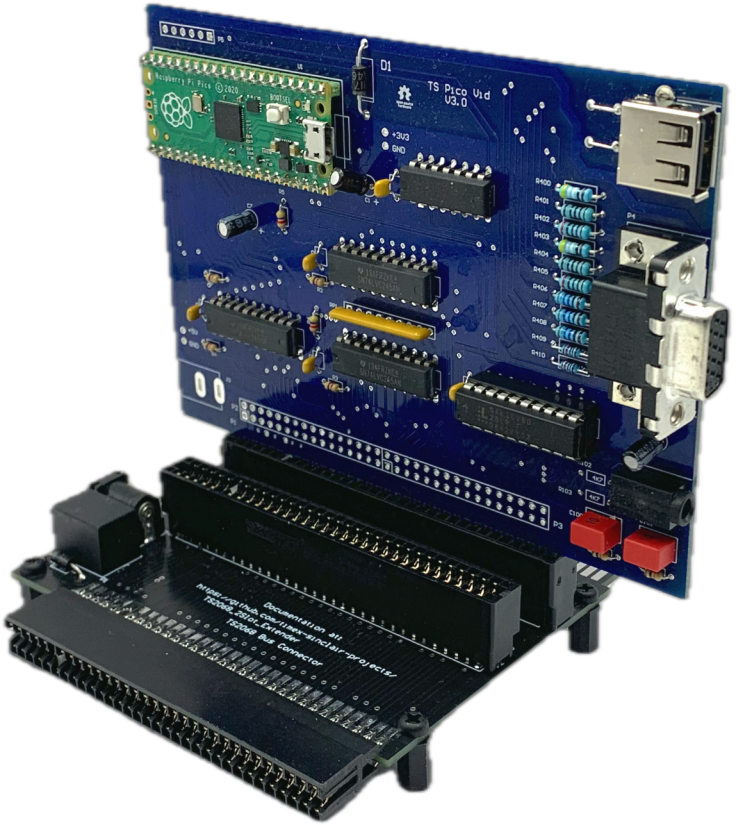
TIMEX sinclair

# 2068 PERSONAL COLOR COMPUTER VIDEO INTERFACE

# USER MANUAL



PicoVideo

The PicoVideo was designed and programmed by Jeff Burrell.

It is an open source hardware project. More information is available here:
https://github.com/timex-sinclair-projects/PicoVideo

Thanks to the TS-Pico team for their help and support.

# Introduction

Your PicoVideo circuit board is the result of many years of experiments to improve the video output of the Timex/Sinclair 2068.

Designer Jeff Burrell explored a number of potential solutions before arriving at the board you now own. While working on the TS-Pico SD storage board project, he decided to investigate the potential of the Raspberry Pi Pico to generate video. The result is a highly capable and affordable solution for the TS 2068 video.

Jeff didn't stop there: the PicoVideo also offer ULA Plus modes. ULA Plus is an enhanced ULA for the Sinclair ZX Spectrum family of computers.

The PicoVideo board builds on the success of other projects, most notably V. Hunter Adams' *PIO Assembly VGA Driver for RP2040 (Raspberry Pi Pico)*.

You can learn more about the VGA driver project here: https://vanhunteradams.com/Pico/VGA/VGA.html

# Using the PicoVideo

The PicoVideo is a plug-and-play solution: there are no special steps required to make it work.

First, make sure your Timex/Sinclair 2068 computer is turned off.

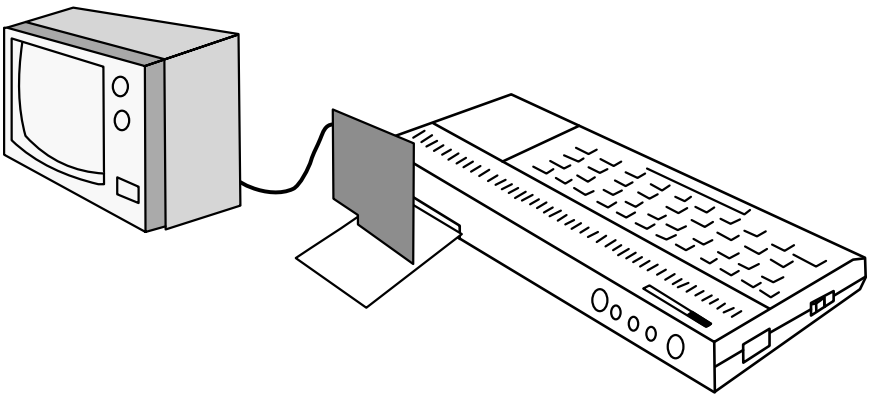
Next, plug your PicoVideo board in to an expansion bus. It's easiest to plug the board in to the bus while it is **not** attached to your computer.

Plug your expansion bus and PicoVideo combination in to your computer. Make sure it is firmly connected.

Connect the VGA output of your PicoVideo to your monitor. If you're using a VGA to HDMI converter, connect it to the converter.

You can use the 3.5mm jack on the PicoVideo board to send the audio from your TS 2068 to your monitor. Use a 3.5mm cable to connect the audio out from the PicoVideo to the audio input of your monitor, VGA to HDMI converter, or other speaker system.

Once everything is connected, turn on your monitor and your TS 2068. You should see a very clear display like the one below.



# How It Works

The PicoVideo monitors writes to the 2068 display file and to ports FEh (keyboard, cassette and border) and FFh (enhancement port). It copies writes to the display files to its own internal memory and reformats them to work with the VGA library.

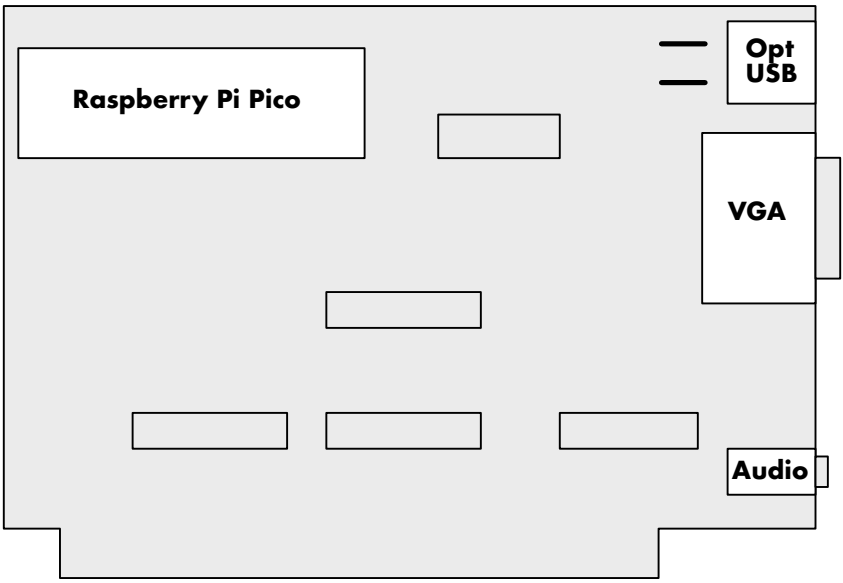
Through these ports, the PicoVideo tracks the current display mode and display file use. In high resolution modes, the PicoVideo monitors both display files and generates the appropriate display signals.

Because it monitors the display file memory, certain visual effects, like the color bars when loading from tape, are not reproduced. You may find that some video demos that use advanced interrupt techniques do not reproduce well on the PicoVideo.

The PicoVideo accurately reproduces the 15 colors (plus black) of the TS 2068, as well as the BRIGHT, FLASH, and INVERSE commands.

## USB-A Port

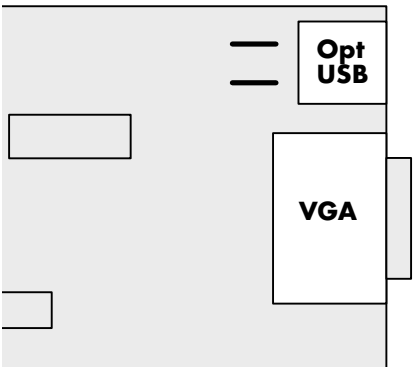
The PicoVideo has an a foot print for a USB-A port in the upper right corner of the printed circuit board. This can be used to



power a small VGA to HDMI converter.

While it's possible that your TS 2068 can provide the power to a VGA to HDMI converter, we recommend using an external power supply with your expansion bus and enabling that power to the the PicoVideo slot for optimal results.

You'll need to solder a USB-A connector and two jumpers, as shown in the illustration below, to enable the port.



# What Can You Do With It?

The PicoVideo requires no special programming to work with your existing programs. You should be able to load and run all software written for the TS 2068 and see it in full color.

The PicoVideo will work with almost all ZX Spectrum software that runs on the TS 2068 with a Spectrum ROM. Advanced demo programs that push the ZX Spectrum's video generation to the edge of its capabilities might not be reproduced faithfully.

## Advanced Video Modes

For an introductory description of TS2068 video modes, see Appendix C of the ***TS2068 Personal Color Computer User Manual***. This section describes how to put the video hardware into the various video modes. It also describes the memory organization of the TS2068, including the areas of memory used in video display, knowns as display files (DFiles).

In each video mode, the display files have the following uses:

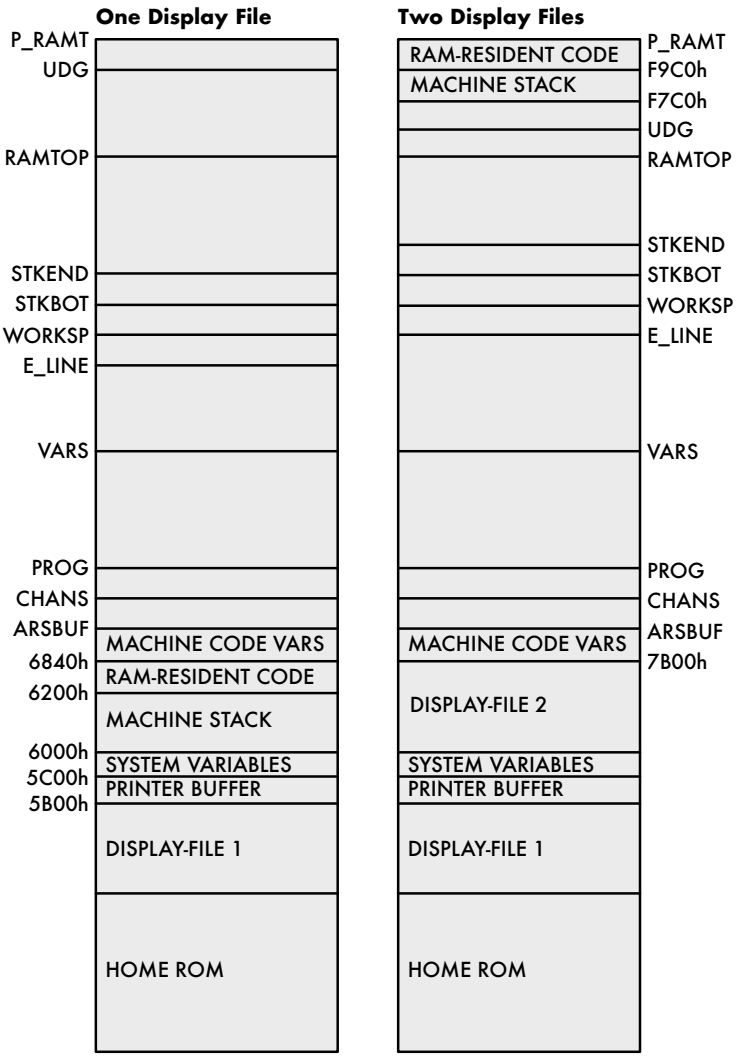
| | |
|---|---|
| Normal Mode | DFile 1 contains pixel and attribute data for the entire screen as described in Appendix C.<br>DFile 2 is not used. |
| Alternate Display File Mode | DFile 1 is not used.<br>DFile 2 contains pixel and attribute data for the entire screen just as DFile 1 does in normal mode. |
| 64-Column Mode | DFile 1 contains pixel data for the even numbered columns of the screen.<br>DFile 2 contains pixel data for the odd numbered columns of the screen.<br>Screen attributes are set by the value used to enable 64 column mode as described in Appendix C. |

| | |
|---|---|
| Extended Color Mode | DFile 1 contains pixel data for the entire screen. DFile 2 contains attribute data for the entire screen, one attribute byte for each scan of each character position. The organization of the attributes is described in Appendix C. |

In order to use the advanced video modes and still be able to use BASIC, the memory organization of the TS2068 must be modified. This is because the second display file (DFile 2) resides from 6000h to 7AFFh and memory starting at 6000h is used by the system software.

See the memory map diagram (left) and Appendix C of the User Guide for more information on the differences between one and two display files.

## Home Memory Map



# ULAplus Programming

The PicoVideo implements the ULAplus™ specification. You can use these features to generate video that surpasses the abilities of the stock TS 2068.

The ULAplus™ specification describes an enhanced ULA for the Sinclair ZX Spectrum family of computers. It is designed for maximum compatibility with existing software.

## I/O ports

ULAplus™ is controlled by two ports. Port BF3B is the register port (write only). The byte output is interpreted as follows:

- **Bits 0-5:** Select the register sub-group
- **Bits 6-7:** Select the register group.

Two groups are available:

- **00 - Palette Group.** When this group is selected, the sub-group determines the entry in the palette table (0-63).
- **01 - Mode Group.** The sub-group is (optionally) used to mirror the video functionality of Timex port FF and the framebuffer select of port 7FFD as follows:
  - bits 0-2: Screen Mode
    - 000: screen 0 (bank 5)
    - 001: screen 1 (bank 5)
    - 010: hi-color (bank 5)
    - 110: hi-res (bank 5)
    - 100: screen 0 (bank 7)
    - 101: screen 1 (bank 7)
    - 011: hi-color (bank 7)
    - 111: hi-res (bank 7)
  - bits 3-5: Screen Color in Hi-Res Mode (foreground/background)
    - 000: 0 / 7
    - 001: 1 / 6
    - 010: 2 / 5
    - 011: 3 / 4

- 100: 4 / 3
- 101: 5 / 2
- 110: 6 / 1
- 111: 7 / 0

Port FF3B is the data port (read/write). When the palette group is selected, the byte written will describe the color. When the mode group is selected, the byte output will be interpreted as follows:

bit 0: ULAplus™ palette on (1) or off (0)

Reading from port FF3B returns the last data byte written to the currently selected register. This can be used to read back the current palette or determine if palette mode is active.

Implementations that support the Timex video modes use port FF as the primary means to set the video mode, as per the Timex machines. It is left to the individual implementations to determine if reading the port returns the previous write or the floating bus.

**Note:** Modes 2 to 255 are reserved for future use. Groups 10 and 11 are reserved for future use.

### GRB Palette Entries

For a device using the GRB color space, the palette entry is interpreted as follows:

- bits 0-1: Blue Intensity
- bits 2-4: Red Intensity
- bits 5-7: Green Intensity

This color space uses a sub-set of 9-bit GRB. The missing lowest blue bit is set to OR of the other two blue bits (Bb becomes 000 for 00, and Bb1 for anything else). This gives access to a fixed half the potential 512 color palette. This reduces the jump in intensity in the lower range in the earlier version of the specification. It also means the standard palette can now be represented by the ULAplus™ palette.

## Limitations

Although in theory 64 colors can be displayed at once, in practice this is usually not possible except when displaying color bars,

because the four CLUTs are mutually exclusive; it is not possible to mix colors from two CLUTs in the same cell. However, with software palette cycling it is possible to display all 256 colors on screen at once.

## Emulation

The 64 color mode look-up table is organized as four palettes of 16 colors.

Bits `7` and `6` of each attribute byte, normally used for `FLASH` and `BRIGHT`, are used as an index value (`0-3`) to select one of the four color palettes.

Each color palette has 16 entries: eight for `INK`, and eight for `PAPER`. Bits `0-2` (`INK`) and `3-5` (`PAPER`) of the attribute byte are used as indexes to retrieve color data from the selected palette.

With the standard Spectrum display, the `BORDER` color is the same as the `PAPER` color in the first CLUT. For example `BORDER 0` would set the border to the same color as `PAPER 0` (with the `BRIGHT` and `FLASH` bits not set).

- The complete index can be calculated as:
- *ink color* = (FLASH * 2 + BRIGHT) * 16 + INK
- *paper color* = (FLASH * 2 + BRIGHT) * 16 + PAPER + 8

When scaling 3-bits of color data to more bits for emulators that operate in high color mode, simply concatenate the bits repeatedly and then truncate to as many bits as needed. For example, for 8-bits the following conversion should be used:

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| h | m | l | h | m | l | h | m |

where h is the high bit, m is the middle bit, and l is the low bit of the original 3-bit value.

With the Timex hi res display, the `BORDER` color is the same as the `PAPER` color in the second CLUT. Bits `3-5` of port `FF` set the `INK`, `PAPER`, and `BORDER` values to the following ULAplus™ palette registers:

| BITS | INK | PAPER | BORDER |
|---|---|---|---|
| 000 | 24 | 31 | 31 |
| 001 | 25 | 30 | 30 |
| 010 | 26 | 29 | 29 |
| 011 | 27 | 28 | 28 |
| 100 | 28 | 27 | 27 |
| 101 | 29 | 26 | 26 |
| 110 | 30 | 25 | 25 |
| 111 | 31 | 24 | 24 |

# Extension to the ZX-State (SZX) Format

## ZXSTPALETTE

The state of the ULA registers found in the 64 color replacement ULA. This block may be present for any machine.

```
// Palette Block flags
#define ZXSTPALETTE_DISABLED 0
#define ZXSTPALETTE_ENABLED 1

// Palette Block (contains the palette register values)
typedef struct _tagZXSTPALETTEBLOCK
{
  ZXSTBLOCK blk;
  BYTE chFlags;
  BYTE chCurrentRegister;
  BYTE chPaletteRegs[64];
  BYTE chFf;
} ZXSTPALETTEBLOCK, *LPZXSTPALETTEBLOCK;
```

## Members

### blk

The block header. The block id is ZXSTBID_PALETTE ('P', 'L', 'T', 'T').

### chFlags

A flags that indicates if the palette is enabled or if the normal display mode is in use. This can be one of:

- ZXSTPALETTE_DISABLED / Normal palette mode with BRIGHT and FLASH
- ZXSTPALETTE_ENABLED / 64 color palette mode

### chCurrentRegister

The currently selected palette register (`0-63`).

### chPaletteRegs

The current values of the palette registers.

### chFf

The current value of port `FF` which controls the Timex screen mode, and high resolution colors.

# Extension to the SCR format

- A 6912 byte .SCR file contains a standard Spectrum screen.
- A 6976 byte .SCR file contains a standard Spectrum screen followed by 64 color registers.
- A 12288 byte .SCR file contains a Timex hi-color screen.
- A 12352 byte .SCR file contains a Timex hi-color screen followed by 64 color registers.
- A 12289 byte .SCR file contains a Timex hi-res screen.
- A 12353 byte .SCR file contains a Timex hi-res screen followed by the hi-res color information that was dumped from port 255, followed by 64 color registers.

# Palette File Format

The palette format doubles as the BASIC patch loader. This enables you to edit patches produced by other people.

```
; 64 color palette file format (internal) - version 1.0
; Copyright (c) 2009 ZX Design and Media
;
; The palette file is stored as a BASIC program with
embedded machine code

header:
    defb 0x00;                          program file
    defb 0x14, 0x01, "64colour";        file name
    defw 0x0097;                        data length
    defw 0x0000;                        autostart line
    defw 0x0097;                        program length

; 0 RANDOMIZE USR ((PEEK VAL "2
; 3635"+VAL "256"*PEEK VAL "23636"
; )+VAL "48"): LOAD "": REM
basic:
    defb 0x00, 0x00, 0x93, 0x00, 0xf9, 0xc0, 0x28, 0x28
    defb 0xbe, 0xb0, 0x22, 0x32, 0x33, 0x36, 0x33, 0x35
    defb 0x22, 0x2b, 0xb0, 0x22, 0x32, 0x35, 0x36, 0x22
    defb 0x2a, 0xbe, 0xb0, 0x22, 0x32, 0x33, 0x36, 0x33
    defb 0x36, 0x22, 0x29, 0x2b, 0xb0, 0x22, 0x34, 0x38
    defb 0x22, 0x29, 0x3a, 0xef, 0x22, 0x22, 0x3a, 0xea

start:
    di;             disable interrupts
    ld hl, 38;      HL = length of code
    add hl, bc;     BC = entry point (start) from BASIC
    ld bc, 0xbf3b;  register select
    ld a, 64;       mode group
    out (c), a;
    ld a, 1;
    ld b, 0xff;     choose register port
    out (c), a;     turn palette mode on
    xor a;          first register

setreg:
    ld b, 0xbf;     choose register port
    out (c), a;     select register
    ex af, af';     save current register select
    ld a, (hl);     get data
    ld b, 0xff;     choose data port
```

```
    out (c), a;      set it
    ex af, af';      restore current register
    inc hl;          advance pointer
    inc a;           increase register
    cp 64;           are we nearly there yet?
    jr nz, setreg;   repeat until all 64 have been done
    ei;              enable interrupts
    ret;             return

; this is where the actual data is stored.; The following
is an example palette.

registers:
    defb 0x00, 0x02, 0x18, 0x1b, 0xc0, 0xc3, 0xd8, 0xdb;
INK
    defb 0x00, 0x02, 0x18, 0x1b, 0xc0, 0xc3, 0xd8, 0xdb;
PAPER
    defb 0x00, 0x03, 0x1c, 0x1f, 0xe0, 0xe3, 0xfc, 0xff;
+BRIGHT
    defb 0x00, 0x03, 0x1c, 0x1f, 0xe0, 0xe3, 0xfc, 0xff;
    defb 0xdb, 0xd8, 0xc3, 0xc0, 0x1b, 0x18, 0x02, 0x00;
+FLASH
    defb 0xdb, 0xd8, 0xc3, 0xc0, 0x1b, 0x18, 0x02, 0x00;
    defb 0xff, 0xfc, 0xe3, 0xe0, 0x1f, 0x1c, 0x03, 0x00;
+BRIGHT/
    defb 0xff, 0xfc, 0xe3, 0xe0, 0x1f, 0x1c, 0x03, 0x00;
+FLASH

terminating_byte:
    defb 0x0d;
```

## Quick Palette Set

```
set_palette:
    ld c, 0x3b;           ULAplus™ port
    ld de, 0x00bf;        d = data, e = register
    ld hl, pal_end;       mode group register
    ld a, 65;             becomes 64

palette_loop:
    dec a;                next register
    ld b, e;              register port
    out (c), a;           select register
    ld b, d;              data port
    outd;                 out bc, (hl); dec hl; dec b
    and a;                was that the last register?
    jr nz, palette_loop;  set all 64 entries
    ret;                  done
```

```
palette:
    incbin "palette.bin"; 64 bytes of G3R3B2 palette
register values
pal_end:
    defb 1;               write 1 to register 64 to enable
ULAplus mode
```

## Standard Spectrum Palette for ULAplus™

```
;                          %G..R..B.
;standard
       black         equ   %00000000
       blue          equ   %00000010
       red                 equ   %00010100
       magenta             equ   %00010110
       green         equ   %10100000
       cyan          equ   %10100010
       yellow        equ   %10110100
       white         equ   %10110110

;bright
       b_black             equ   %00000000
       b_blue        equ   %00000011
       b_red         equ   %00011100
       b_magenta     equ   %00011111
       b_green             equ   %11100000
       b_cyan        equ   %11100011
       b_yellow      equ   %11111100
       b_white             equ   %11111111

;flash
       f_black             equ   %10110110
       f_blue        equ   %10110100
       f_red         equ   %10100010
       f_magenta     equ   %10100000
       f_green             equ   %00010110
       f_cyan        equ   %00010100
       f_yellow      equ   %00000010
       f_white             equ   %00000000

;flash_bright
       fb_black      equ   %11111111
       fb_blue             equ   %11111100
       fb_red        equ   %11100011
       fb_magenta    equ   %11100000
       fb_green      equ   %00011111
```

```
        fb_cyan             equ    %00011100
        fb_yellow     equ    %00000011
        fb_white      equ    %00000000

palette:
        defb black, blue, red, magenta
        defb green, cyan, yellow, white

        defb black, blue, red, magenta
        defb green, cyan, yellow, white

        defb b_black, b_blue, b_red, b_magenta
        defb b_green, b_cyan, b_yellow, b_white

        defb b_black, b_blue, b_red, b_magenta
        defb b_green, b_cyan, b_yellow, b_white
        defb f_black, f_blue, f_red, f_magenta
        defb f_green, f_cyan, f_yellow, f_white

        defb f_black, f_blue, f_red, f_magenta
        defb f_green, f_cyan, f_yellow, f_white

        defb fb_black, fb_blue, fb_red, fb_magenta
        defb fb_green, fb_cyan, fb_yellow, fb_white

        defb fb_black, fb_blue, fb_red, fb_magenta
        defb fb_green, fb_cyan, fb_yellow, fb_white
```

## Legal

# Upgrading the PicoVideo

While we don't expect you'll need to upgrade the firmware in the PicoVideo, you can upgrade it at home if necessary. Follow these instructions to upload a new firmware file to your PicoVideo.

1. Turn off your Timex/Sinclair 2068. If your have external power to your expansion bus, turn that off, too.

2. Disconnect the VGA and audio cables from your PicoVideo.

3. Gently remove the PicoVideo from your expansion bus.

4. Connect a USB cable with a USB Micro B connector to your computer.

5. Press and hold the BOOTSEL button on the Raspberry Pi Pico (it's a small white button, near the Micro B connector).

6. Plug the USB cable into the Raspberry Pi Pico.

7. Release the BOOTSEL button.

8. You should see the Raspberry Pi Pico show up on your computer as a storage device, like a thumb drive.

9. Click and drag the new firmware file (which ends in .uf2) to the Raspberry Pi Pico.

10. The firmware file will update the Raspberry Pi Pico and it will automatically unmount. Your computer will probably complain that it did not eject properly. That's ok.

11. Disconnect the USB cable from the PicoVideo board.

12. Plug the PicoVideo board back into your expansion bus.

13. Turn everything back on and you should be in business!