Acolyte Computer Design Document

By: Steven Chad Burrow

The Acolyte Computer uses a W65C02S microprocessor running at 3.14 MHz. It contains up to 48KB of RAM and up to 8 banks of ROM, each bank being 16KB. Supported features include VGA video display, PS/2 keyboard input, square wave audio output, and SPI interfacing an EEPROM and a Micro SD Card.

Memory Map:
$0000 - $07FF = System RAM
$0800 - $7FFF = Video RAM
$8000 - $87FF = Unused RAM
$8800 - $BFFF = BASIC RAM
$C000 - $FFFF = Banked ROM

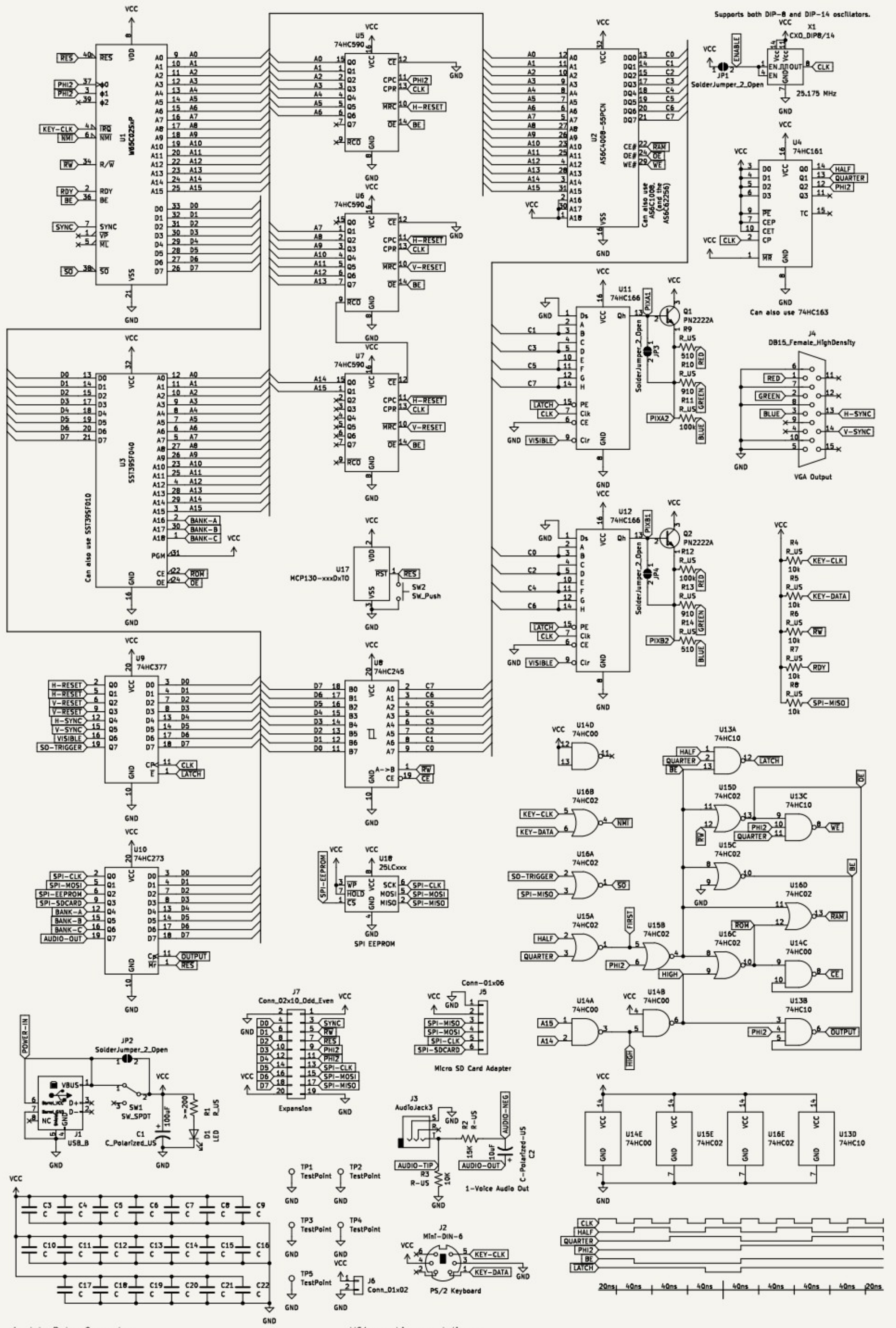
Primary Bank:
$C000 - $CFFF = OS and Monitor ROM
$D000 - $DFFF = BASIC ROM
$E000 - $F5FF = Data tables
$F600 - $FFFF = System ROM

The VGA video display has a resolution of 320x240 using 4 colors.

Output is done by writing to ROM, where the data bus is captured by a latch IC.

Input is through /IRQ, /NMI, and /SO lines.

The design goal of this system is to minimize chip count, while maximizing capability of required chips. It is also designed to be easy to install and understand, while still providing a full user experience.



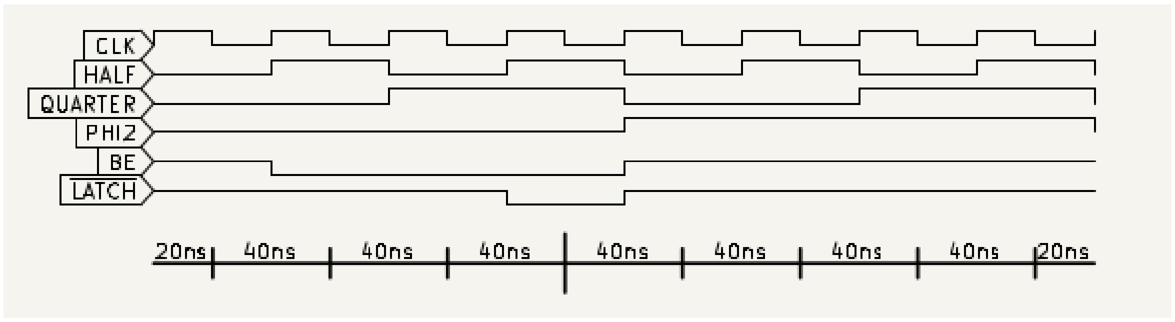
Acolyte Retro Computer
By: Steven Chad Burrow
W65C02 running at 3.14 MHz
18KB General Purpose RAM
30KB Video RAM
16KB Banked ROM
  (up to 8 banks)

Memory Map:
$0000-$07FF = RAM
$0800-$7FFF = Video RAM
$8000-$BFFF = RAM
$C000-$FFFF = ROM (banked)

VGA graphics resolutions:
320x240 4-colors
  (black, orange, blue, white)

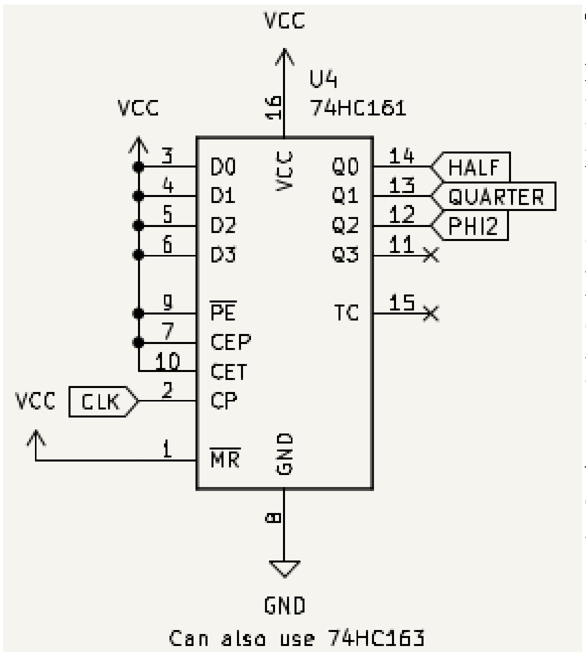
PS/2 Keyboard
SPI EEPROM and SDcard
Square Wave Audio

Timing:



The overall design of the Acolyte Computer is focused around the video circuit. While the BE pin on the processor is high, it keeps the address and data buses online and in normal operation. When the BE pin goes low, the address and data buses are in a high-Z state from the processor, allowing for the video counters to access the RAM for color data and the ROM for video sync and reset signals. The BE pin is kept high during the first quarter of PHI2-low because the processor reads data from the data bus on the falling edge of PHI2. /LATCH is during the last quarter of PHI2-low, where both the color data and video sync signals are latched for the rest of the cycle.

Divider:

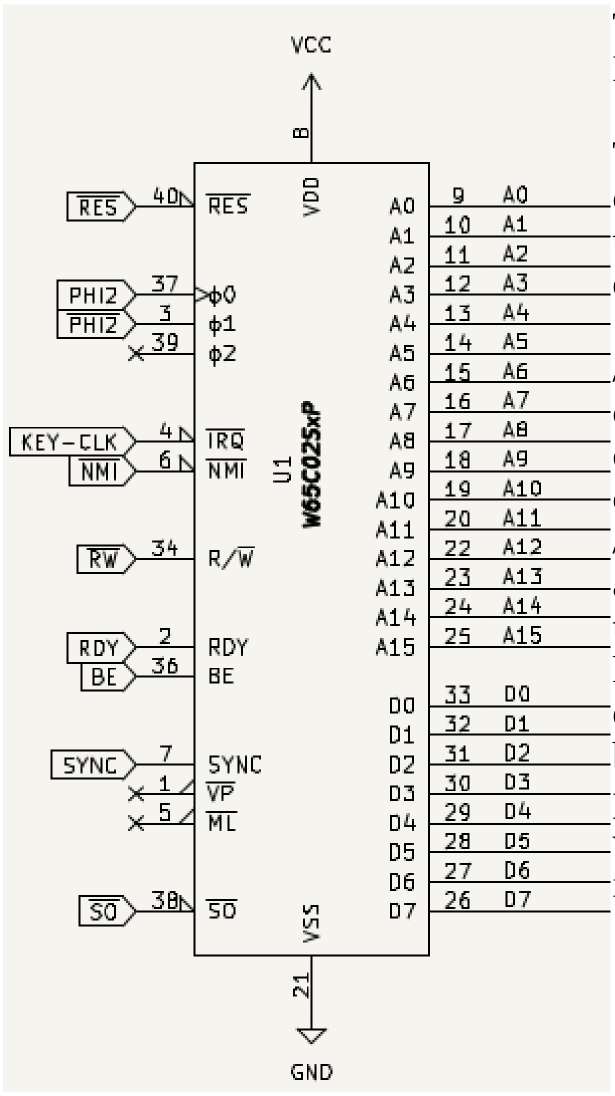


The divider is a 74HC161 counter. It could be replaced with a 74HC163 with no changes. This IC simply divides the CLK signal down so that it is usable for different timing logic situations.

CLK is running at 25.175 MHz, so thus:
HALF is 12.5875 MHz,
QUARTER is 6.29375 MHz, and
PHI2 is 3.146875 MHz.

All other control signals are tied high so that the counter is never reset or loaded, but is allowed to free run from CLK.

Processor:

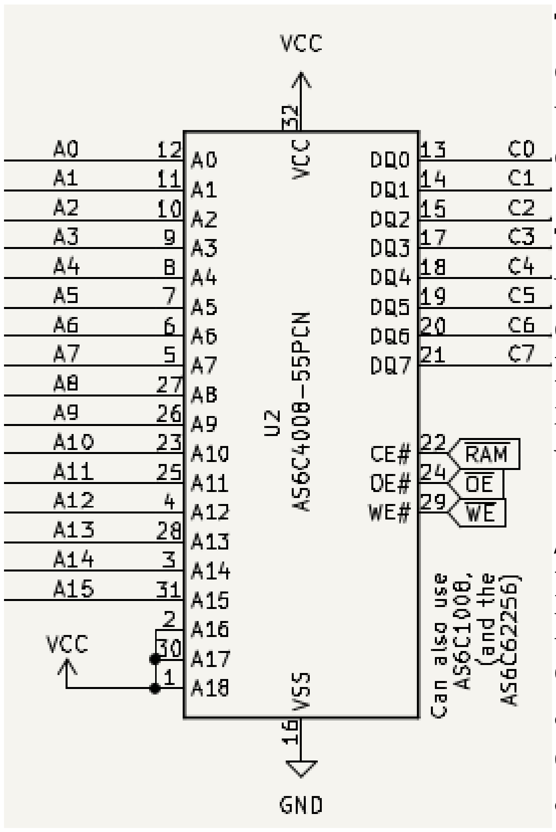


The W65C02S microprocessor running at 3.14 MHz is at the heart of the Acolyte Computer.

The usual address and data buses are connected. Other typical signals used are the /RES reset line, PHI2 clock input, /RW direction output, and the RDY pulled up.

/PHI2 is not used on this board, but is connected to the expansion port. KEY-CLK comes from the PS/2 keyboard and is directly connected to the /IRQ interrupt. Edge-sensitive /NMI is KEY-CLK nor KEY-DATA, which allows for the interrupt to trigger halfway through the keyboard's incoming signal. BE is high for all of PHI2-high, as well as the first quarter of PHI2-low. SYNC is not used on this board, but is connected to the expansion port. /SO is SPI-MISO nor SO-TRIGGER, which could set the overflow flag shortly after it is cleared.

RAM:



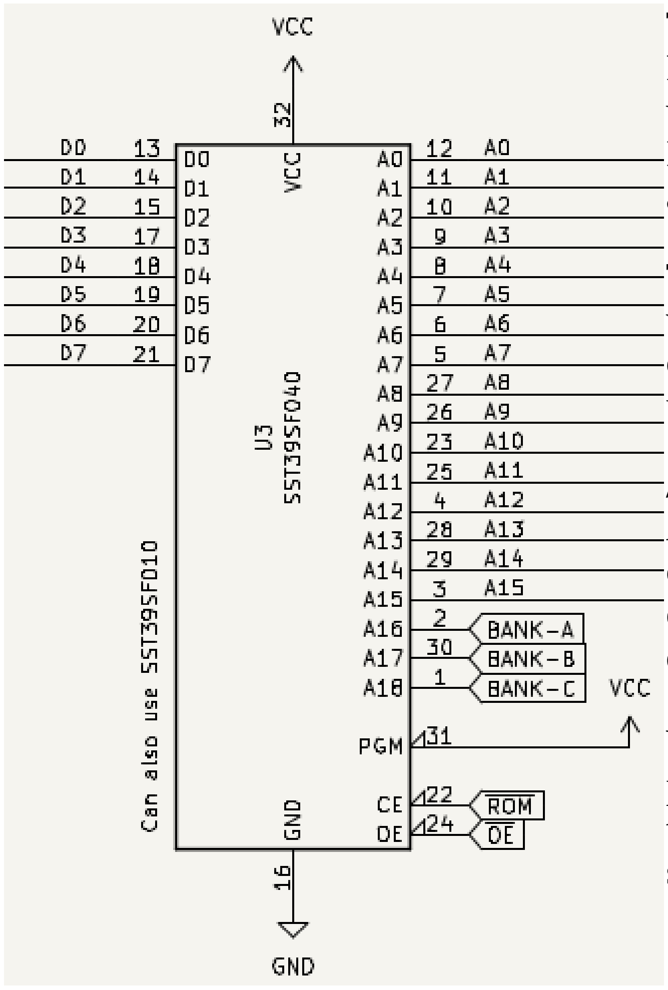
The RAM used is the AS6C1008-55 SRAM. It contains 128KB of RAM, yet only 48KB of it is used on this system. The board allows for the use of the AS6C4008-55 as well.

The AS6C62256-55 chip is also supported, though with only 32KB of RAM any programs using the extra 16KB of RAM will not work. This includes BASIC. It has been coded to not glitch the system in the event that the user tries to program in BASIC while only having 32KB of RAM.

/RAM is enabled when either BE is low or when it ROM is not being accessed. /OE is low during BE-low or determined by /RW when PHI2 is high. /OE on the RAM chip could be permanently grounded and the system will still work. /WE is never enabled while BE is low, but is determined by /RW and qualified to the second half of PHI2-high.

The C data bus is connected to the D data bus through a transceiver IC. While BE is low, the transceiver disconnects the C and D data buses. At that time, the address bus is populated by the video counters so that the RAM can push color data onto the shift registers. While BE is high, the transceiver connects the C and D data buses only when accessing RAM.

ROM:



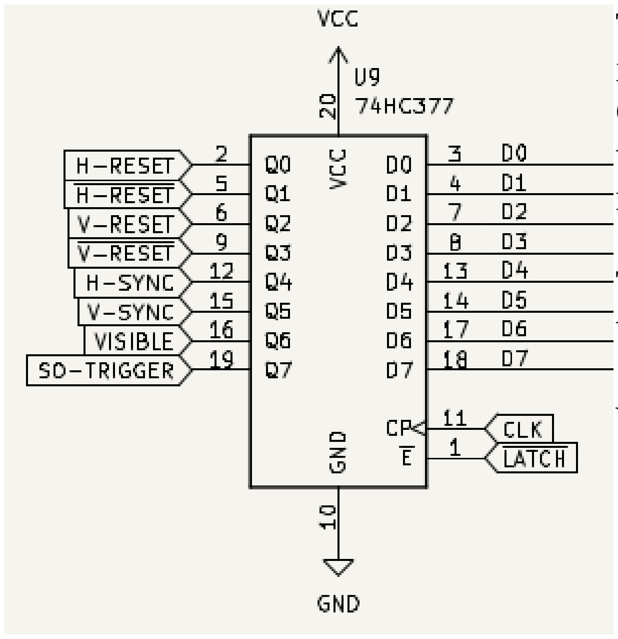
The ROM used is the SST39SF010-70 Flash ROM. It contains 128KB of memory, which technically only allows for 2 banks. If an SST39SF040-70 is used, it can access all 8 banks.

The top 48KB of each bank contains the video circuit’s sync and reset signals. Thus, only the bottom 16KB of each bank are usable for code and data.

/ROM is enabled when BE is low or when A15 and A14 are both high, which would only happen during PHI2-high. /OE is enabled when BE is low or when determined by /RW during PHI2-high.

When BE is low, the address bus is populated by the video counters, pushing the ROM data containing sync and reset signals onto a latch.

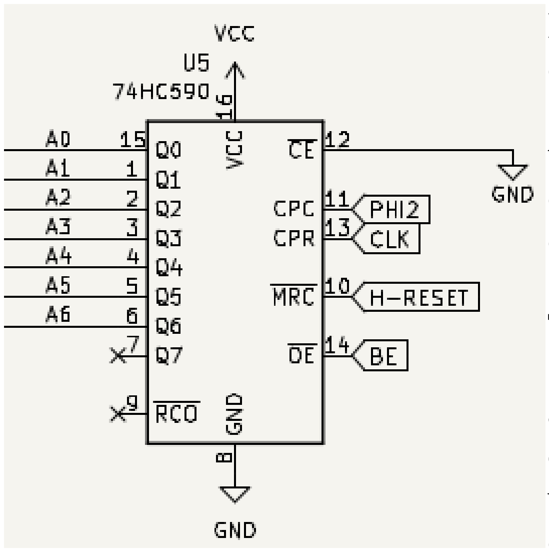
Video Latch:



The video latch is a 74HC377. When /LATCH is enabled during the last quarter of PHI2-low, CLK triggers the data bus to latched. During that time, the ROM is pushing video sync and reset signals onto the data bus.

The H-RESET and V-RESET signals are tied to the video counters, and the H-SYNC and V-SYNC signals go to the VGA output. VISIBLE will reset the color shift registers to ground. SO-TRIGGER toggles the /SO line when the SPI-MISO line is low.

Horizontal Counter:

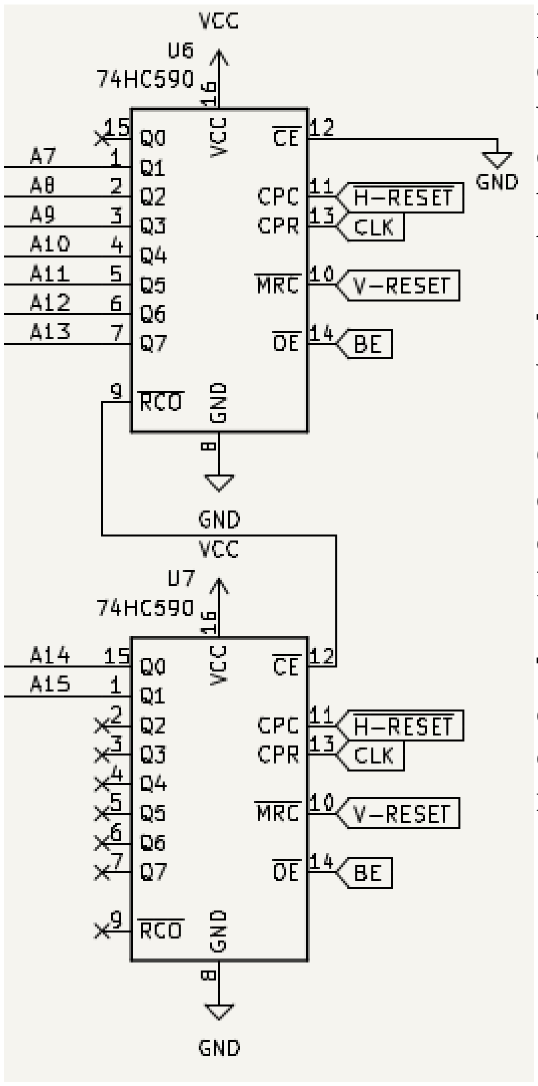


Each video counter is a 74HC590. The horizontal counter determines the addresses within each video scan line. When BE is high, the counter’s output is turned to high-Z, allowing for the processor to take control of the address bus. When BE is low, the counters control the address bus.

The counters change each PHI2 cycle. Because the CPC and CPR pins connected together would cause a delay by one cycle, CLK was instead used to quickly put the counter registers on the output pins when changed by PHI2. H-RESET resets the counter to zero.

Notice only the lower seven address lines are used here, thus able to access 128 bytes of memory. Though only 80 bytes of memory are used for color data, and 100 bytes of memory are used for sync and reset signals. The extra bytes of memory are overscanned, and can technically be used as variable data in RAM.

Vertical Counters:

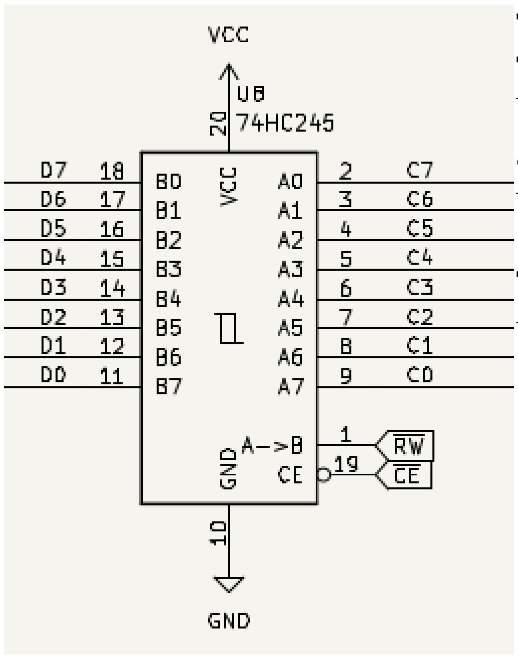


Each video counter is a 74HC590. The vertical counters determine the addresses differentiating the video scan lines. When BE is high, the counter's output is turned to high-Z, allowing for the processor to take control of the address bus. When BE is low, the counters control the address bus.

The counters change each /H-RESET cycle, thus when the horizontal counter is reset the vertical counters are incremented. Because the CPC and CPR pins connected together would cause a delay by one cycle, CLK was instead used to quickly put the counter registers on the output pins when changed by PHI2. V-RESET resets the counter to zero.

These counters are connected with the /RCO line, creating a cascading effect. The second vertical counter only has A14 and A15, and A15 stays low most of the time and for all of the visible area.

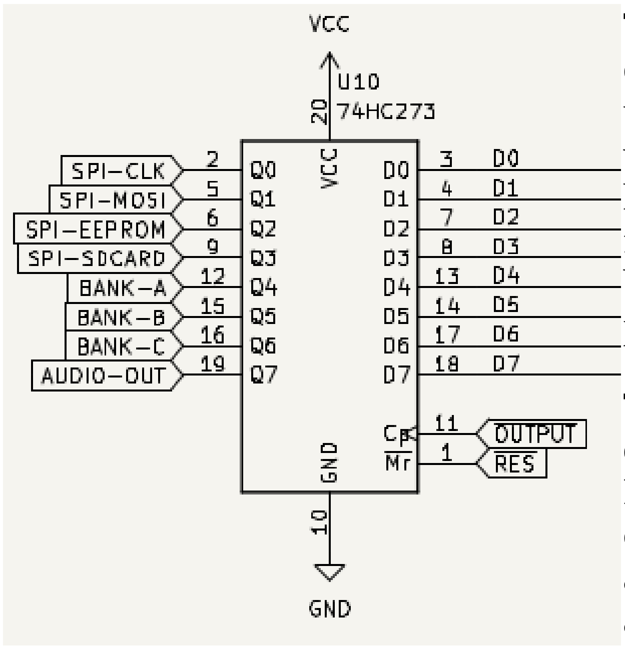
Transceiver:



The transceiver is a 74HC245 which is bi-directional. This either connects or disconnects the C and D data buses. The direction is determined by /RW. /CE is always disabled while BE is low, and only enabled while BE is high if RAM is being accessed.

This orientation of the pins was used to help route traces on the printed circuit board easier.

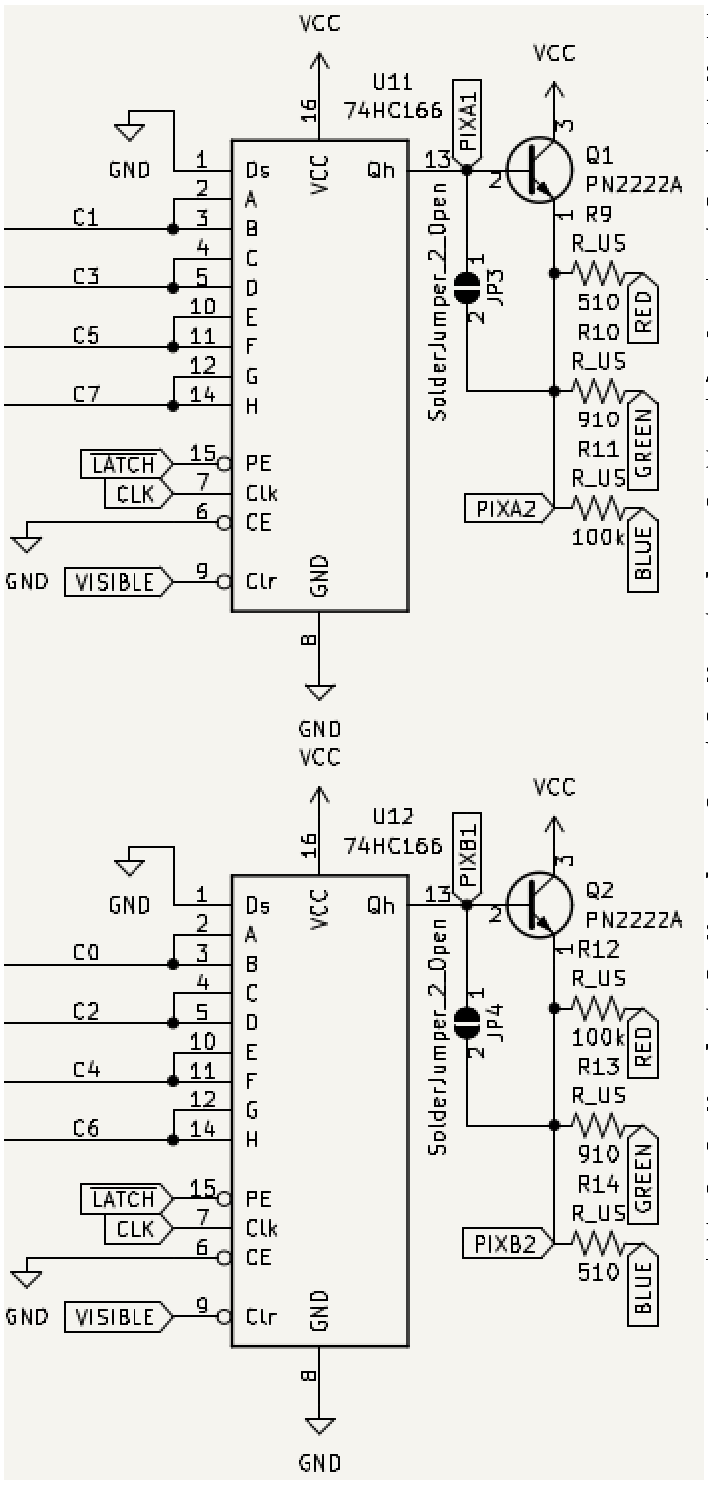
Output Latch:



The output latch is a 74HC273. This is essentially a replacement for the commonly used W65C22S VIA chip. /OUTPUT is low when writing to ROM, thus A14 and A15 are high and also PHI2 is high. When PHI2 falls, /OUTPUT rises, thus latching the data bus. /RES is the reset line used by the 6502 processor as well, grounding all lines.

The first 4 output pins are to control SPI devices. BANK-X pins are tied directly to the ROM’s higher address pins. AUDIO-OUT connects to a 3.5mm jack through a small audio circuit. Toggling the AUDIO-OUT pin at certain frequencies creates sound output.

Shift Registers:

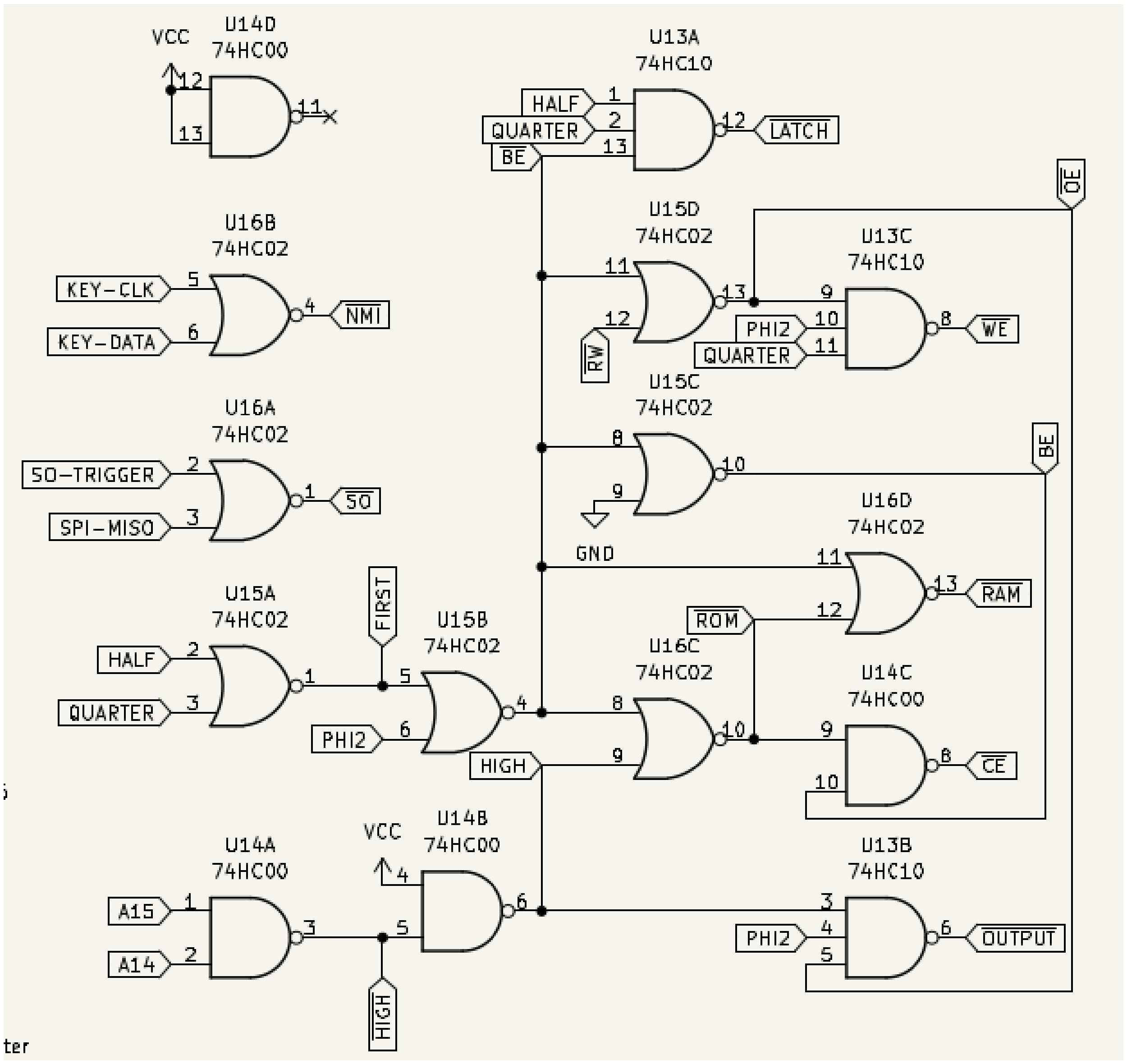


Each shift register is a 74HC166. These shift registers hold color data from the RAM. /LATCH happens at the same time as the video latch, during the last quarter of PHI2-low. A low signal on VISIBLE will ground the shift registers, thus all overscanned memory locations are never visible on the video display. /CE is permanently grounded, though if brought high it would disable these shift registers allowing for other ICs to push color data.

The C data bus is alternating between both shift registers. Because the first shift register controls the orange color, all odd bits likewise control orange. Even bits control the blue color. When combined it creates white color.

The PN2222A transistors are used to supply enough current to the video display, though are not needed when using the intended orange/blue colors. Thus, those could be bypassed with the solder jumper. Different color schemes could be used if the resistor values are changed, though the transistors might be needed to supply sufficient current to the VGA monitor.

Glue Logic:



The glue logic contains four ICs: 1x 74HC00, 2x 74HC02, and 1x 74HC10. Most of this logic is discussed earlier when talking over the other components.

One notable feature is how /NMI is KEY-CLK nor KEY-DATA. This causes the /NMI line to stay low when the keyboard is inactive. When the keyboard is active, the /NMI line will go high if both KEY-CLK and KEY-DATA are both low. This happens during the first half of the KEY-CLK cycle and when KEY-DATA has a zero value. When KEY-CLK rises during the second half of the cycle, the /NMI will always go back low. Thus if zero was present on KEY-DATA, the /NMI would have a falling edge causing an interrupt. If a one was present on KEY-DATA, /NMI would never rise, and no interrupt would occur.

Because of this style, the IRQ-ISR (interrupt subroutine) must wait for the /NMI interrupt to either occur or not occur. This works well as the /IRQ interrupt is level-sensitive, thus waiting to the second half of the KEY-CLK cycle when it is high again makes sense.

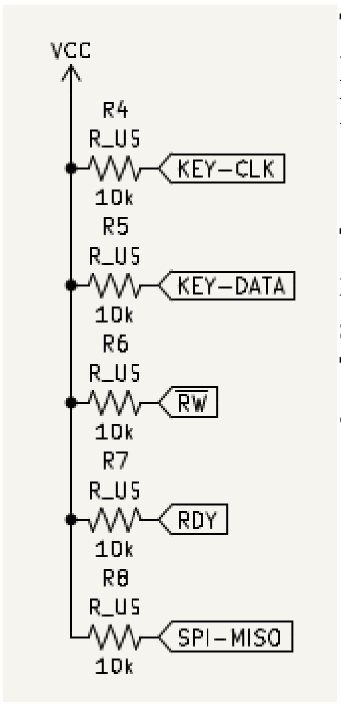
Another notable feature is how /SO is SPI-MISO nor SO-TRIGGER. SO-TRIGGER comes from the video sync and reset signals on the ROM, and thus has an adjustable frequency. Theoretically it could be replaced with pin #11 on the 74HC161 divider (which would be SIXTEENTH if labeled). SO-TRIGGER constantly toggles, but when the MISO line is not in use it is pulled high, so /SO stays low. If the MISO line were to fall, the /SO line would toggle with the SO-TRIGGER.

In code, upon waiting for a signal on the MISO line, the overflow flag should be cleared and then it should wait until the next SO-TRIGGER cycle to see if /SO toggled. Because the falling edge of /SO sets the overflow flag, after just a short wait a branch statement can be used to determine if the MISO line was either high or low.

Lastly, /WE must be qualified with both PHI2 and QUARTER, thus only allowing writing to RAM during the second half of PHI2-high. While testing, a 55ns SRAM ran so fast that spurious data was written in random locations, causing the computer to glitch. This is because the BE line goes high when PHI2 goes high, but there is still some time that the address and data buses are not yet correct.

As of now, there is one spare NAND gate. It will be used for some extra feature eventually.

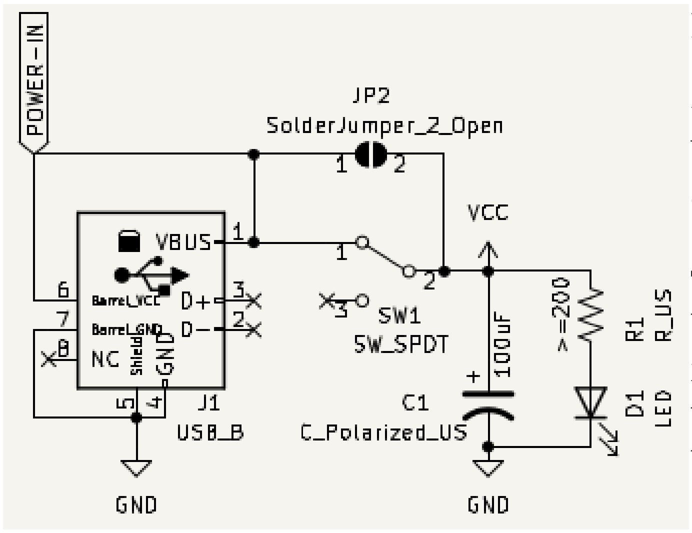
Pull Up Resistors:



The pull-up resistors are all 10K ohms, a standard value. Both KEY-CLK and KEY-DATA require pull-up resistors for the keyboard to work correctly. RDY is not utilized on this system. SPI-MISO needs to be pulled up for normal operation.

The oddity is a pull-up on /RW. This is most likely not required, its removal should not affect the system. But it is included for safety, because when BE goes low, the /RW line goes high-Z. Though the glue logic already compensates for this, and will not allow spurious writes to RAM during BE-low, extra safety is best.

Power Supply:



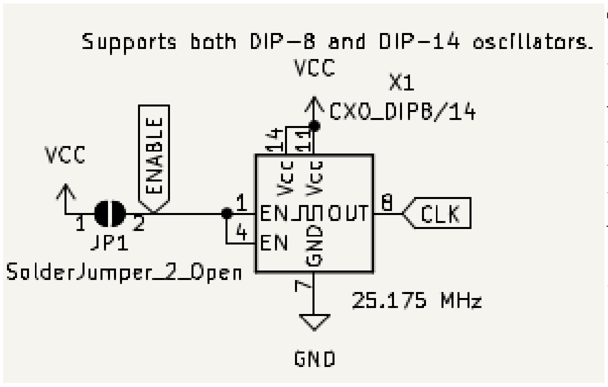
Power is supplied to the printed board at +5V and 0V GND. POWER-IN can come from either the USB connector or the barrel jack. The footprint allows for either connector. At least 2 amps is needed.

The SPDT switch connects POWER-IN to VCC. A bypass solder jumper is available if not wanting to use a switch at all, but turning off the computer by unplugging the power cord.

The 100uF polarized capacitor helps regulate power for the entire system.

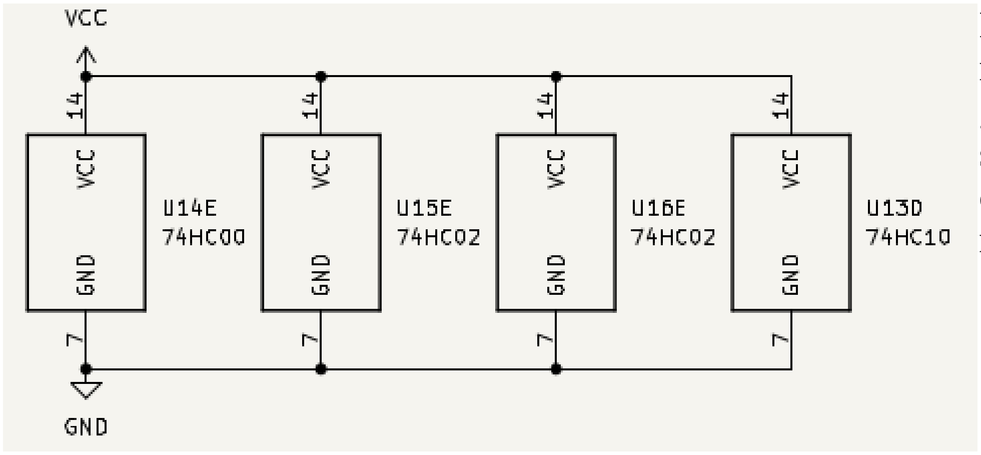
The LED needs a resistor of at least 200 ohms, depending on the strength of the LED and how much light is desired. A 1K ohm resistor has been tested and works well.

Oscillator:



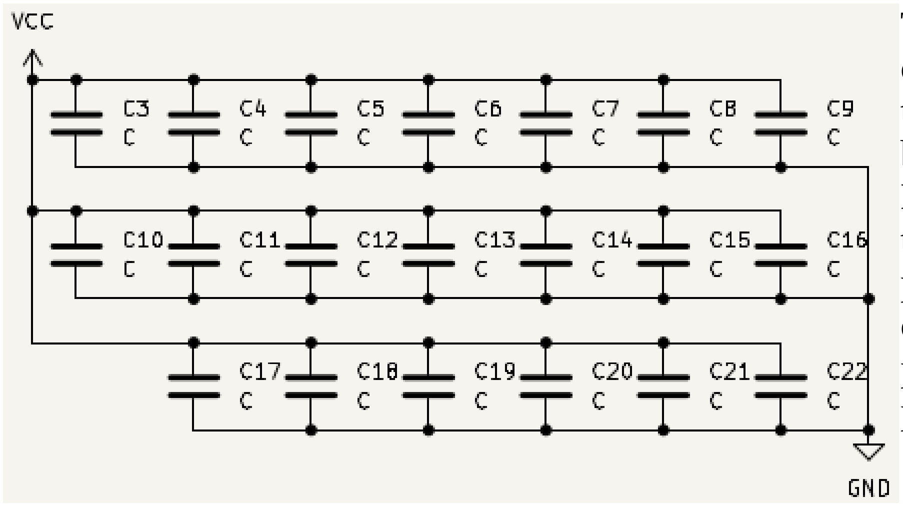
This is a 25.175 MHz oscillator, the standard frequency for VGA signals. The footprint on the printed board will allow for either DIP-8 or DIP-14 packages. A solder jumper is available for oscillators requiring their ENABLE line to be high, though is not required for other oscillators.

Glue Logic Power:



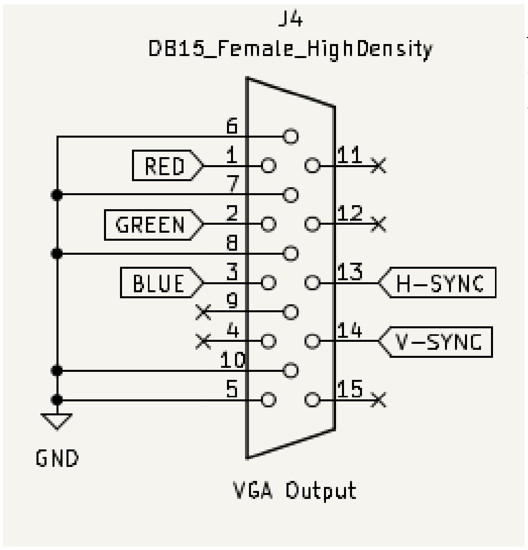
Every glue logic IC needs power and ground to run. The schematic splits these components up for readability.

Bypass/Decoupling Capacitors:



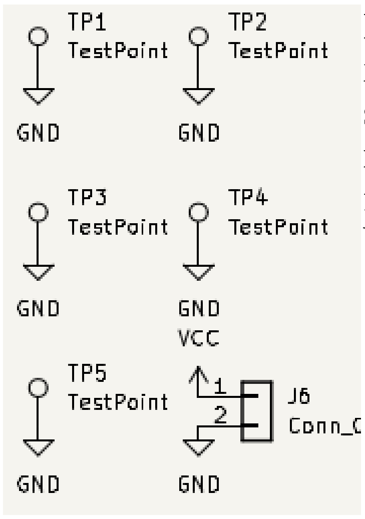
These bypass / decoupling capacitors are spread throughout the printed board, one per each IC. Each VCC is as close to the the IC’s VCC pin as possible, and each GND connects to the IC’s GND pin with as short a trace as possible.

VGA Connector:



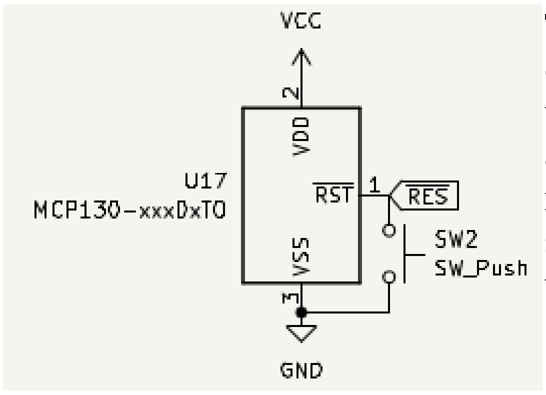
A standard 15-pin D-SUB connector is used for VGA output. Some lines must be grounded as comparators for the color signals.

Screw Holes:



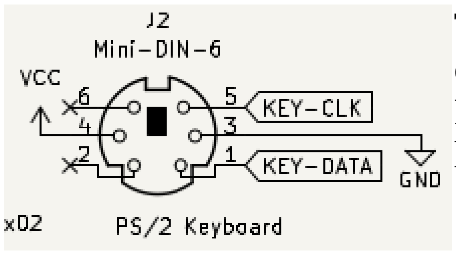
Each TestPoint is a screw hole, five in total. The printed board is a mini-ITX form factor, which has screws at 3 corners of the square board. A fourth screw hole was added for stability when not inside of a PC case. The small 1x2 connector can be used for powering additional devices, or supplying power to the board from an alternate source.

Reset:



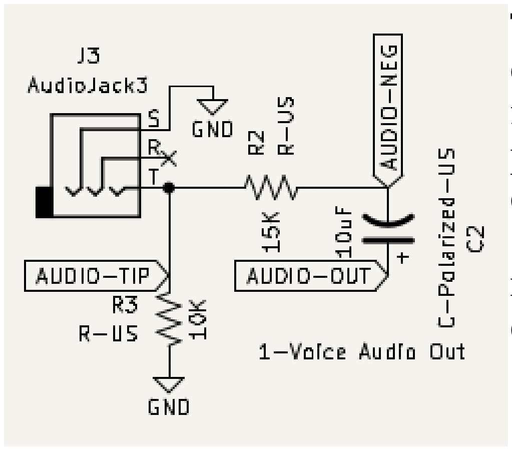
The reset IC is a MCP130-XXXD, but is also compatible with the DS1813-5. It is important to use a XXXD version, the pinout for other versions are different. This IC has an internal pull-up resistor, thus the /RES line does not need another separate pull-up resistor. The SW is a simple SPST tactile push button switch.

Keyboard:



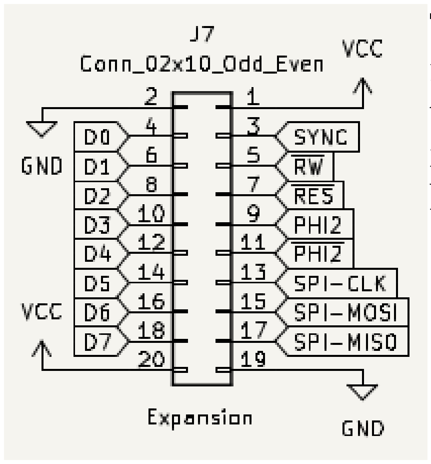
The PS/2 keyboard is connected through a Mini DIN-6 connector. Pins #2 and #6 could be used for connecting a PS/2 mouse as well, but the mouse requires bi-directional lines to operate and this system is not capable of that.

Audio:



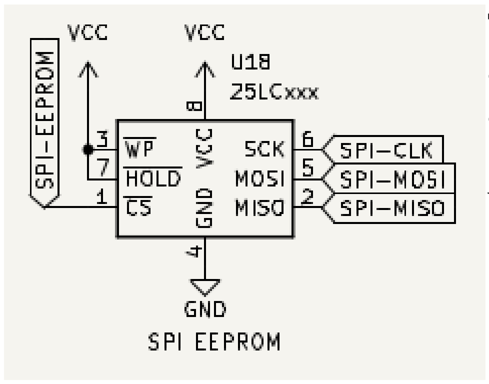
The audio circuit starts with AUDIO-OUT from the output latch. Toggling the AUDIO-OUT pin at certain frequencies from around 100 Hz to 10,000 Hz will produce sound. AUDIO-OUT goes through a 10uF capacitor to put a negative swing on the usual 0V to 5V voltage range. The 15K ohm and 10K ohm resistors drops the voltage max to around 1V with a center at 0V.

Expansion:



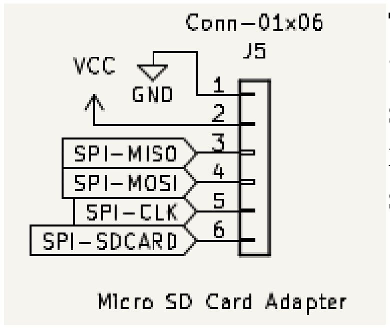
The expansion connector is designed to add more SPI functionality to the board, and/or create more output pins through the use of illegal $_3 and $_B instructions. /PHI2 is supplied directly from the 6502 processor, and is only here to help reduce glue logic on expansion boards.

EEPROM:



The EEPROM used here is a 25LCXXX, various sizes are compatible. Specific signals needed to use this IC are supplied through the datasheet. Any SPI IC with the same pinout is technically compatible, including SRAM variations.

SDcard:



This 6-pin connector is specifically designed to be used with a "SPI MicroSD Card Adapter", which is easily found when searching the internet. MicroSD cards that are around 2GB are ideal, specifically SDHC cards. Incompatible cards will simply not work.

Pricing and Sourcing:

The Acolyte Computer is designed to be to simple and low-cost yet high on features. Some IC's are unnecessary for typical operation, and many connectors and switches are optional.

The price goal for a completed board will be no more than $75 with shipping included. Prices could go even lower with better parts sourcing and buying in bulk. Currently the two suppliers are JLCPCB.com and Mouser.com

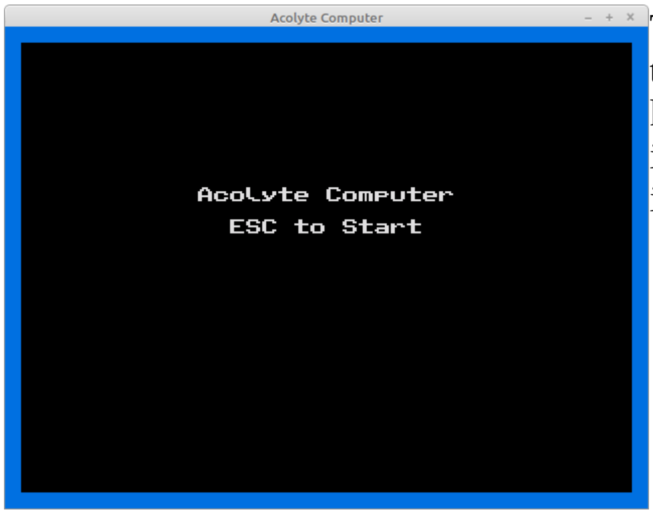
Other design decisions were to use only commonly available parts from major distributors, only through-hole components, and no programmable logic chips. This makes the board approachable to beginners and easy to understand.

Many designs on the board have cost savings in mind, such as using a single RAM and ROM chip, minimizing glue logic, and the exclusion of a 6522 VIA. Simplicity, efficiency, yet utility are the main design goals of this project.

IRQ-ISR and NMI-ISR:

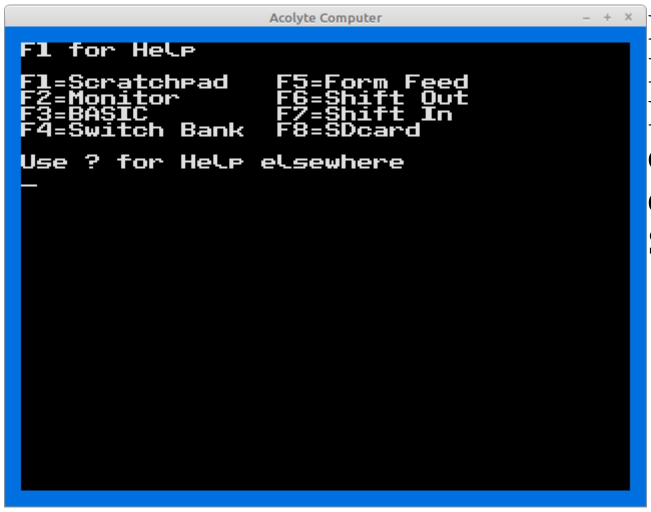
```
; 6502 running at 3.14 MHz,
; PS/2 keyboard running at 17 kHz
; That gives me 184 cycles between signals.
; /IRQ = Keyboard-Clock
; /NMI = Keyboard-Data NOR Keyboard-Clock

vector_irq                         ; 7
     PHA                           ; 3
     STZ key_bit                   ; 4
     DEC key_bit                   ; 6
     LDA #$1A ; semi-arbitrary     ; 2
vector_irq_loop                    ; (1)
     DEC A                         ; 6
     BNE vector_irq_loop           ; 2, sub-total = 100+
     LDA key_bit                   ; 4
     ROR A                         ; 2
     ROR key_data                  ; 2
     DEC key_counter               ; 6
     BEQ vector_irq_store          ; 2
     LDA key_counter               ; 4
     CMP #$FE                      ; 2
     BNE vector_irq_exit           ; 2
     LDA #$09                      ; 2
     STA key_counter               ; 4
vector_irq_exit                    ; 1
     PLA                           ; 4
     RTI                           ; 6
vector_irq_store                   ; 1
     PHX                           ; 3
     LDA key_data                  ; 4
     LDX key_write                 ; 4
     STA key_array,X               ; 5
     INC key_write                 ; 6
     PLX                           ; 4
     PLA                           ; 4
     RTI                           ; 6

vector_nmi                         ; 7
     STZ key_bit                   ; 4
     RTI                           ; 6
```

Operating System:



```
Acolyte Computer
ESC to Start
```

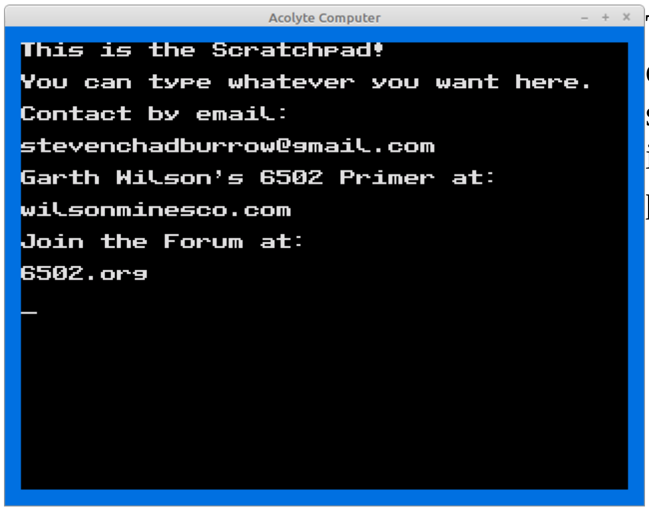
Turning the Acolyte Computer on brings up the splash screen on the video display. A keyboard must be connected to continue. It is best to connect the keyboard while power is off, then power the computer back on.



```
F1 for Help
F1=Scratchpad    F5=Form Feed
F2=Monitor       F6=Shift Out
F3=BASIC         F7=Shift In
F4=Switch Bank   F8=SDcard
Use ? for Help elsewhere
_
```

Pressing ESC will bring up the Scratchpad. It prompts with “F1 for Help”. By pressing F1 you get a list of different function key commands. This is the whole of the operating system. By default you start in the Scratchpad.

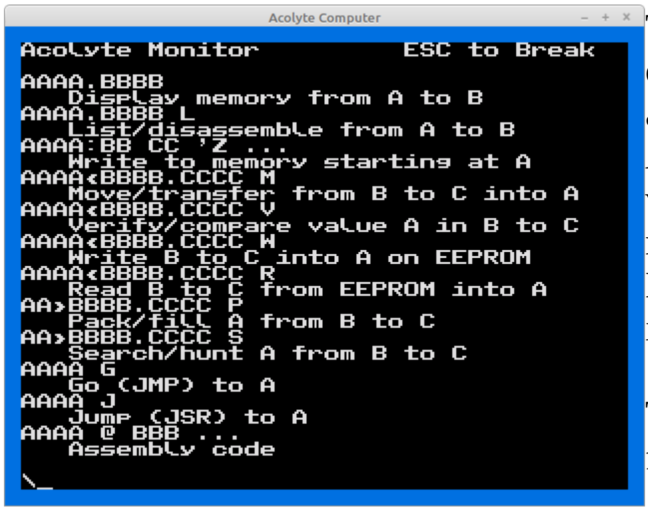
Scratchpad:



```
This is the Scratchpad!
You can type whatever you want here.
Contact by email:
stevenchadburrow@gmail.com
Garth Wilson's 6502 Primer at:
wilsonminesco.com
Join the Forum at:
6502.org
_
```

The Scratchpad is where you can simply type characters from the keyboard. It is a very simple way to demonstrate that the keyboard is functional. It can serve various educational purposes as well.

Monitor:

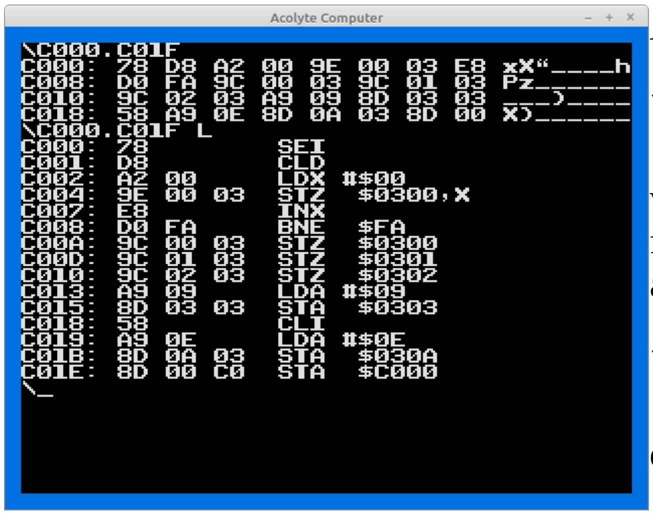


Acolyte Computer

```
Acolyte Monitor            ESC to Break

AAAA.BBBB
   Display memory from A to B
AAAA.BBBB L
   List/disassemble from A to B
AAAA:BB CC 'Z ...
   Write to memory starting at A
AAAA<BBBB.CCCC M
   Move/transfer from B to C into A
AAAA<BBBB.CCCC V
   Verify/compare value A in B to C
AAAA<BBBB.CCCC W
   Write B to C into A on EEPROM
AAAA<BBBB.CCCC R
   Read B to C from EEPROM into A
AA>BBBB.CCCC P
   Pack/fill A from B to C
AA>BBBB.CCCC S
   Search/hunt A from B to C
AAAA G
   Go (JMP) to A
AAAA J
   Jump (JSR) to A
AAAA @ BBB ...
   Assembly code

\_
```

The Monitor is where you can read and write directly to memory locations. You know you are in the Monitor when you see the \ prompt. Additional functions include list, move, verify, pack, search, and jump. From the monitor you can write and read from the EEPROM. Lastly, a mini assembler is included.

The Monitor was built to resemble WozMon from the Apple][.



Acolyte Computer

```
\C000.C01F
C000: 78 D8 A2 00 9E 00 03 E8 xX"____h
C008: D0 FA 9C 00 03 9C 01 03 Pz______
C010: 9C 02 03 A9 09 8D 03 03 ___)____
C018: 58 A9 0E 8D 0A 03 8D 00 X)______
\C000.C01F L
C000: 78          SEI
C001: D8          CLD
C002: A2 00       LDX #$00
C004: 9E 00 03    STZ  $0300,X
C007: E8          INX
C008: D0 FA       BNE  $FA
C00A: 9C 00 03    STZ  $0300
C00D: 9C 01 03    STZ  $0301
C010: 9C 02 03    STZ  $0302
C013: A9 09       LDA #$09
C015: 8D 03 03    STA  $0303
C018: 58          CLI
C019: A9 0E       LDA #$0E
C01B: 8D 0A 03    STA  $030A
C01E: 8D 00 C0    STA  $C000
\_
```

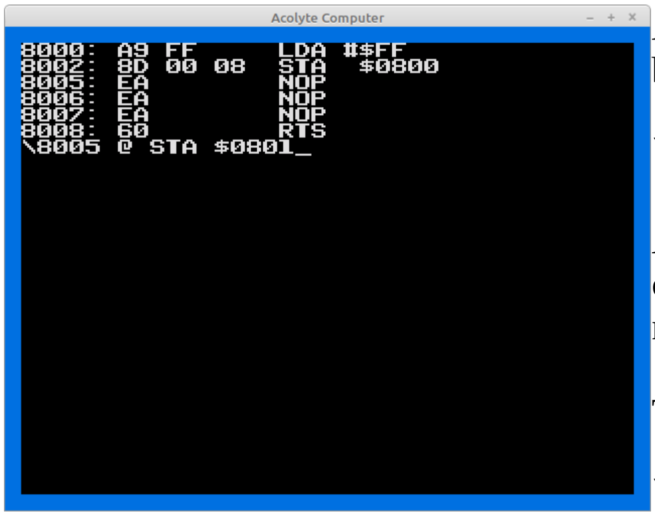
An example command would be:

```
\C000.C01F
```

which would display the memory contents from $C000 to $C01F. ASCII equivalents also are displayed. Another command:

```
\C000.C01F L
```

displays the assembly code equivalent.



Acolyte Computer

```
8000: A9 FF       LDA #$FF
8002: 8D 00 08    STA  $0800
8005: EA          NOP
8006: EA          NOP
8007: EA          NOP
8008: 60          RTS
\8005 @ STA $0801_
```

An example of an assembly program would be like this. The first command was:

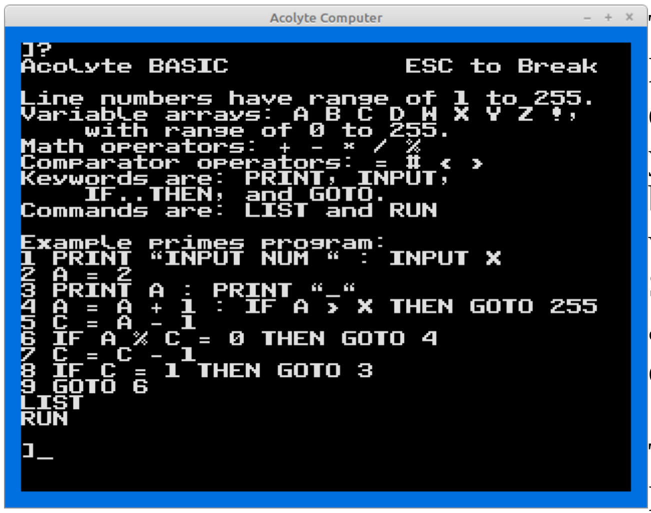
```
\8000 @ LDA #$FF
```

After hitting ENTER the command line changes to the interpreted equivalent and the next instruction can be entered immediately.

To run this simple program, use:

```
\8000 J
```

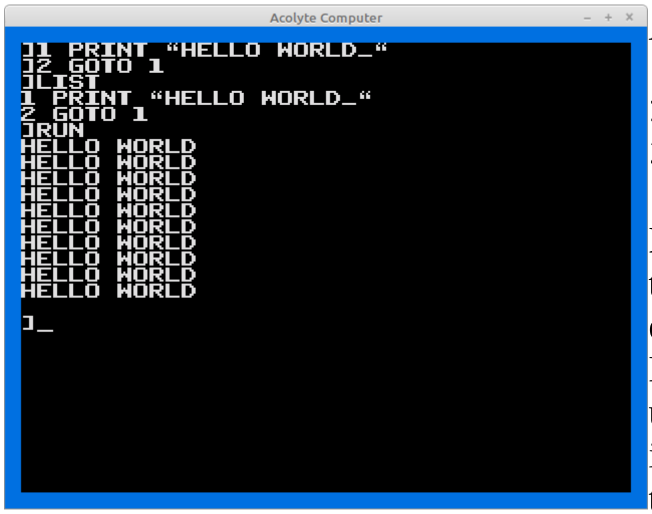
BASIC:



```
]?
Acolyte BASIC               ESC to Break

Line numbers have range of 1 to 255.
Variable arrays: A B C D W X Y Z !,
    with range of 0 to 255.
Math operators: + - * / %
Comparator operators: = # < >
Keywords are: PRINT, INPUT,
    IF..THEN, and GOTO.
Commands are: LIST and RUN

Example primes program:
1 PRINT “INPUT NUM “ : INPUT X
2 A = 2
3 PRINT A : PRINT “_“
4 A = A + 1 : IF A > X THEN GOTO 255
5 C = A - 1
6 IF A % C = 0 THEN GOTO 4
7 C = C - 1
8 IF C = 1 THEN GOTO 3
9 GOTO 6
LIST
RUN

]_
```

The Acolyte Computer comes with a 4K BASIC, which is very simple and a little quirky. You know you are in BASIC when you see the] prompt. It only has 8 variables, but each variable is an array of 256 bytes which can be accessed using parenthesis. Simple math and comparator operators are available, and a pseudo-random number is created using the exclamation mark.

This BASIC was build to resemble MINOL BASIC.

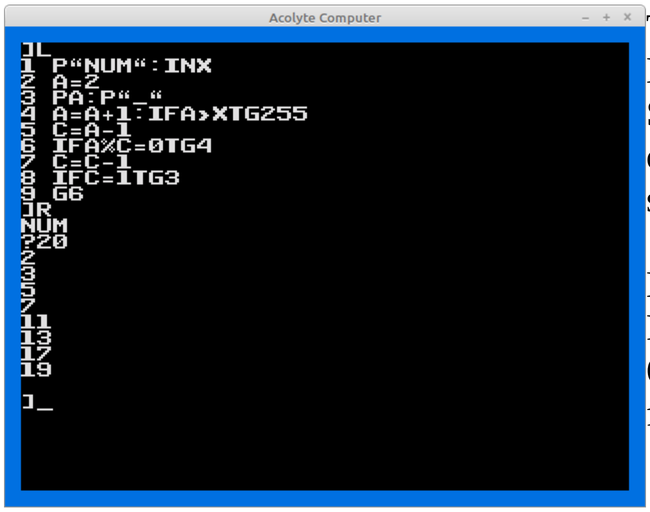


```
]1 PRINT “HELLO WORLD_“
]2 GOTO 1
]LIST
1 PRINT “HELLO WORLD_“
2 GOTO 1
]RUN
HELLO WORLD
HELLO WORLD
HELLO WORLD
HELLO WORLD
HELLO WORLD
HELLO WORLD
HELLO WORLD
HELLO WORLD
HELLO WORLD
HELLO WORLD

]_
```

A very simple test program would be:

```
1 PRINT “HELLO WORLD_”
2 GOTO 1
```

Line numbers are from 1 to 255, and having them closer together makes for faster execution. The underscore prints a newline. Notice the LIST and RUN commands do not use line numbers. Pressing ESC while code is excuting will break the program back into the prompt.

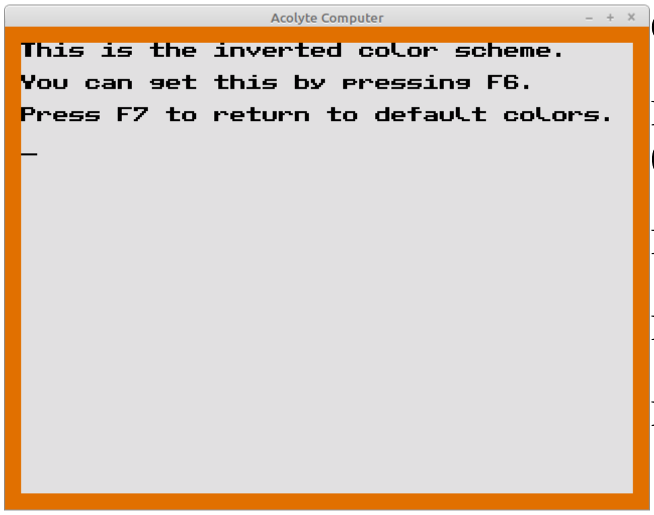


```
]L
1 P“NUM“:INX
2 A=2
3 PA:P“_“
4 A=A+1:IFA>XTG255
5 C=A-1
6 IFA%C=0TG4
7 C=C-1
8 IFC=1TG3
9 G6
]R
NUM
?20
2
3
5
7
11
13
17
19

]_
```

This BASIC also allows for shortened names. Note the brevity of the program code here. Shortening code text helps increase code compactness, as there is only 12KB of code space available in RAM.

Lastly, BASIC will only run with extended RAM installed in the computer. Using the 62256 32KB SRAM will disable all BASIC functions.

Other Features:



```
This is the inverted color scheme.
You can get this by pressing F6.
Press F7 to return to default colors.
_
```

Other features available at any time are:

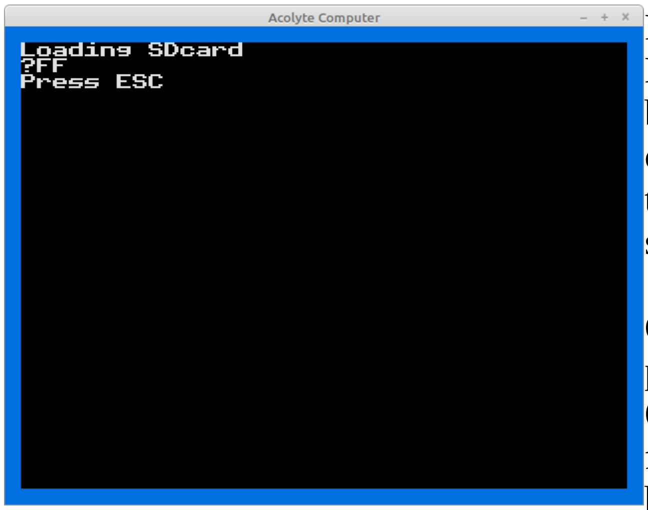
F4 = Switch Bank, starts second ROM bank
(This might cause problems.)

F5 = Form Feed, clears the screen

F6 = Shift Out, uses inverted colors

F7 = Shift In, uses default colors

SDcard:



```
Loading SDcard
?FF
Press ESC
```

Pressing F8 brings up the SDcard screen. Here the program is reading the first 1K of bytes from the SDcard that is plugged into the computer. If there is no SDcard present or if that particular SDcard is not supported, it will show an error.

Otherwise, data from the SDcard will populate the $0400 to $07FF range and the 6502 processor will start executing that code from the location $0400. This is intended to be a type of bootstrap program to load more data and code from the SDcard later. To exit any glitched program, simply press the RESET button on the computer board itself.

Contacts:

For questions and/or comments, contact Steven “Chad” Burrow at:

stevenchadburrow@gmail.com

Garth Wilson’s 6502 Primer at:

http://wilsonminesco.com/6502primer/

Join the 6502 Forum at:

http://forum.6502.org/