MMBasic 5.08

Please find attached the first beta for the USB enabled version of the PicoMiteVGA

PicoMiteVGAUSB.zip

This supports USB keyboards, mouse, PS4 controller, PS3 controller and a generic (SNES) type gamepad

The Pico must be powered with 5V on the VBUS pin and you can connect a USB-TTL converter as a console to pins 11 and 12 of the Pico (GP8=TX and GP9=RX)

A maximum of 4 devices may be connected at any one time. These are reference by an channel index number (1-4). Use MM.INFO(USB n) to return the device code for any device connected to channel n.

Returns are:
0=not in use, 1=keyboard, 2=mouse, 128=ps4, 129=ps3, 130=SNES/Generic

You should be able to use more than one of any of the controllers, mice, or keyboard tested with two keyboards)

Please see the **demo video** showing simultaneous support for a keyboard, mouse, PS4 controller and PS3 controller

The code to run this demo is attached below:
A USB keyboard acts exactly like any other console keyboard
The mouse and controllers are supported through the DEVICE command and function as in the example

The mouse is constrained to return x and y values between 0 and MM.HRes-1 and 0 and MM.VRES-1

The scroll wheel on the mouse will act as a button but the wheel itself is not currently enabled (I Don't know the USB report to send to enable Intellimouse mode)

All buttons on the various controllers can be selectively enabled to provide interrupts as per the Wii Classic in the documentation

Note: The TinyUSB host code is far from robust and devices and hubs can get into undefined states, particularly with powered hubs. To ensure things work, start by powering everything off, disconnect everything from the hub. Then power the hub (or connect an unpowered hub the the Pico. Power the Pico and then one at a time add devices into the hub.
Both and power-off and reset with an unpowered hub should result in the devices reconnecting. Powering off the Pico and not a powered hub may give problems

---

OPTION SERIAL CONSOLE TXpin, RXpin
can change the pins used for the console

MMBasic 5.09

Version 5.09.00b0

https://geoffg.net/Downloads/picomite/PicoMite_Beta.zip

Contains:
PicoMite
PicoMiteVGA
WebMite
PicoMiteUSB
PicoMiteVGAUSB

See the **PicoUSB thread for more details of the USB variants**

All code and build files on https://github.com/UKTailwind/ including USB variants

Changes to BLIT and SPRITE

Background
The PicoMite is normally used with SPI displays and these are too slow to use the full sprite fuctionality that was originally developed for the CMM2. Because of this I left sprite functionality out of the PicoMite and only included it in the PicMiteVGA.
However, since that early decision, I have included the framebuffer and layerbuffer functionality into the PicoMite. Like the PicoMiteVGA, these are based in RAM and pixels are stored as 4-bits each - 1 nibble per pixel. This is efficient in terms of memory and allows very fast read-modify-write transactions which are the basis of the sprite functionality.
According, I have decided to incorporate true sprites into the PicoMite and WebMite variants (and the PicoMiteUSB) to work on the framebuffer or layerbuffer.
At the same time, it is clear that the use of the commands SPRITE and BLIT which were interchangeable has resulted in confusion so I am now including both in all variants with the relevant functionality limited to the appropriate command.
In order to free-up the extra command slot the, little used, REFRESH and FLUSH commands have been combined (FLUSH without a parameter = REFRESH).
In addition to free up a function slot for the SPRITE functions MM.HPOS and MM.VPOS are removed and replaced by MM.INFO(HPOS) and MM.INFO(VPOS).
Existing code does not need to change as the firmware will make the changes automatically.

SPRITE command and function changes
The sprite command and function function as per the VGA manual with the exception that its use for a BLIT operations (SPRITE x, y, newx, newy, width, height; SPRITE memory) is removed.
SPRITES are ALWAYS 4-bits per pixel and therefore for the non-VGA variants can only be used on a FRAMEBUFFER or LAYERBUFFER. Because of this storage of sprites is very memory efficient
New command
SPRITE SET TRANSPARENT RGB121colour 'sets the colour(1-15) that will be used as transparent when displaying sprites - defaults to zero
In addition the maximum number of available sprites is increased to 64 as per the CMM2
Full List:
SHOW SAFE, SHOW, HIDE ALL, RESTORE, HIDE SAFE, HIDE, SWAP, READ, COPY, LOADARRAY, LOAD, INTERRUPT,
NOINTERRUPT, CLOSE ALL, CLOSE, NEXT, WRITE, LOADBMP, MOVE, SCROLL, SET TRANSPARENT

BLIT command
The BLIT command can work on both framebuffers and physical displays. Blit buffers, as used in READ/LOAD and WRITE are RGB888 and therefore take 3 bytes per pixel

The maximum number of available BLIT bufferss is increased to 64

Full List:
MEMORY, COMPRESSED, FRAMEBUFFER, LOADBMP/LOAD, MERGE, READ, WRITE, CLOSE, normal BLIT

This change may break existing programs but the modification needed should be trivial - i.e. swapping BLIT for SPRITE commands or visa-versa

I attach a simple demo program showing a standard PicoMite (or PicoMiteVGA) using all 64 sprites with full collision detection

---

Quote I'm not sure what all the difference between blit/image buffers and sprites are, though I know the latter have some collision detection support.

The concept of BLIT is simply that you move one area of memory to another replacing what was there and of course those areas of memory are going to be related to some sort of image data or a physical screen

SPRITES are different. You can create a sprite in various ways but essentially you are just storing an image in a buffer. The difference comes when you SHOW the sprite. In this case, first time in, the firmware stores the area of memory (or display real-estate) that will be replaced by the sprite and then draws the sprite in its place. Subsequent SHOW commands replace the sprite with the stored background, store the background for the new location and finally draw the sprite.

In this way you can move the sprite over the background without any extra code.

Collision detection then sits on top of this and looks for the rectangular boundaries of sprites touching to create an interrupt or a sprite touching the edge of the frame.

Finally sprites are ordered so the drawing order is held in a lifo. suppose you have sprite 1 overlapped by sprite 2 and then by sprite 3. If you simply moved sprite 1 then its background would overwrite bits of 2 and 3 - not what we want. SPRITE SHOW SAFE unwinds the LIFO by removing each sprite in reverse order, moves sprite 1 and then restores first 2 and then 3 on top of it.

Finally and just to complicate things more there is the concept of layers (this is the 4th parameter in SPRITE SHOW).

As per the Appendix in the manual

- Sprites collide with sprites on the same layer, layer 0, or the screen edge.
- Layer 0 is a special case and sprites on all other layers will collide with it.
- The SCROLL commands leave sprites on all layers except layer 0 unmoved.
- Layer 0 sprites scroll with the background and this can cause collisions.

Hopefully you can see from the above why it is unrealistic to implement true sprites on an SPI display the replace,save, draw sequence results in artifacts and things like scroll where all sprites have to be removed before the background scrolls and then replace after is just a non-starter

---

Version 5.09.00b1

https://geoffg.net/Downloads/picomite/PicoMite_Beta.zip

Contains:
PicoMite
PicoMiteVGA
WebMite
PicoMiteUSB
PicoMiteVGAUSB

No changes to the VGA versions so all of the below applies to the three non-VGA firmware variants

New optional parameter for 8-bit LCD displays (SSD1963_XX_8)
OPTION LCDPANEL DISPLAYTYPE, orientation [,backlightpin] [,DCpin] [,NORESET]

If NORESET is specified the firmware only allocates 3 consecutive control pins DC, WR, and RD. In this case the RESET pin should be tied high or connected to the RUN pin. Use IFF this doesn't create any initialisation issues.
As before by default DC, WR and, RD will be pins GP13, GP14, and GP15 by default but this can be changed to any other set of consecutive pins using the DCpin parameter

Full support for 16-bit parallel displays - all types of SSD1963, ILI9341, and IPS 4" displays using the OTM8009A or NT35510 controller (NT35510 untested - I don't have one). These give exceptional perfomance, even compared to the 8-bit parallel versions (e.g. CLS on a ILI9341 @ 378MHz takes just 4.5mSec)

OPTION LCDPANEL DISPLAYTYPE, orientation [,backlightpin] [,DCpin][,NORESET]
Pins GP0 to GP15 are allocated to the D0 to D15 data pins, By default pins GP16-GP19 are allocated to DC, WR, RD, and RESET respectively - use the optional DCPIN parameter to change this. As above use the optional parameter NORESET to keep the fourth control pin for other use and tie the RESET pin high.

New command for initialising an SD card (not VGA versions)

OPTION SDCARD COMBINED CS

If this is specified the touch chip select pin is also used for the SDcard. In this case external circuitry is needed to implement the SD chip select.
The firmware uses the touch pin as follows:
TOUCH_CS low: touch pin driven low
SD_CS low: touch pin driven high
TOUCH_CS and SD_CS high: touch pin set as input (high impedance)

A suitable tested circuit to implement this is as below:
Thanks to Turbo46 and Volhout for design and simulation of the circuit

---

For everyone with the USB variants

MAKE SURE THE TERMINAL PROGRAM IS SET TO 115200. It tells the CH340 what baudrate to use on the TTL side over the USB descriptor. This is not the same as a CDC port where the baudrate setting doesn't matter. The Pico COM port is configured for 115200. This can be changed with OPTION BAUDRATE but get it

working at 115200 first

Version 5.09.00b2

https://geoffg.net/Downloads/picomite/PicoMite_Beta.zip

Fixes bug in FRAMEBUFFER COPY for SPI displays introduced in these betas
Major performance improvements for 16-bit parallel displays

Version 5.09.00b3

https://geoffg.net/Downloads/picomite/PicoMite_Beta.zip

All versions:

Fixes bug in LINE command when using integer arrays or integer array elements
SPTITE MEMORY is automatically converted to BLIT MEMORY for backwards compatibility (PETSCII tested)

PicoMite, PicoMiteUSB and WebMite only:

Completes support for 16-bit displays

pinout for the 16-bit displays is always:
GP0-GP15 = DB0 to DB15
GP16 = RS/DC
GP17 = WR
GP18 = RD
GP19 = RESET

You can use the optional parameter NORESET to free up GP19 in which case the RESET pin should be tied high or connected to the RUN pin. The optional parameter DCPin is ignored for 16-bit displays. Both DCPin and NORESET are available for 8-bit parallel displays.

```
OPTION LCDPANEL displaytype, orientation [,BacklightPin][,DCPin][,NORESET]
```

Valid 16-bit displays are:
ILI9341_16,
IPS_4_16,
SSD1963_4_16, SSD1963_5_16,SSD1963_5A_16, SSD1963_7_16,
SSD1963_7A_16, SSD1963_8_16, SSD1963_5ER_16, SSD1963_7ER_16,

New command
OPTION LCD320 ON/OFF

This enables or disables 16-bit displays in 320x240 mode allowing things like games on these larger displays

In the case of 800x480 displays the 320x240 image is scaled by 2 and occupies the screen area 80,0 to 719,479
In the case of 480x272 displays the 320x240 image is windowed and occupies the screeen area 80,16 to 399,255

New sub-function
MM.INFO(LCD320)

Returns true if the display is capable of 320x240 operation using the OPTION LCD320 command.

So: In a game just include the statement

```
if MM.INFO(LCD320) THEN OPTION LCD320 ON
```

In any games that run in 320x240 resolution to allow them to use these bigger displays

---

Version 5.09.00b4

https://geoffg.net/Downloads/picomite/PicoMite_Beta.zip

Fixes bug in MATH(CHI
Allows MATH C_MUL and MATH C_MULT

Fixes compile error with PICOMITEUSB version NB: Installing this version (PICOMITEUSB) will delete the A: drive and all options

---

Version 5.09.00b5
https://geoffg.net/Downloads/picomite/PicoMite_Beta.zip

Puts a block on OPTION LCDPANEL CONSOLE n where font n would result in less than 30 characters display width
Enables defaulting caps_lock and num_lock in OPTION KEYBOARD lang [,capslock] [,numlock]

In both case values of 0 or 1 are allowed specifying if the lock is on-1 or off-0.
Default if not specified is caps-lock off and num-lock on

In betas previous to 4 Ctrl-C or program end would disable framebuffers and restore FRAMEBUFFER WRITE to the main display

b4 left the write status as-is. Seems like this confused some users who didn't see the cursor after ctrl-C if the write was to a framebuffer so b5 reverts to previous functionality

---

Version 5.09.00b6

https://geoffg.net/Downloads/picomite/PicoMite_Beta.zip

Fixes bug in MATH M_MULT

Blocks ILI9481IPS display from being used as the console

Major internal re-engineering of command tokens

Programs in memory will be corrupt and must be re-loaded from disk or re-downloaded

FOR KEVIN

There is a bug in BLIT of a framebuffer when a 480x320 SPI display is the main display. I don't believe that there is a bug for an ILI9341 display or VGA. Please confirm.

Firmware V5.09.00 release candidate RC1 is now available for download from

https://geoffg.net/Downloads/picomite/PicoMite_Beta.zip

This release includes some major internal changes including the move to 16-bit command tokens. The source has been updated on GitHub.

There are 5 firmware variants:

PicoMite - base version
PicoMiteVGA - base VGA version
WebMite - base version for Pico-W
PicoMiteUSB - Version with serial console support on GP8/GP9 and HID device support
PicoMiteVGAUSB VGA version with serial console support on GP8/GP9 and HID device support

HID device support includes up to 4 of USB keyboard, USB mouse, PS3 controller, PS4 controller and SNES gamepad directly connected for one device and via a 4-port hub for more than one device.

In addition this release includes LCD console support for any connected TFT display. Various optimisations have been included to overcome the speed constraints of SPI displays.

The release also includes support for 16-bit parallel connected displays for the ultimate in display speed.

A number of bugs have been corrected

In addition the release includes support for fast configuration of a number of boards using the OPTION RESET board command. More will be added during the release candidate process. No longer will you have to keep visiting the various sources to work out how to configure a board. Where applicable the configuration will include touch calibration but as this may vary between displays you may need to re-calibrate once the firmware is configured.

Please read the development threads **here** and **here** for mere details on the contents of the release

This release will delete all current options and in some cases the A: drive

OPTION RESET CMM1.5

VGA version now allow up to 378MHz CPU speed again. Use at your own risk.
Standard PicoMite now allows up to 420MHz. Likewise.

For all versions the CPUspeed is now checked as to whether the H/W can be configured to deliver it and an error is given before it is set if it can't.

Also, Even with a valid CPU speed, the firmware now sets the H/W watchdog before the CPU is configured. If this fails then the watchdog is triggered and the CPU speed is set back to default with an appropriate error message.

Fixes error in OPTION RESET RP2040-GEEK
Fixes spurious output when plugging a USB keyboard while running a program

New boards

PICOGAME 4-PWM
PICOGAME 4
SWEETIEPI
VGA Basic
USB Edition V1.0

Use OPTION RESET LIST to see boards that are available on any specific version of the firmware

OPTION LCDPANEL anyspidisplay orientation, dcpin, resetpin, cspin [,backlightpin] [,INVERT]

if the new optional parameter "invert" is used the colours are inverted - supports the IPS ILI9488 display

OPTION LCD320 ON/OFF now supports 480x320 SPI displays. This allows programs that run on a 320x240 display to be run and tested on the bigger display by restricting the framebuffer size to 320x240

Firmware V5.09.00 release candidate RC3 is now available for download from

https://geoffg.net/Downloads/picomite/PicoMite_Beta.zip

Improves handling of mount/unmount of USB devices
Removes spurious extra MM.INFO$(in list functions

NB: CONFIGURE name == OPTION RESET name

NB: for USB devices
The one thing that isn't handled is reconnecting USB devices when the Pico is reset without a power cycle or a powered hub is used. There seems to be something missing in TinyUSB to handle resetting hubs as part of the TinyUSB initialisation.

It is recommended to use an un-powered hub and in a perfect world pressing reset would remove power to the hub

---

Firmware V5.09.00 release candidate RC5 is now available for download from

https://geoffg.net/Downloads/picomite/PicoMite_Beta.zip

RC5 just updated - no version change

colour map functionality correct in all versions

new function

PIN(BOOTSEL)

returns the state of the boot select switch allowing you to use it as a user input in a program

orientation is available in BLIT WRITE and SPRITE SHOW, SPRITE SHOW SAFE, SPRITE WRITE

Remember the difference between BLIT and SPRITE commands is that BLIT works in the resolution of the display device whereas SPRITE is always RGB121

Prompted by a comment from Stanley, Ive added the map command and function to the PicoMiteVGA and PicoMiteVGAUSB based on the CMM2 functionality (see the CMM2 manual).

Quote Can you explain what the difference is between the Map(I) as function, and an array Map() in this application.?

The map(function returns a 24-bit RGB value that will pick up the slot that you have specified a colour for

so:

MAP(1)=rgb(brown)
map set

Will cause any "normally" blue pixels to turn brown. In order to then use a drawing command with the brown colour use the map(1) colour

This should be obvious in the example. This is how the CMM2 does it using a H/W colour lookup table (CLUT) whereas the PicoMi VGA driver in S/W.

NB: this functionality is only available at CPU speeds of 252 and 378 MHz

---

RP2350 introduced

Should run on any RP2350 with pins GP12 to GP19 broken out.
Wiring:
GP12 : D0+ GP13 D0-
GP14 : CK+ GP15 CK-
GP16 : D2+ GP17 D2-
GP18 : D1+ GP19 D1-

This should behave pretty much like the VGA firmware. CPU speed is fixed at 315MHz and your screen will see 640x480@75Hz
The maximum number of variables is increased to 768 and the maximum number of function/subroutines/labels to 512.
Maximum program size is now 180Kbytes and there is 42Kbytes used for simple variables and variable headers and another 180Kbytes for arrays and strings.
I expect with the M33 core that it should be possible to enable MP3 playback with just the standard audio connections. This is the next job.
There are many conditional compilation directives to support the HDMI/RP2350 version with exactly the same codebase as all other Picomite variants so there may well be bugs and/or functionality I have patched out and not yet converted - please report anything you find

At the moment graphics modes 1 and 2 are implemented and things like frame-buffers and layer-buffers should work like the VGA version EXCEPT that when creating them you don't use any of the user memory. Set OPTION COLOURCODE ON to have colour highlighting in mode 1 editing as before.

The map command is now more powerful and will work in mode 2.

MAP(n)=RGB888colour 'n=0 to 15

'this sets up an internal mapping for colour n in RGB555

MAP SET

'this actions any previous map(n) commands and the screen will then change colour. So you can now use any 16 of 32768 colours in mode 2

MAP RESET

'sets the mapping back to the default RGB332 mapping

MAP GREYSCALE/GRAYSCALE

'instantly changes all 16 colours to greyscale levels. Try this then do GUI TEST LCDPANEL to see the effect. You can then still use the MAP(n)/MAP SET to change some of the greys to another colour

MAP MAXIMITE

'instantly changes all colours so that colours 0 to 7 are the same as the colour Maximite

REMEMBER to access a mapped colour use the MAP function

MAP(1)=RGB(50,50,50):MAP SET
TEXT 0,0,"Hello world",,,,MAP(1) 'output text in the grey colour just defined

PSRAM can be enabled with

OPTION PSRAM PIN n ' n=0, 8, 19, or 47. If this is done on a RP2350 without PSRAM then the option will be automatically disabled.
To access PSRAM use the address &H11000000

e.g.
poke word &H11000000,&H12345678
? peek(word &H11000000)

More to come on this later

This is for Harm

PicoMiteVGARP2350.zip

New PIO stuff which I may or may not have understood: totally untested

In theory the following assembler statements are now supported
mov rxfifo[y], isr
mov rxfifo[<index>], isr
mov osr, rxfifo[y]
mov osr, rxfifo[<index>]

In addition two new parameters to the shiftctlr function

push threshold,pull
threshold,[autopush],[autopull],[IN_SHIFTDIR],[OUT_SHIFTDIR],[FJOIN_RX],[FJOIN_TX],[FJOIN_RX_GET,[FJOIN_RX_PUT

Then a new subcommand

WRITEFIFO pio, sm, fifo, value

and a new function

PIO(READFIFO pio, sm, fifo)

If I understand correctly, you need to enable either FJOIN_RX_GET, and/or FJOIN_RX_PUT in order for the new assembler statements to do anything and then, depending on them being enabled you can read and/or write directly to the FIFO cells

Quote When only SHIFTCTRL_FJOIN_RX_PUT is set (in SM0_SHIFTCTRL through SM3_SHIFTCTRL), the system can also read the RX FIFO registers with random access via RXF0_PUTGET0 through RXF0_PUTGET3 (where RXFx indicates which state machine's FIFO is being accessed). In this state, the FIFO register storage is repurposed as status registers, which the state machine can update at any time and the system can read at any time. For example, a quadrature decoder program could maintain the current step count in a status register at all times, rather than pushing to the RX FIFO and potentially blocking.

Quote When only SHIFTCTRL_FJOIN_RX_GET is set, the system can also write the RX FIFO registers with random access via RXF0_PUTGET0 through RXF0_PUTGET3 (where RXFx indicates which state machine's FIFO is being accessed). In this state, the RX FIFO register storage is repurposed as additional configuration registers, which the system can update at any time and the state machine can read at any time. For example, a UART TX program might use these registers to configure the number of data bits, or the presence of an additional stop bit.

Quote When both SHIFTCTRL_FJOIN_RX_PUT and SHIFTCTRL_FJOIN_RX_GET are set, the system can no longer access the RX FIFO storage registers, but the state machine can now put/get the registers in arbitrary order, allowing them to be used as additional scratch storage.

V6.0.0b1 for HDMI, PicoMite and VGA - no USB versions yet

PicoMiteRP2350V60.0b1.zip

This has the new PIO functions as above + full transparent support for PSRAM

USB editions all now tested and working with a USB keyboard and USB game controller. Remember to apply 5V on pin 40 (VBUS) and serial console on pins GP8 and GP9

PicoMiteUSBRP2350.zip

Non-USB editions

PicoMiteRP2350.zip

Should fix the PIO issue above

All HDMI and VGA versions now support "MODE 3" which is 640x480c16 colour.
This can be set as the default with
OPTION DEFAULT MODE 3

All versions now support PSRAM if enabled with
OPTION PSRAM PIN pinno 'pinno is typically GP47 for Pimoroni boards
When enabled PSRAM can be used like normal ram to store large arrays/strings etc.

V6.00.00b2

Minor bug fixes and I've re-coded pause although it hasn't changed in months.
Should fix artifacts in HDMI mode 1
Fixes bug in map() function for VGA in mode 3
Increases program size HDMI/VGA = 180Kb, PicoMite = 256Kb

PicoMiteRP2350.zip

PicoMiteUSBRP2350.zip

V6.00.00b3

PicoMiteRP2350.zip

Fixes the WS2812 bug

Lots of work on the video output. NB: For both VGA and HDMI versions 320x240*16 bits of memory are allocated for video memory. In all cases the screen will see a 640x480 image with a refresh rate of 75Hz (31.5MHz pixel rate)

Video Modes VGA and HDMI

Mode 1: 640x480x2bits with tiles.
Tiles width is fixed at 8 pixels. Tile height defaults to 12 pixels but can be from 8 to MM.HRES. Tiles colours are specified using the standard RGB888 notation. This is converted to RGB121 for VGA versions and RGB555 for HDMI versions. A framebuffer (F) and a layer buffer (L) can be created. These have no impact on the display and do not use user memory but both can be used for creating images and copying to the display screen (N)

Mode 2: 320x240x4 bits.
A framebuffer (F) can be created. This have no impact on the display and does not use user memory but can be used for creating images and copying to the display screen (N).
In addition a layer buffer can be created. This also does not use user memory. any pixels written to the layer buffer will automatically appear on the display sitting on top of whatever may be in the main display buffer. A colour can be specified (0-15: defaults to 0) which does not show allowing the main display buffer to show through.

Map functionality is available to override the default colours of the 16 available

In the case of VGA, the hardware is limited to the 16 colours defined by the resistor network.
In the case of HDMI each of the 16 colours can be mapped to any RGB555 colour. Use MAP GREYSCALE to have 16 shades of grey from black to white available.

```
mode 2
framebuffer layer
framebuffer write l
text 160,120,"hello",cm,,5,rgb(red)
framebuffer write n
gui test lcdpanel
```

Mode 3: 640x3200x4 bits.
A framebuffer (F) can be created. This have no impact on the display but uses a big chunk of user memory. It can be used for creating images and copying to the display screen (N).
In addition a layer buffer can be created. This also uses a big chunk of user memory. Any pixels written to the layer buffer will automatically appear on the display sitting on top of whatever may be in the main display buffer. A colour can be specified (0-15: defaults to 0) which does not show allowing the main display buffer to show through. The RP2350 memory can only hold one of a layer buffer and a framebuffer unless PSRAM is available. If PSRAM is available the layer buffer must be created first as it must be in the processors tightly coupled memory.

Map functionality is available to override the default colours of the 16 available

In the case of VGA, the hardware is limited to the 16 colours defined by the resistor network.
In the case of HDMI each of the 16 colours can be mapped to any RGB555 colour. Use MAP GREYSCALE to have 16 shades of grey from black to white available.

Additional HDMI Video Modes

Mode 4: 320x240x16 bits

This is full RGB555 allowing good quality colour images to be displayed
A framebuffer (F) and a layer buffer (L) can be created. These have no impact on the display and use big chunks of user memory but both can be used for creating images and copying to the display screen (N). Only one can be created unless PSRAM is available.

Mode 5: 320x240x8 bits
A framebuffer (F) can be created. This have no impact on the display but uses a big chunk of user memory. It can be used for creating images and copying to the display screen (N).
In addition a layer buffer can be created. This does not use user memory. Any pixels written to the layer buffer will automatically appear on the display sitting on top of whatever may be in the main display buffer. A colour can be specified (0-255: defaults to 0) which does not show allowing the main display buffer to show through.

Map functionality is available to override the default colours of the 256 available
Each of the 256 colours can be mapped to any RGB555 colour. Use MAP GREYSCALE to have 32 shades of each of the 7 standard colours conveniently available.
MAP(0) to MAP(31) - grey, MAP(32) to MAP(63) - blue etc. Have a play.

```
MODE 5
Map greyscale
j=(320-256)/2
For i=0 To 255
Line i+j,0,i+j,MM.VRes-1,1,Map(i)
Next
```