command character is listed, but the descriptions are applicable to all the other equivalents.

Mode	Command Character	Data Bytes	Action
CMEG	@ (0x40)		Does nothing.
CMEG	A (0x41)		Selects alternate font (see section on printable characters)
C	B (0x42)	m	Clears or sets print and non-print colour and attribute bits according to m = “0” (0x30) to m = “8” (0x38): “0”: Clear. Black foreground and background. “1”: Set red foreground bit. “2”: Set green foreground bit. “3”: Set blue foreground bit. “4”: Set red background bit. “5”: Set green background bit. “6”: Set blue background bit. “7”: Set blink bit. “8”: Set plot glyphs bit.
M	B (0x42)	m	Clears or sets print and non-print attribute bits according to m = “0” (0x30) to m = “8” (0x38): “0”: Clear. Green on black text. “1”: Set underline bit. “2”: No effect. “3”: Set bright bit. “4”: No effect. “5”: Set inverse video bit. “6”: Set shaded background bit. “7”: Set blink bit. “8”: Set plot glyphs bit.
EG	B (0x42)	m	Clears or sets print and non-print attribute bits according to m = “0” (0x30) to m = “8” (0x38): “0”: Clear. All attributes off. “1”: Set underline bit. “2”: No effect. “3”: No effect. “4”: No effect. “5”: Set inverse video bit. “6”: Set XOR drawing bit. “7”: Set blink bit. “8”: Set plot glyphs bit.
CMEG	C (0x43)		Sets scroll mode.
CMEG	D (0x44)		Sets page mode.
CMEG	E (0x45)		Turn on display of the cursor position.
CMEG	F (0x46)		Turn off display of the cursor position.
CMEG	G (0x47)		Selects special graphics font (see section on printable characters).
CMEG	H (0x48)		Deletes character under cursor.

Mode	Command Character	Data Bytes	Action
CMEG	I (0x49)		Moves all the text on the line containing the cursor and below down one line. The bottom line is lost. The line containing the cursor is cleared (using space character and non-printing attribute). The cursor retains its previous position (on the now blank line).
CMEG	J (0x4A)		Moves all text in lines below the cursor up one line. The new bottom line is cleared (using space character and non-printing attribute). The cursor retains its previous position.
CMEG	K (0x4B)		Duplicate the line containing the cursor on the next line down, pushing all lines below it down one line. The bottom line is lost. The cursor retains its previous position.
CMEG	L (0x4C)		Does nothing.
CMEG	M (0x4D)	n, m, p0, p1, p2, p3...	Redefine a character glyph. Each glyph is defined by five blocks of four rows of eight pixels. Therefore each block requires four bytes to redefine. n = Character glyph to redefine. m = Contains three bit fields: Bit 0: 0 = Text glyph, 1 = Graphics glyph. Bits 1-3: First block to redefine (0-4). Bits 4-6: Number of consecutive blocks to redefine. Bit 7: Unused. p1… = Bytes defining the pixels to set in successive rows of the glyph.
CMEG	N (0x4E)	m	Sets colour and attributes as per “<Esc>B” except only affects non-print output.
CMEG	O (0x4F)	m	Selects virtual screen 0 – 7 given by bottom 3 bits of m. Note: These virtual screens are not the same as those from Basic.
CMEG	P (0x50)	m	Sets colour and attributes as per “<Esc>B” except only affects print output.
CMEG	Q (0x51)	n, d1...	Input a number of 8-bit glyph codes without them being interpreted as control and escape sequences. n = Number of following byte glyph codes. d1 … = Bytes to display.
CMEG	R (0x52)	n, d1...	Input a number of characters in the video generator internal format (documented in the following section). n = Number of characters to input in raw internal format. d1… - Bytes forming each character. Four bytes per character for CME modes, ten bytes per character for G mode.
CMEG	S (0x53)		Selects standard font (see section on printable characters)

Mode	Command Character	Data Bytes	Action
C	T (0x54)	m	Sets the printing colour and attributes according to the bits of m: Bit 0: Red foreground. Bit 1: Green foreground. Bit 2: Blue foreground. Bit 3: Red background. Bit 4: Green background. Bit 5: Blue background. Bit 6: Blink. Bit 7: Plotting glyphs.
M	T (0x54)	m	Sets the printing attributes according to the bits of m: Bit 0: Underline. Bit 1: No effect. Bit 2: Bright text. Bit 3: No effect. Bit 4: Inverse video. Bit 5: Shaded background. Bit 6: Blink. Bit 7: Plotting glyphs.
EG	T (0x54)	m	Sets the printing attributes according to the bits of m: Bit 0: Underline. Bit 1: No effect. Bit 2: No effect. Bit 3: No effect. Bit 4: Inverse video. Bit 5: XOR printing. Bit 6: Blink (Enhanced mode only). Bit 7: Plotting glyphs.
CMEG	U (0x55)	m	As per “<Esc>T” but sets the non-print colours and attributes.
CMEG	V (0x56)	m	As per “<Esc>T” but sets both the print and non-print colours and attributes.
CMEG	W (0x57)	m	Sets the write mask: If m = “0” (0x30) enable both character and attribute writes. If m = “1” (0x31) disable attribute writes. If m = “2” (0x32) disable character writes. Any other value of m has no effect.
CMEG	X (0x58)	m	Simulate the effect of the control code given by m & 0x1F.
CMEG	Y (0x59)	n, x, y, w, h	Define a virtual screen: n = Virtual screen number (0 to 7). x = Leftmost column position (0 to 79). y = Top row position (0 to 23). w = Width (0 to 80 – x). h = Height (0 to 24 – y). Note this is not the same as the Basic CRVS.
CMEG	Z (0x5A)		Restarts the CFX-II video generator.

Mode	Command Character	Data Bytes	Action
CMEG	[(0x5B)		Does nothing.
CMEG	\ (0x5C)		Does nothing.
CMEG	] (0x5D)		Does nothing.
CMEG	^ (0x5E)	n, x1, y1, x2, y2	Copies n characters from position x1, y1 to position x2,y2 without moving the cursor.
CMEG	_ (0x5F)	n, x, y	Blanks n characters at position x, y without moving the cursor.

Hardware Interface – CP/M Modes

Input and Output Ports 0x60

For all except VDP emulation mode the CFX-II video generator displays characters, control and escape sequences sent to Z80 port 0x60.

Processing some sequences may take a little time, so the input characters are stored in a circular buffer until they can be processed. Reading Z80 port 0x60 returns a byte indicating how full this buffer is. A value of zero indicates the buffer is empty, larger values indicate increasingly full, with 0xFF indicating completely full. Writing to port 0x60 when the buffer is full is likely to result in loss of data and display corruption.

The CP/M driver for the CFX-II display simply checks that the buffer is not full, waits if it is, and then writes the characters to display to port 0x60.

Input and Output Ports 0x61

Port 0x61 is used for examining the state of the CFX-II video generator. Firstly, a one or two byte command is written to port 0x61 to select the required data, and then the required data is read back from port 0x61. The following sections describe the various output modes.

Zero Readback

In this mode, reading from port 0x61 will always return a zero. This is used to detect that the Propeller has restarted following a reset. The Z80 instruction “IN A,(0x61)” will return a value of 0x61 if there is nothing responding on that port. Receiving a zero value indicates that the Propeller has restarted and is returning data.

This is the default mode when the Propeller is restarted. It can also be re-enabled by writing 0xFE to port 0x61.

Reading Text Characters

Reading back text characters for compatible, monochrome or enhanced mode is initiated by writing two bytes to port 0x61 with the bit patterns:

0ccccccc, 100rrrrr

where ccccccc (0 – 79) is the column number, and rrrrr (0 – 23) is the row number.

Four bytes are then read for each successive character, containing the data as stored internally by the Propeller chip.:

Byte 0: Background colour (in bit pattern bbggrrxx).

Byte 1: Foreground colour (in bit pattern bbggrrxx).

Byte 2: Character code.

Byte 3: Attributes (this bit order is not the same as the attribute bits used for escape sequences):

Bit 0: 0 = Text, 1 = Graphics (effectively 9th bit of character code).

Bit 1: Underscore

Bit 2: Inverse video

Bit 3: Blink

Bit 4: Cursor

Bit 5: XOR mode

Reading Graphics Mode Characters

In graphics mode, character codes are not stored, only the resulting bit patterns. To read these characters, write two bytes to port 0x61 with bit patterns:

00cccccc, 101rrrrr

Twelve bytes are then read from port 0x61 for each successive character, containing:

Byte 0: Background colour (in bit pattern bbggrrxx).

Byte 1: Foreground colour (in bit pattern bbggrrxx).

Bytes 2 – 11: Pixels for each row of the character (top to bottom).

Reading Internal Registers

To read back a circular buffer of 32 internal registers, write a byte to port 0x61 with bit pattern:

110rrrrr

where rrrrr (0 - 31) is the register to start reading at. The buffer contains:

Byte	Contents
0	Video mode: 0 = Compatible, 1 = Monochrome, 2 = Enhanced, 4 = Graphics
1	Version byte (Year - 2000)
2	Version byte (Month)
3	Version byte (Day)
4	Control or Escape sequence state
5	Screen width (40 or 80)
6	Character size (4 or 12)
7	Row in display buffer for top of visible screen (updated on scrolling)
8-11	Print style (Background colour, Foreground colour, Character space, Attributes)
12-15	Non-print style (Background colour, Foreground colour, Character space, Attributes)
16	Width of current virtual screen
17	Height of current virtual screen
18	Left position of current virtual screen
19	Top position of current virtual screen
20	Scroll method: 0 = Copy characters, 1 = Pointer update

Byte	Contents
21	Cursor X position
22	Cursor Y position
23	Cursor visible (0 = No, 1 = Yes)
24	0 = Scroll mode, 1 = Page mode
25	0 = Standard character set, 1 = Alternate character set, 2 = Special graphics characters
26	Write mask: 0 = Both, 1 = Characters, 2 = Attributes
27	Current virtual screen number
28-31	Reserved. Currently zero.

Reading Font Table

The bit pattern for each glyph in the font table can be read by writing two bytes to port 0x61 with bit patterns:

0ccccccc 111000dd

to start reading at glyph 128 * dd + ccccccc. 0 – 255 are the text glyphs, and 256 – 511 are the plotting glyphs.

Then read 20 bytes to obtain each row (top to bottom) of the glyph. Continuing to read will return the patterns for successive glyphs, but reading past the end of the final glyph (511) will not correctly return to glyph 0.

Reset

Writing a byte 0xFF to port 0x61 will reset the Propeller , restoring everything to the initial state. The Propeller will stop producing video and stop responding to the Z80 ports for a few seconds while it restarts.

Polling either port 0x60 or 0x61 can be used to tell when it has restarted.

Hardware Interface – VDP Emulation

Port 0x60

VDP emulation mode is entered by writing the escape sequence 0x1B, 0x9B to port 0x60. In response to this command the Propeller will stop responding for a few seconds while it restarts in VDP emulation mode. It is possible to tell when the restart has completed by polling port 0x61 (see later). There will be no further response to port 0x60 until the Propeller is reset.

Ports 0x01 and 0x02

In VDP emulation mode the Propeller chip on the CFX-II will respond to the writes to these two ports in the same way as the VDP chip internal to the MTX. As a result it will largely reproduce the VDP display. There are two possible causes for differences:

- The Propeller may have been reset after the VDP was configured, in which case the Propeller emulation may not have been correctly reconfigured with the MTX font and memory layout. The Basic NEW command is one way to refresh the configuration, but remember that this will erase any currently loaded program.
- The Propeller produces frames at 60Hz, whereas the VDP produces frames at 50Hz. Therefore, games which attempt to update the display during periods between VDP frames may be part way through updates when the Propeller produces a frame.

The Propeller will never respond to a read request on ports 0x01 or 0x02, so it will never conflict with the VDP.

Port 0x61

As for the other modes, port 0x61 may be used to examine the internal status.

Zero Readback

In this mode, reading from port 0x61 will always return a zero. This is used to detect that the Propeller has restarted following a reset. The Z80 instruction “IN A,(0x61)” will return a value of 0x61 if there is nothing responding on that port. Receiving a zero value indicates that the Propeller has restarted and is returning data.

This is the default mode when the Propeller is started in VDP emulation mode. It can also be re-enabled by writing 0xFE to port 0x61.

Read Display Memory

To read bytes from the CFX-II copy of video RAM, write two bytes to port 0x61 with bit patterns:

0xxxxxxx 10yyyyyy

This will start reading from address 256 * yyyyyy + 2 * xxxxxxx. Note that it is only possible to start reading from an even address.

Then reading from port 0x61 will return successive bytes of video RAM, starting at this location.

Reading Internal Registers

To read back a circular buffer of 16 internal registers, write a byte to port 0x61 with bit pattern:

1100rrrr

where rrrr (0 – 15) is the register to start reading at. The buffer contains:

Byte	Contents
0	Video mode: 0x80 = VDP emulation mode
1	Version byte (Year - 2000)
2	Version byte (Month)
3	Version byte (Day)
4-7	Reserved. Undefined
8	VDP control register 0
9	VDP control register 1
10	VDP control register 2
11	VDP control register 3
12	VDP control register 4
13	VDP control register 5
14	VDP control register 6
15	VDP control register 7

So writing a single byte 0xC0 to port 0x61 then reading a single byte back will identify which mode the CFX-II display is in:

0x00 = Compatibility mode.

0x01 = Monochrome mode.

0x02 = Enhanced mode.

0x03 = Graphics mode.

0x61 = Restarting (null response)

0x80 = VDP emulation mode.

Also this provides the ability to read back the VDP control registers, which are write only on the real VDP.

Reset

Writing a byte 0xFF to port 0x61 will reset the Propeller , restoring everything to the initial state. This is the only way of exiting VDP emulation mode.

The Propeller will stop producing video and stop responding to the Z80 ports for a few seconds while it restarts. Polling either port 0x60 or 0x61 can be used to tell when it has restarted.

Interfacing Displays to MTX Basic

The MTX manuals document the existence of a 24 byte table TYPTBL at 0xFFD5, for interfacing up to eight different screen types, but no details on how it is used. This section describes what has been learnt in interfacing the type 2 and 3 displays for CFX-II.

The contents of TYPTBL are:

Byte	Description
0	ROM selector for screen type 0
1-2	Address in above ROM of command table for screen type 0
3	ROM selector for screen type 1
4=5	Address in above ROM of command table for screen type 1
6	ROM selector for screen type 2
7-8	Address in above ROM of command table for screen type 02
9	ROM selector for screen type 3
10-11	Address in above ROM of command table for screen type 3
12	ROM selector for screen type 4
13-14	Address in above ROM of command table for screen type 4
15	ROM selector for screen type 5
16=17	Address in above ROM of command table for screen type 5
8	ROM selector for screen type 6
19-20	Address in above ROM of command table for screen type 6
21	ROM selector for screen type 7
22-23	Address in above ROM of command table for screen type 7

The command table for each screen type consists of a list of 19 addresses for routines in that ROM to control that screen type. The functions of the 19 routines are:

Entry	Description
0x00	Return screen width in A
0x01	Update cursor position. Called repeatedly from a timer interrupt in order to flash the cursor. Only called if cursor is visible. Need to return with bit 7 of (IX+08h) cleared to ensure that this is called once when cursor is hidden.
0x02	Turn on cursor
0x03	Copy BC characters from HL to DE. Positions are specified in bytes from top left hand corner of display.
0x04	Display BC space characters at position HL. Uses plot colours. Should not move cursor.
0x05	Display the character in E at position HL
0x06	Initialise the virtual screen.

Entry	Description
0x07	Return colour of character at position HL, in A. Never actually used by MTX Basic.
0x08	Set print colours to C and clear print attributes.
0x09	Set both print and plot paper colour to the low 4 bits of next character. Set border colour for text screen.
0x0A	Set both print and plot ink colour to the low 4 bits of next character.
0x0B	Sets white text on black (called by <Ctrl+X>). Used by CRVS to initialise colours.
0x0C	Process device dependant control or escape sequence stored at BSSTR (0xFE3F)
0x0D	Return number of characters (including current) from row D column E to end of screen.
0x0E	Read character at position HL. Result in WKAREA. For graphics screen works by matching pattern.
0x0F	Test for control sequence valid and return length in H.
0x10	Cold start of display.
0x11	Configure BASIC virtual screens
0x12	Configure PANEL virtual screens

These routines are called from routine SCENT at 0x1876. SCENT is called with IX pointing to one of the virtual screen descriptions SCRN0 (0xFF5D) to SCRN7 (0xFFC6), and A containing the index of the routine to call. SCENT then:

- Sets up return via SCENTR (0x00CB) which will reset the ROM page to 0x10.
- Obtains the screen type from the top three bits of (IX+00).
- Obtains the ROM of the command table from TYPTBL.
- Selects that ROM, preserving RAM page.
- Finds the Ath entry in the command table for the display type.
- Calls that address in the ROM.

For CFX-II, although most of the routines for CFX Basic are in ROM 0x50, the command table and display routines for the type 2 and 3 display types are in ROM 0x40. There was not sufficient spare room in ROM 0x50 for this code, while there is spare space in ROM 0x40. The fact that the TYPTBL specifies which ROM to use made it easy to relocate this code.

Basic Memory Usage

High Memory

Support of the additional video modes makes use of the following locations in high memory:

Name	Address	Length	Description
vgamode	0xF600	1	Current mode for the CFX-II VGA output: 0 = Echoing VDP output 1 = 80 column text (Basic screen type 3) 2 = 40 column text (Basic screen type 2)
vgaedit	0xF601	1	Current USER VGA mode: 0 = Single monitor, 40 column edit (Screen type 0) 1 = Single monitor, 80 column edit (Screen type 3) 2 = Dual monitor, 40 column edit (on composite monitor) 3 = Dual monitor, 80 column edit (on VGA monitor)
Vgadbuf	0xF602	16	Buffer used for copying data from VDP to CFX-II when switching back to VDP echo mode.
vgapalette	0xF612	16	VGA colour values (bbggrr) for each of the 16 Basic colours.
vgatyp0	0xF622	38	Copy of TYPTX from ROM1, patched to support switching out of screen types 2 or 3, when type 0 selected.
vgatyp1	0xF648	38	Copy of TYPG2 from ROM1, patched to support switching out of screen types 2 or 3, when type 1 selected.
vgavdp1	0xF66E	7	Start of routine to switch to type 1 screen.
vgavdp0	0xF675	20	Start of routine to switch to type 0 screen.
jmprom1	0xF689	13	Routine to jump to address (HL) in ROM 1.

ROM 4

The following table summarises the relevant routines and tables in ROM 4.

Name	Address	Description
vgatt	0x2525	Data to be copied to TYPTBL. Points to patched command tables (in high memory) for types 0 and 1 screens, and the tables in ROM 4 for types 2 and 3 screens.
vgainit	0x2531	Updates TYPTBL and loads routines into high memory to support type 2 and 3 screens.
vgatyptx	0x2556	Command table for type 3 screen.
vgatypgr	0x257C	Command table for type 2 screen.
vganative	0x25A2	Reset Propeller VGA generator to native mode, so port 0x60 is active.
vdpdup	0x25B1	Copy DE bytes of VDP VRAM to CFX-II emulated VRAM, starting at HL.
vgatxreg	0x25ED	VDP register settings for type 0 screen.
vgagrreg	0x25F4	VDP register settings for type 1 screen.

Name	Address	Description
vgacold	0x25FB	Set Propeller to VDP emulation then do a cold restart of VDP display.
vgatest	0x2602	Test for 80-column editing. If not reset Propeller to VDP emulation mode.
vga2vdp	0x2609	Reset Propeller to VDP emulation and copy current VDP VRAM.
vgagraph	0x2659	Switch to type 2 screen.
vgatext	0x265D	Switch to type 3 screen.
vgamode1	0x265F	Switch to screen type selected by A.
vgainiclr	0x2684	Set print and plot colours according to VS definition at (IX).
vgaecho	0x26B1	Reset Propeller to VDP emulation.
vgaconf	0x26CF	Select screen type A. Return NZ if a change of screen type.
vgaout	0x26EF	Send a character to port 0x60, first checking that buffer is not too full.
vgastr	0x26FD	Output a string at (HL), terminated by a character with high bit set.
vgasend	0x270D	Output the string following the routine call, terminated by high bit set.
vgatxwth	0x2712	Return (in A) character width of type 3 screen.
vgagrwth	0x2715	Return (in A) character width of type 2 screen.
vgacsron	0x2718	Turn on display of cursor.
vgacsrflsh	0x271C	Update position of visible cursor.
vgacsrpos	0x272A	Update cursor position from VS definition at (IX).
vgarc	0x2744	Convert offset from top of screen (in HL) to row (H) and column (L).
vgaposn	0x276B	Set cursor position to offset HL from top of screen.
vgadisp	0x2784	Display character (E) at offset HL from top of screen.
vgaspace	0x278B	Display C spaces at offset HL from top of screen.
vgaflush	0x2797	Ensure Propeller has processed all characters (port 0x60 returns zero).
vgaread	0x279D	Prepare to start reading raw characters at offset HL from top of screen.
vgapeek	0x27AC	Read ASCII character at offset HL. Basic SPK$.
vgamfont	0x27C8	Match pattern in WKAREA against VGA font.
vgagrpk	0x27EC	Identify graphics character at offset HL. Graphics SPK$.
vgacopy	0x2820	Copy C characters from offset HL to offset DE on screen. (Used to scroll line).
vgacolour	0x2841	Convert Basic colour in A to 6-bit colour from vgapalette.
vgaprtclr	0x284F	Set print colours to C and clear print attributes.
vgagetch	0x2871	Get next character pointed to by CHPTR (in BSSTR).
vgappap	0x287C	Set print paper colour as specified by VS at (IX).
vganpap	0x2886	Set non-print paper colour as specified by VS at (IX).
vgapaper	0x2890	Set paper colours as specified by VS at (IX).
vgapink	0x28A9	Set print ink colour as specified by VS at (IX).
vganink	0x28B3	Set non-print ink colour as specified by VS at (IX).

Name	Address	Description
vgaink	0x28BD	Set inks colour as specified by VS at (IX).
vgasetclr	0x28DB	Set colour according to Basic <Ctrl+P> sequence at HL.
vgawhbk	0x291E	Set white text on black.
vgagrsiz	0x292E	Return remaining space on type 2 screen.
vgatxsiz	0x2933	Return remaining space on type 3 screen.
vgagetsiz	0x2957	Return length of control or escape sequence in WKAREA.
vgatxctl	0x2981	Test for valid screen type 3 control or escape sequence and return length.
vgagrctl	0x298D	Test for valid screen type 2 control or escape sequence and return length.
vgaclen	0x299B	Control sequence length and validity flags.
vgaelen	0x29BB	Escape sequence length and validity flags.
vgacmnd	0x29D5	Send control or escape sequence to Propeller.
vgaesc	0x29F8	Process escape sequence.
vgaattr	0x2A07	Process Basic ATTR (<Esc> A) escape sequence.
vgapsize	0x2A41	Get size in plot points of VS.
vgachkx	0x2A80	Check valid X coordinate (also add offset of 32 for type 3 screen).
vgachky	0x2A8C	Check valid Y coordinate (also add offset of 32 for type 3 screen).
vgaplot	0x2A92	Process a PLOT control sequence.
vgapat	0x2AFD	Process a GENPAT command.
vgatpix	0x2B2D	Get plot points from text screen.
vgagrpk	0x2B49	Process GR$ command.
vgahipk	0x2BB0	Process GR$ command for type 2 display.
vgavspan	0x2C08	Configure virtual screens for PANEL
vgavsbas	0x2C17	Configure virtual screens for Basic editing.
vgasetscr	0x2C4D	Load a virtual screen configuration.
vgascr0	0x2C72	VS configurations for Basic.
vgapan0	0x2C8A	VS configurations for PANEL.

ROM 5

The following table summarises the relevant routines and tables in ROM 5.

Name	Address	Description
echovdp	0x3B04	Configure the CFX-II display to echo the VDP.
hrout	0x3B1B	Send a character to the CFX-II display after checking buffer not full.
hrdev	0x3B29	Check that current VS is a type 2 display.
hrsize	0x3B36	Get size in pixels of VS in type 2 display.
hrpoint	0x3B60	Check valid X and Y coordinates and add to WKAREA.

Name	Address	Description
hrplot	0x3B73	Process USER PLOT command for type 2 screen.
hrline	0x3B95	Process USER LINE command for type 2 screen.
hrcolour	0x3BAD	Process USER COLOUR command.
initvga	0x3BCF	Initialise support for type 2 and 3 screens.
svga	0x3BD4	Process USER VGA command.
callvga	0x3BFA	Call routine at HL in ROM 4, and return to current ROM.