How to Program in Assembly Language

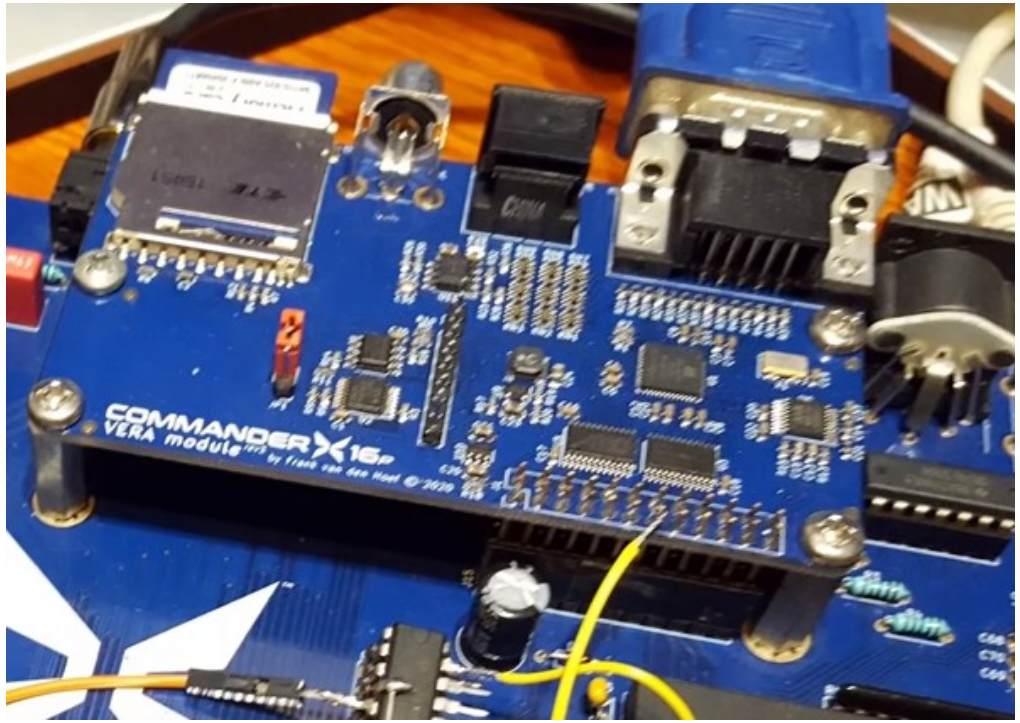
for the

COMMANDER X16™

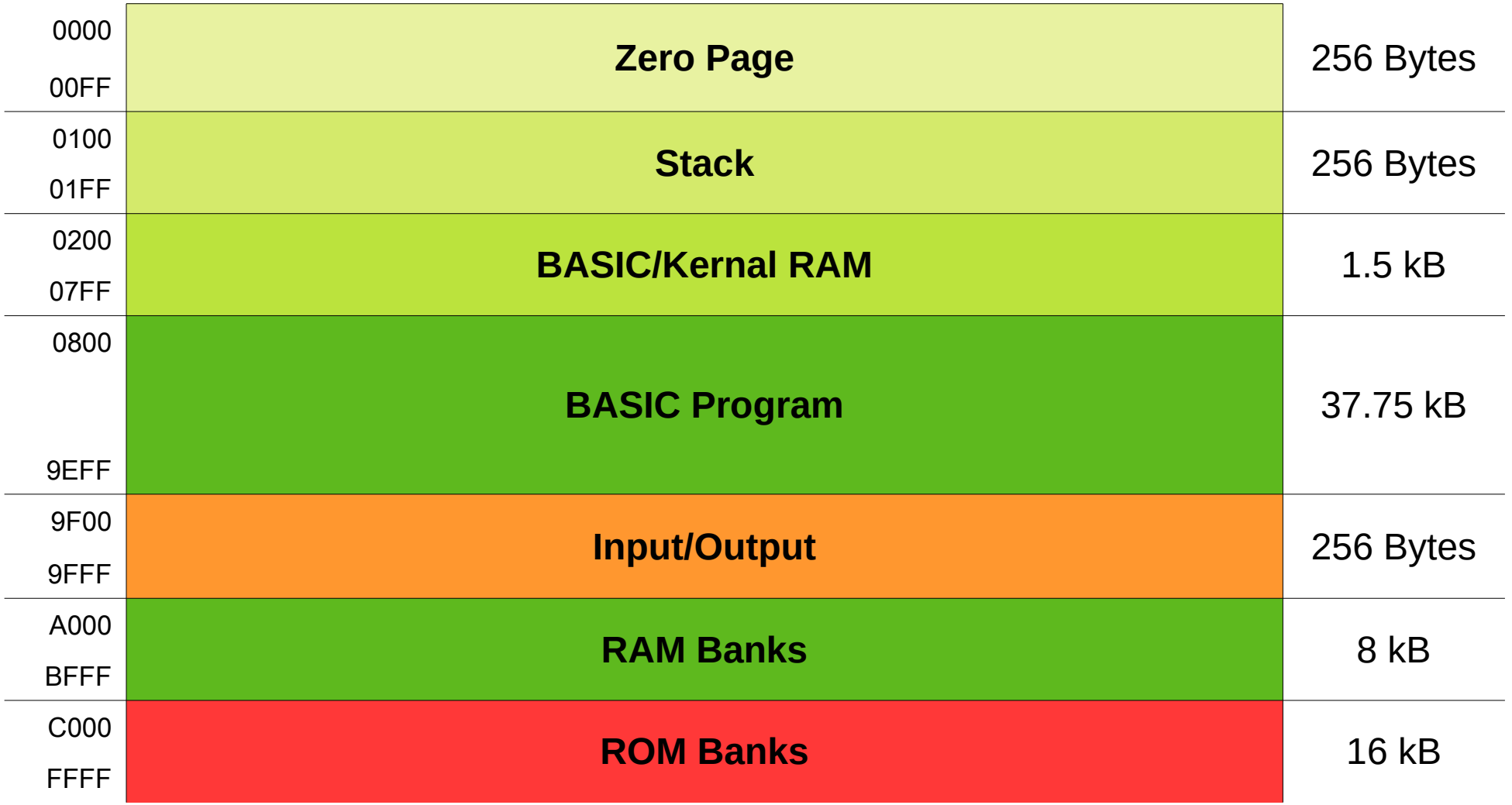
Lesson 9: Hello, VERA!

VERA: Versatile Embedded Retro Adapter

- 640x480 12-bit color graphics
  - VGA (Standard, Interlaced)
  - NTSC (Composite, S-Video)
  - 256 color palette
  - Bitmap, Tiles, Text and Sprites
- SD Card interface via SPI
- Programmable Sound Generator (PSG)
  - 16 stereo channels, 4 waveforms, variable pulse width
- Pulse Code Modulation Audio
  - Up to 48kHz 8-bit or 16-bit stereo sample playback



X16 Memory Map



VERA Registers

- I/O Address Space: $9F20-9F3F (32 bytes)
- Only way for CPU program to control and access VERA
- Two 8-bit data channels to 17-bit VRAM address space (128 kB)
- 8-bit data channel to 4 kB PCM FIFO (write only)
- 8-bit data channel to SPI controller

VRAM Interface

- CTRL: $9F25
  -

Reset (W/O)	-	-	-	-	-	DCSEL	ADDRSEL

  - Set address port (ADDRSEL) in bit 0
- ADDRx_H: $9F22
  -

Address Increment (Stride)	DECR	-	-	ADDR(16)

  - Set bit 16 of VRAM address in bit 0
- ADDRx_M (VRAM address bits 15:8): $9F21
- ADDRx_L (VRAM address bits 7:0): $9F20
- DATA0 (read or write byte from address 0): $9F23
- DATA1 (read or write byte from address 1): $9F24

Reading and Writing VRAM

- Copy byte from $02468 to $13579 in VRAM

```
VERA_addr_low  = $9F20
VERA_addr_high = $9F21
VERA_addr_bank = $9F22
VERA_data0     = $9F23
VERA_data1     = $9F24
VERA_ctrl      = $9F25
VRAM_SOURCE    = $02468
VRAM_DEST      = $13579
start:
   stz VERA_ctrl
   lda #^VRAM_SOURCE
   sta VERA_addr_bank
   lda #>VRAM_SOURCE
   sta VERA_addr_high
   lda #<VRAM_DEST
   sta VERA_addr_low
   lda VERA_data0
   sta VERA_data1
   sta VERA_addr_high
   lda #<VRAM_SOURCE
   sta VERA_addr_low
   lda #1
   sta VERA_ctrl
   lda #^VRAM_DEST
   sta VERA_addr_bank
   lda #>VRAM_DEST
```

VRAM Striding

- ADDRx_H: $9F22
Address Increment (Stride)	DECR	-	-	ADDR(16)
  - Set address increment (bits 7:4) to stride for each data access
  - Set DECR (bit 3) to 1 to decrement by the stride
- Stride Values (mostly, stride = 2^{N-1}):

$0:	No Stride	$1:	1 byte	$2:	2 bytes	$3:	4 bytes
$4:	8 bytes	$5:	16 bytes	$6:	32 bytes	$7:	64 bytes
$8:	128 bytes	$9:	256 bytes	$A:	512 bytes	$B:	40 bytes
$C:	80 bytes	$D:	160 bytes	$E:	320 bytes	$F:	640 bytes

Writing Array to Even Addresses

- Copy array bytes to even bytes in VRAM starting at $00000

```
VERA_addr_low  = $9F20
VERA_addr_high = $9F21
VERA_addr_bank = $9F22
VERA_data0     = $9F23
VERA_ctrl      = $9F25
VRAM_DEST      = $00000
array: .byte 8,5,12,12,15
start:
   stz VERA_ctrl
   lda #($20 | ^VRAM_DEST)
   sta VERA_addr_bank
            lda #>VRAM_DEST
            sta VERA_addr_high
            lda #<VRAM_DEST
            sta VERA_addr_low
            ldx #0
        @loop:
            lda array,x
            sta VERA_data0
            inx
            cpx #(start-array)
            bne @loop
```

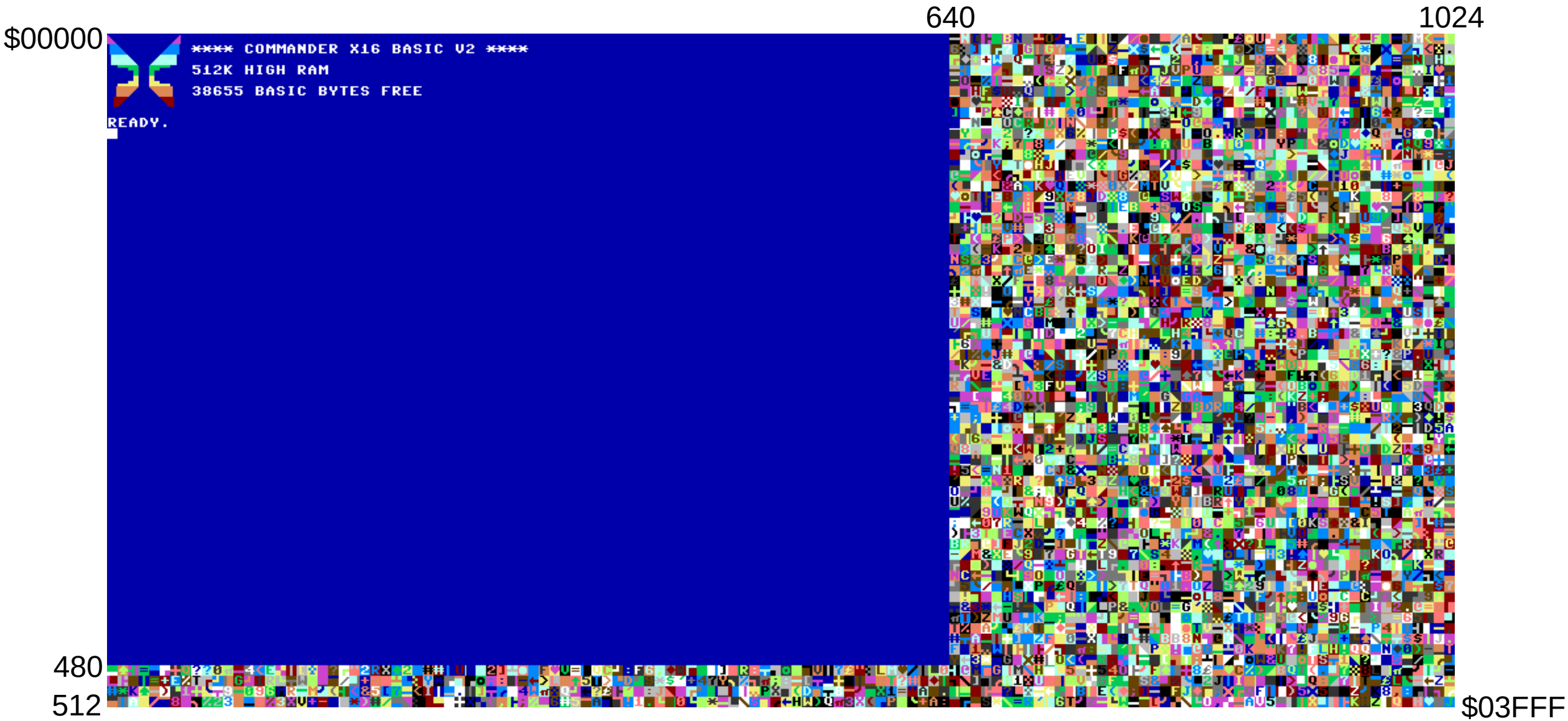
Wait, what?


HELLO
**** COMMANDER X16 BASIC V2 ****
512K HIGH RAM
38655 BASIC BYTES FREE

VERA Graphics Configuration

- DC_VIDEO: $9F29
Current Field	Sprites Enable	Layer1 Enable	Layer0 Enable	-	Chroma Disable	Output Mode
  - Default: $21 (Layer1 Enable, Output Mode = Standard VGA)
- DC_HSCALE: $9F2A
  - Default: 128 (640 pixels wide)
- DC_VSCALE: $9F2B
  - Default: 128 (480 pixels high)
- L0_CONFIG: $9F2D; L1_CONFIG: $9F34
Map Height	Map Width	T256	Bitmap Mode	Color Depth
  - Layer 1 Default: $60 (128x64 16-Color Text)
- L0_MAPBASE: $9F2E; L1_MAPBASE: $9F35
  - Layer 1 Default: $00 (Tile map starts at $00000 = $00 << 9)
- L0_TILEBASE: $9F2F; L1_TILEBASE: $9F36
Tile Base Address (16:11)	Tile Height	Tile Width
  - Layer 1 Default: $7C (8x8 tiles start at $0F800 = $7C << 9)

Layer 1 Tilemap



16-Color Text Map Entry

Byte 0	Character Screen Code	
Byte 1	Background Color	Foreground Color

- Character Screen Code ≠ PETSCII Code ≠ ASCII Code

Upper/Graphics

Lower/Upper

ISO 8859-1

- Background and Foreground colors from first 16 colors in palette

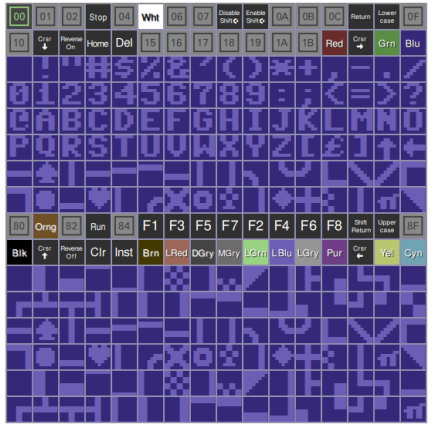
Screen Code vs. PETSCII Code

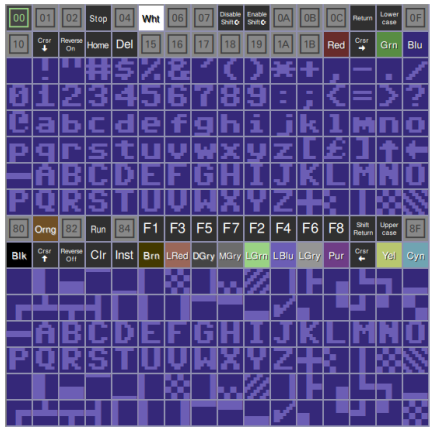
00000	DF 64
00002	20 64
00004	20 64
00006	20 64
00008	20 64
0000A	20 64
0000C	E9 64

08 64
05 64
0C 64
0C 64
0F 64
20 64
E9 64

08 61
05 61
0C 61
0C 61
0F 61
20 64
E9 64

PETSCII Codes:
(with control codes)



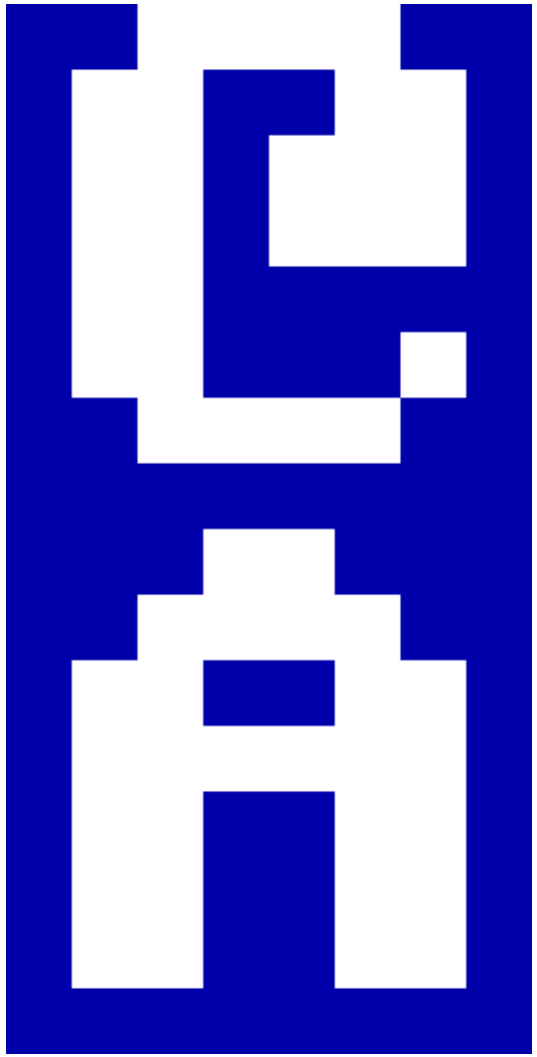


```
hello: .byte 72,69,76,76,79
start:
   clc
   ldx #0
   ldy #0
   jsr PLOT
@loop:
   lda hello,x
   jsr CHROUT
   inx
   cpx #5
```

8x8 1 Bit-per-Pixel (Text) Tiles

VRAM Addr = $0F800 + Code*8

0F800	00111100
0F801	01100110
0F802	01101110
0F803	01101110
0F804	01100000
0F805	01100010
0F806	00111100
0F807	00000000
0F808	00011000
0F809	00111100
0F80A	01100110
0F80B	01111110
0F80C	01100110
0F80D	01100110
0F80E	01100110
0F80F	00000000



256-Color Palette

- Reserved VRAM (512 bytes)
- 12-bit RGB color (2 byte entries)
  - Low Byte:

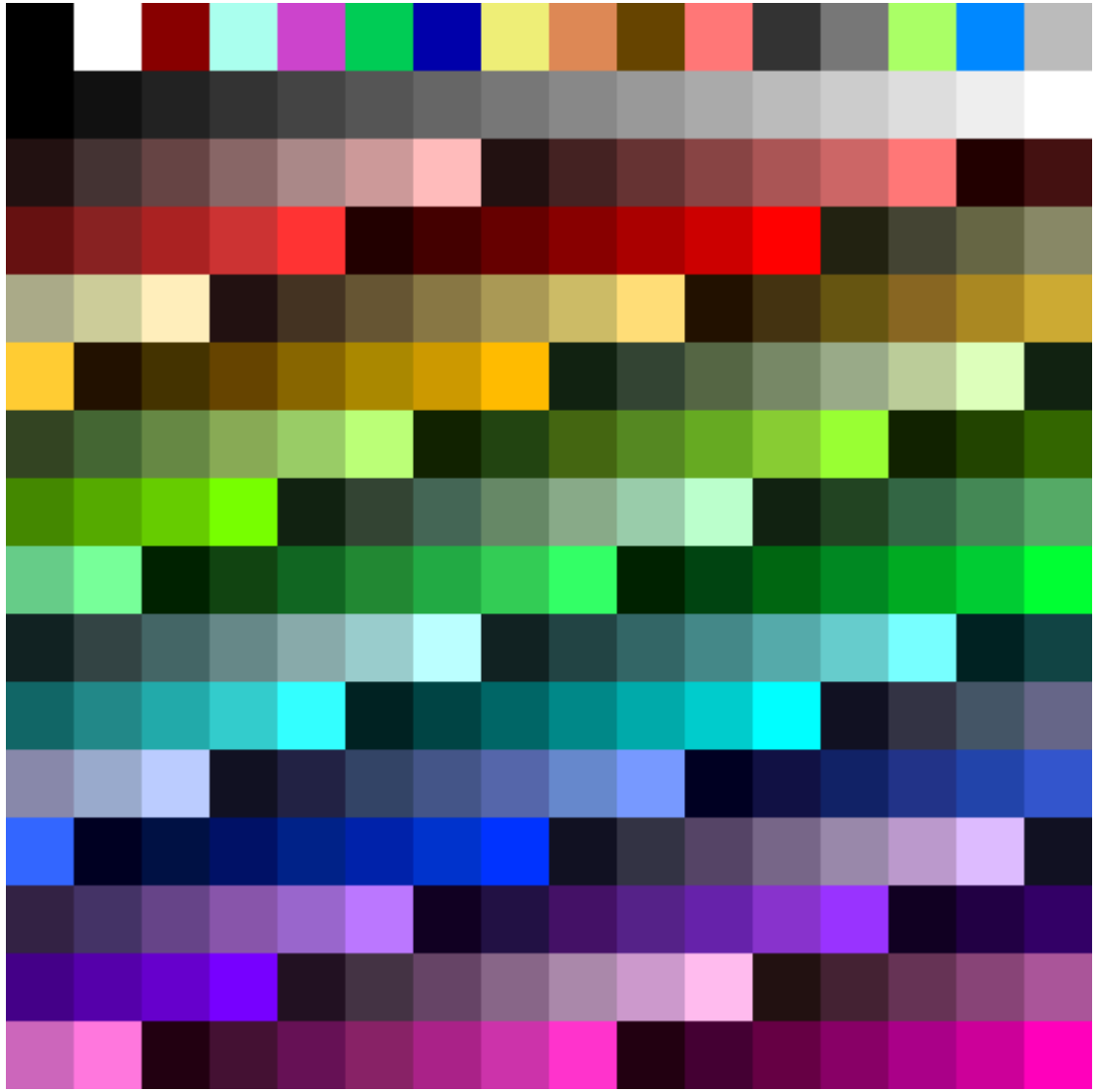
Green	Blue

  - High Byte:

0	Red

- 4-bit value * 17 ~ 8-bit value
  - e.g. $23,$01 ~ #112233
- Or, truncated 8-bit ~ 4-bit value
  - e.g. #456789 ~ $68,$04

$1FA00

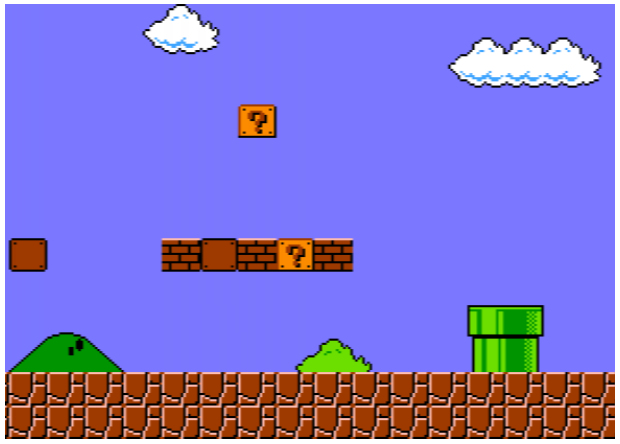


$1FBFF

Layer 0

- Disabled by default
- Rendered “behind” Layer 1
- Visible where Layer 1 color = color index 0 (default: black)
- Tied to same display scale, but otherwise independent of Layer 1

Layer 0

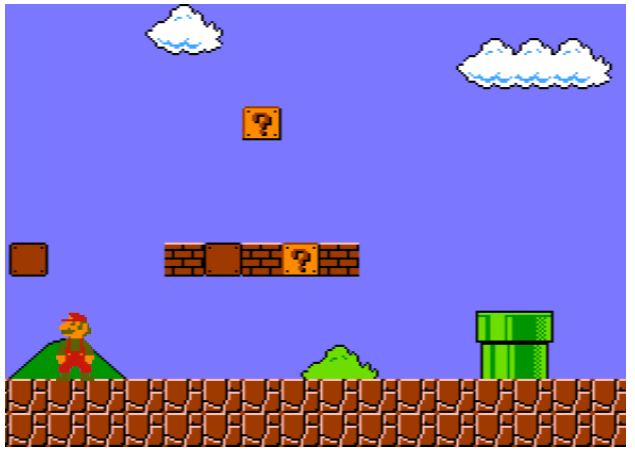


+

Layer 1

=

Display



Example Program

1-5: Set Foreground Color
0,6-9: Set Background Color
I: Zoom In
O: Zoom Out
R: Toggle Character Set
S: Convert @→☺
T: Toggle Layer 1
P: Toggle Layer 0
C: Toggle Color 1 White/Black