Fennec 6800 Video Terminal Interface

A Motorola 6845 Based Composite Video and ASCII Keyboard Terminal Interface for the Fennec 6800

Designed By: Chartreuse
2023

Dedicated to the Retro Dreams Discord Server for all the help and discussions over the years.

Introduction

The Fennec 6800 VTI is a simple composite video based terminal interface with parallel ascii keyboard input for the Fennec 6800 single board computer. Video timing is generated by a Motorola 6845 CRT controller, allowing for flexible timings and customization by the end user. 2kB of display memory is provided as well as a 2kB to 8kB font ROM (2716/2732/2764) allowing for an ASCII text display, and optionally pseudo graphics with an appropriate ROM. Input is handled by a parallel ASCII keyboard connection on either a DA-15 connector or a DIP-14 socket. Eight data inputs and a strobe line from the keyboard are passed to a 6820/6821/6520/6521 PIA chip for handling of the keyboard. An option ROM on the board hooks into the primary monitor ROM and redirects the primary I/O from the first serial port to the VTI interface instead.

Specifications

- MC6845 CRT Controller (CRTC) for flexible display timing and control
- 2kB of video character memory (6116 SRAM)
- 2-8kB Font ROM allowing for a variety of font heights.
- Option ROM provides a local console for the Fennec, negating need for an external terminal
- DA-15 or DIP-14 parallel ASCII keyboard interface provided through 6820/6821/6520/6521 PIA
- Prototyping area on PCB for simple expansions and projects.
- Second un-used port on PIA broken out to headers for prototyping and interfacing.
- Jumperable inverse video allows for both white on black and black on white.
- RCA composite video output. (“NTSC” timing by default)
- DE-9 TTL Video Output (monochrome CGA by default)
- 12MHz dot clock.
- 64x24 Text Display with default dot clock, controllable with CRTC registers.
- Beautiful blue PCB with exposed traces for a refined look.
- Compact Board design, only 8.6×5.2 inches!

The Fennec VTI provides a flexible platform for text-based video generation on the Fennec as well as other systems with a similar bus interface providing a ~4kB memory window for interfacing. The low 2kB is directly mapped to the video memory, with the upper 16 bytes mapping to the 6845 registers. The upper 2kB is mapped to the option rom, with the upper 16 bytes mapping to the PIA registers.

The parallel ASCII keyboard interface is provided on both a DA-15 interface along with a DIP-14 header. The interface consists of 8 data lines, 1 strobe line, and 5v power and ground. Some vintage keyboards may require different voltages in which case an external adapter will be required. The keyboard interfaces directly to the PIA interface, with the data lines going to the A port, and the strobe line to CA1. Due to the flexibility of the PIA this allows the polarity of the strobe to be configured in software, as well as allowing for interrupt driven capabilities on the input in addition to polling.

The PIA also provides a 14 pin prototyping header with the B data port, as well as CA2, CB1, CB2, 5v and ground. This allows for quick and easy prototyping of small circuits to be controlled directly by the

Fennec computer. Coupled with the onboard prototyping area, small circuits may be built up semi-permanently into the computer itself.
Two jumpers are also provided which connect VSYNC and BLANK to CB1 and CA2 respectively, allowing for their status to be either queried by the program, or used as an interrupt source. These signals can be used for either simple timing with the VSYNC pin as a 60/50Hz interrupt source, or for advanced CRTC tricks such as only writing to display memory during BLANK to provide snow free operation.

Design Goals

The VTI was designed to give the Fennec it's own built in I/O system. I find that a computer doesn't really feel complete by itself without a way of hooking up a display and keyboard directly to it; more-so in terms of homebrew machines where serial console based systems are a dime-a-dozen. The VTI board is intended to fill this gap by providing both a composite video text terminal as well as a keyboard interface. Part of this goal was also in demystifying the 6845 CRTC for myself and learning to interface one to a microprocessor. The VTI originally started as a breadboard project to just get a video circuit working with an interface to the processor. The Fennec's simple bus allowed for a quick and easy setup and interfacing for the design.

One consideration when using the 6845 and similar CRTC based circuits is how to handle the situation where both the video circuit and the microprocessor both want to access the display memory. There are many approaches here such as synchronizing the processors clock rate to that of the video circuit to allow for seamless access; accessing the video memory only during blanking intervals where the video circuit is not using/caring about the video on screen; adding wait states to the processor till the video controller is no longer accessing the memory; or the simplest option that the CPU always has priority. The simplest option is what I decided to go with in this case.

Now the upside to this CPU always wins option is simplicity, all we need to control the VRAM access is a set of multiplexer chips and data bus buffers. The multiplexer is on the address lines to the VRAM, and control is simply controlled by the CPU selecting the VRAM memory space. The big downside to this approach is what happens during these conflicts? Well "snow" is what happens. Snow is the term used for the appearance of small glitches on the display when the CPU access the VRAM while the CRTC is also trying to display pixel data. Instead of the intended character data, the video circuit instead picks up the state of the data bus from the CPU and displays that instead. This results in an 8x1 pixel section of the video containing garbage for a single frame for each VRAM access. This type of access was fairly common on some early video cards, most notably the IBM CGA card when in 80 column mode. It is an unwanted side effect, but not terribly distracting when VRAM is only updated periodically. Since the VTI card is intended primarily as a text-terminal, this distraction would be minimal and is made up for in the design simplicity.

One thing that can be done in the future with this simple interface is to convert it to the second type. That is only doing VRAM access during blanking. When I created the PCB version of the VTI I thought to add a jumper to allow for the BLANK signal to be hooked up to the PIA providing the keyboard access. Using this the ROM can be upgraded to wait until BLANK is asserted before doing any VRAM accesses. This would slow down the characters per second that the display is capable of but would completely eliminate all snow from the display with only a software and jumper upgrade.

The keyboard interface was a later addition to the project and not present on the original prototyped breadboard circuit. I decided to go with a parallel ASCII keyboard interface to fit with the late 70s type feel and goal of the Fennec, as such keyboards were ubiquitous in systems of the era. In fact, the pinout used on the DIP-14 keyboard connector is taken from a 1976 Polymorphic Systems VTI card for S100 systems that I have in my collection; the card also being one of my primary inspirations for the project. While a PS/2 keyboard interface would be more convenient for users it broke the immersion of the system too much. To compensate though, and because I do not own any parallel ASCII keyboards

personally, I did make a side project PCB based off of https://www.n4vlf.net/ps2.html to convert a PS/2 keyboard to the parallel ASCII required.
For interfacing with the keyboard, while I could simply use an 8-bit latch such as the 74LS574 or a period parallel interface like the 8212. I decided to stick with Motorola parts to fit the design of the system. To this end I selected the 6820/6821 PIA chip. This chip provides for two 8-bit general purpose I/O ports along with 4 additional I/O pins that can be configured to fire interrupts among other uses. With this chip I can simply hook up the 8 data lines from the keyboard to one of the I/O ports, and hook up the strobe into to one of those additional I/O lines. Then the setup can be used in either a polled or interrupt driven mode to get the characters from the keyboard. Another benefit of using PIA chip is the ability of the strobe polarity to be chosen and edited in software to adjust for different keyboards, along with translating key codes if needed.

As the PIA provides for two I/O ports and I did not require the second port for the terminal interface, I brought out the port to a pin header on the PCB, allowing for it to be used by the end-user for interfacing with whatever devices and projects they see fit. As well with the extra space on the board a 0.1” pad per hole prototyping area was also left for this purpose.

While not implemented on the board a feature that I believe I should have added was for jumpers to hook up the high address lines on 2732 and 2764 EPROMS to pins on the PIA instead of to additional row address lines on the CRTC. This would allow for software to change between different fonts or even pseudo-graphics on the font rom for display.

Theory of Operation

Video Display

The heart of the video timing is the Motorola 6845 CRTC. This integrated circuit is by and large a glorified counter chip, capable of counting up characters in the horizontal direction, and characters made up of a number of rows in the vertical direction. With this counting it divides a line into a blanking region, a sync pulse in the middle of that blanking region, and a visible region. Likewise in the vertical direction it has a blanking region, a vertical sync region in the middle of the blanking, and a visible region. With just these attributes we get a vertical sync signal, a horizontal sync signal, and a blanking signal. These alone are enough to get a display to synchronize but we still need more in order to display information on the screen. That's where the address generator comes into play. For each character cell on the screen the CRTC generates an address for us, starting from the top left of the screen, working it's way to the right, then going down one cell and continuing. The power of the CRTC here is that it's able to do these for arbitrary row widths, as opposed to discrete circuits that are typically limited to powers of 2 for simplicity sake. As characters on the display are normally composed of a number of rows, for example 8, the CRTC repeats the character address for that many rows down the screen. To differentiate the rows and move to different rows of the font the CRTC also provides a row address output. This output simply counts the current scanline within the character row. Now the 6845 CRTC is more powerful than this and offers some additional features, but this is enough to explain the heart and building blocks of our video generation circuit.

We'll start with the visible portion of the screen. For any given time in the visible area the CRTC is outputting a character address on the MA0-MA13 bus. For the Fennec VTI the display memory is 2kB provided by a 6116 SRAM chip thus we need just 11 address lines MA0 to MA10, the upper 3 address lines are left disconnected. These address lines are connected to the SRAM chip through a multiplexer which we will discuss later, but for now just imagine them directly connected to the SRAM chip.

The Fennec VTI video circuit is designed as character based display dividing the screen up into a region of 64x24 characters, and the CRTC chip is designed especially to make this kind of setup easy. The character addresses will be appropriately repeated on subsequent scan-lines with the row address incremented until the number of rows in a character is reached. What all this means for us is that in the SRAM one byte corresponds uniquely to each character cell on the screen in row-column order. The output of the RAM is then latched into a 74LS574 octal d-flip-flop. This flip flop acts to keep the output stable to the font ROM. While likely strictly not needed it should help to slightly reduce the amount of snow on the screen. From this latch the value is then connected to the A3-A10 address inputs to the 2716 font ROM.

At the font ROM we need to combine the the character value we get from the RAM with the row address from the CRTC to select the line in the font that corresponds to the current line on the display. The VTI is primarily designed for use with an 8x8 font, meaning we need 8 rows or 3 bits to uniquely select a row of the font. This row address on RA0-2 from the CRTC is directly connected to the A0-A2 inputs on the font ROM. With a 2716 EPROM this combination of 8+3 bits fully utilizes the ROM. If a

larger ROM such as a 2732 or 2764 are used then more rows can be added to the font. To allow the use of this a jumper is provided to connect RA3 to A11 on a 2732. And if the 2764 footprint is used RA4 will also be connected to A12. Keep in mind that on a standard NTSC television we only have 262 scanlines to work with without going into interlaced mode, with only around 240 visible at most, which will limit the useful font heights without compromising with fewer rows of text.

The output of the font ROM is then latched into a 74LS166 8-bit parallel to serial shift register. This shift register is then clocked at the dot clock (12MHz) to output the pixel data serially to the display.

Now the complex part here is latching the pixel data at the right point. If we tried to do it right away as the CRTC changes character address we'd be latching the previous character value's data from the ROM. This is because every step of the circuit has its own delay. The RAM has an access time on the order of 100ns (though in my build my D4016 SRAM is a fast 50ns model). The ROM has an access time on the order of 150ns for the slowest ROMs. And the latch between them is also clocked at a certain point, but also has a small delay on the order of 10ns or so. Now ~300ns doesn't sound like a lot of time, but at television frequencies and with a dot clock of 12MHz, that 300ns corresponds to 3.6 pixel times. To account for variance and provide a stable synchronous setup I decided that the latch between the RAM and ROM will latch at the same time as the shift register will latch the output of the font ROM. This setup means that there will be a 1 character delay or pipeline from the address changing and the shift register getting that character data. This delay we will account for in the software timing and adjusting the blanking interval, but will provide a rock solid setup without having to worry about variable asynchronous delays.

To time this latch we need to go to the dot clock and how the CRTC is clocked. The 6845 CRTC does not take a dot clock as input but takes in a character clock; remember the CRTC is a character display device. Since our characters are 8 pixels wide (corresponding conveniently to one byte worth of pixels) we need to divide our dot clock by 8 to get our character clock. A 74LS191 4-bit universal bi-directional pre-loadable counter is used for this purpose. To generate the dot clock we make use of the 3rd bit of the counter to act as a divide by 8 of the dot clock. For timing reasons we will get back to, this character clock is actually inverted before being input into the CRTC.

The counter is pre-loaded with the value 8 (1000 in binary), and is reset to this value every time it overflows. We do this as we only want to use the low 3 bits of the counter, but we also want to make use of the single clock pulse long carry output that the counter provides. This output is normally used for creating a ripple-carry system of multiple counter chips, but we wish to use it to act as our latch signal for the flip flop and shift register. This output is also hooked back up to the parallel load input which will reset the counter to 8 on the next clock cycle, effectively turning the counter into a 3-bit counter. The output is active low and in the schematic is called the SH/LD signal. When high the shift register will be shifting out the data currently in it to the display, and for the single dot clock that it is low the shift register will latch in the new data from the font ROM.

...To be completed...

Assembly

Assembly is rather straight forward but should be accompanied by the schematic or bill of materials to know where to place the various resistors. I've corrected this for my future boards, but this board only has the reference designators and not the component values marked.

Like other throughole kits I would recommend starting from the shortest components to the tallest components, and I would personally recommend sockets for the ICs, especially the vintage ones, though they are not required. My personal assembly order would be to do the IC sockets first, even if they're taller than the resistors, but either approach has it's merits. For the tantalum capacitors to the side of the RAM, and the two electrolytic capacitors, make sure to get their polarity correct. Both have their positive side marked on the board, for tantalum capacitors their **positive** lead is normally marked with a + sign or stripe on the capacitor, and for electrolytics their **negative** side is normally marked with a stripe. If cost is a factor, feel free to omit some of the tantalum capacitors as the number is likely overkill as I miscalculated the current dram of the SRAM chips (mainly due to only a single pair being active at any one time).

Assembly

Assembly is rather straight forward but should be accompanied by the schematic or bill of materials to be safe. All component and IC values are marked on the board itself to facilitate painless assembly. For the dual-size ROM sockets align the socket to the appropriate pin 1 mark depending if using a 28 pin socket for 2764 EPROMs or 24 pin socket for 2716 EPROMs. The inner marking is for 24 pin ROMs. If installing a 24 pin ROM into a 28 pin socket, take care to seat it all the way to the right, leaving the 4 extra socket holes on the pin 1 side.

Like other throughole kits I would recommend starting from the shortest components to the tallest components, and I would personally recommend sockets for the ICs, especially the vintage ones, though they are not required. My personal assembly order would be to do the IC sockets first, even if they're taller than the resistors, but either approach has it's merits. For the two electrolytic capacitors, make sure to get their polarity correct. Both have their positive side marked on the board; for electrolytics their **negative** side is normally marked with a stripe.

If you are using a 74LS86 IC then use the resistor values printed in the footprints, 220 and 470 for R4 and R5 respectively. If you are substituting in a 74HCT86 then use the values written above, substituting the 220 resistor for a 470, and the 470 resistor for a 1k.

Unless you've modified the software to work in another slot, place a jumper shorting cap on the left-most (pin 1) pair of pins on J2 to select the EXP6 slot for the board.

As well, place a shorting jumper on pins 1-2 (right most) of J2, connecting the side labelled NORM to the center pin. This configuration will give white text on a black background, use the other position for black text on a white background if desired.

Program the Option ROM with the latest version of *vtirom.bin* This ROM will provide the routines for the display as well as hook into the main ROM to provide the main monitor console over the VTI instead of the first serial port.

Program the Font ROM with an 8x8 bitmap font of your choice, I do not currently provide one but I recommend the IBM CGA font ROM or similar as a good base choice. The font is stored as 1 byte per row, and top to bottom row order for each ASCII character in order, with the high bit of each byte being the left most pixel of the character.

Required Bodges (REV.0 ONLY)

I highly recommend upgrading to a rev 1 board if possible however a rev 0 board can be brought up to a working state with a few bodges. For wiring I recommend 26 or 30 AWG wire wrap wire (0.4 to 0.25mm wire) for doing the neatest bodges. Wire-wrap wire has thin insulation of Kynar (PVDF) or sometimes Teflon (PTFE) and is silver plated copper wire, and is ideal for bodges like these.

Inverting 74LS157 Select Line

Due to a mix-up in the wiring the select line for the 74LS157 mutliplexers is the opposite polarity to expected. This is fixed in rev 1, but in rev 0 boards requires the addition of a 74LS04 hex-inverter to the prototyping area. Install a 74LS04 inverter into the bottom left most area of the prototyping area of the card. Hook up VCC to pin 14, and GND to pin 7. Find the trace that runs under all 3 74LS157 multiplexers, and is between the power and ground lines running horizontally. Cut this trace to the right (if looking from the bottom) of all 3 multiplexers. This should be near where the two power traces end.

Solder a wire from the right of this cut to pin 1 on the added inverter. And solder a wire from pin 2 of the inverter to anywhere on this trace to the left of the cut.

Inverted 74LS245 R/W Line

We will be using the same 74LS04 inverted added by the bodge above, so complete that first. On the top side of the board locate U6, the 74LS245. Cut the left most trace below in to the right of the ceramic capacitor, this should be the small trace that runs under the ROM below the IC. Flip the board over. Run a wire from pin 1 of the 74LS245 to pin 4 of the added inverter. Run a wire from pin 3 of the inverter to pin 34 of the expansion connector (if looking from the back with the dsub connectors on the right this is the 4th pin in on the top row).

INV Display Mixup

Directly above the 3 pin header for INV and NOR display. Cut the trace going to the left-most (INV) position of the header below the IC. Flip the board over. Solder a short wire between the INV pin and pin 5 of the 74LS139 U9.

Flip the board right side up again. To the right of the NOR text, cut the pair of traces leading downward before the via. This should be approximately level with the R in NOR. Using a pair of short bodge wires, cross these traces from one side to the other. That is, on the left trace, above the cut run a wire to the right trace below the cut, and for the right trace, above the cut run a wire to the left trace below the cut.

Flipped Composite Jack

The composite video RCA jack has been wired with the signal on the outer shell, and ground on the center pin. You can chose to make an adapter cable which flips this, or perform the following bodges to repair it correctly. On the top side of the board, cut the thick ground trace below the RCA jack between the jack and the DE-9 connector. On the top side of the RCA jack cut **both** of the thick traces leading from the jack below resistor R5. Flip the board over. Run a wire from the large ground trace below the jack to the single pin on the board edge side of the RCA jack. Cut the trace from the pin 1 side of the ceramic capacitor to the left of the RCA jack leading to the two pin side of the RCA jack. Run a wire from the same ground as before to the pin 1 side of that ceramic capacitor. Run another wire from that capacitor up to the ground lead of capacitor C2. Finally run a wire from the two pin side of the RCA jack (tip) to the board edge side of resistor R5 (directly above the RCA jack).

Bill of Materials

Ceramic Capacitors

- 17x 0.1uF 50v X7R with 0.1” (2.54mm) lead spacing

Aluminum Electrolytic Capacitors

- 1x 470uF 16v with 3.5mm lead spacing, max diameter 10mm.
- 1x 47uF 16v with 2mm lead spacing, 5mm diameter

Sockets (Optional)

- 4x DIP-14
- 6x DIP -16
- 1x DIP-14 *or* DIP-8 crystal socket (only 4 corner pins fitted) for main oscillator
- 2x DIP-20 (0.3” wide)
- 1x DIP-24 (0.6” wide) socket for 6116 SRAM
- 2x DIP-24 (0.6” wide) sockets for ROMs if using 2716/2732
- 2x DIP-28 (0.6” wide) sockets for ROMs if using 2764
- 2x DIP-40 (0.6” wide)

Connectors

- D-sub DA-15 Right-Angle Female with 7.70mm pin edge offset, 9.12mm mounting hole offset
- D-sub DE-9 Right-Angle Female with 7.70mm pin edge offset, 9.12mm mounting hole offset
- RCA Female Jack, Right-Angle
- 1x 2x20 0.1” male pin header
- 1x 2x6 0.1” male pin header
- 3x 1x2 0.1” male pin header
- 1x 1x14 0.1” male pin header
- 1x 1x3 0.1” male pin header
- *(I recommend getting 1x40 and 2x40 breakaway headers and cutting them down)*
- 2x 0.1” shorting jumper (+1 if using 2732 or larger font ROM)

Resistors

- 1x 10k 1/4W Carbon Film

- 2x 220 1/4W Carbon Film
- 1x 4.7k 1/4W Carbon Film
- 1x 470 1/4W Carbon Film
- *If using a 74HCT86 or similar then add 1x 1k 1/4W Carbon Film, and leave out one of the 220 ohm resistors*

Diodes

- 1x 1n4148 or similar small signal high speed diode
- 1x 5mm LED. Colour of your own preferenceor but a deep red low brightness LED recommended.

Transistors

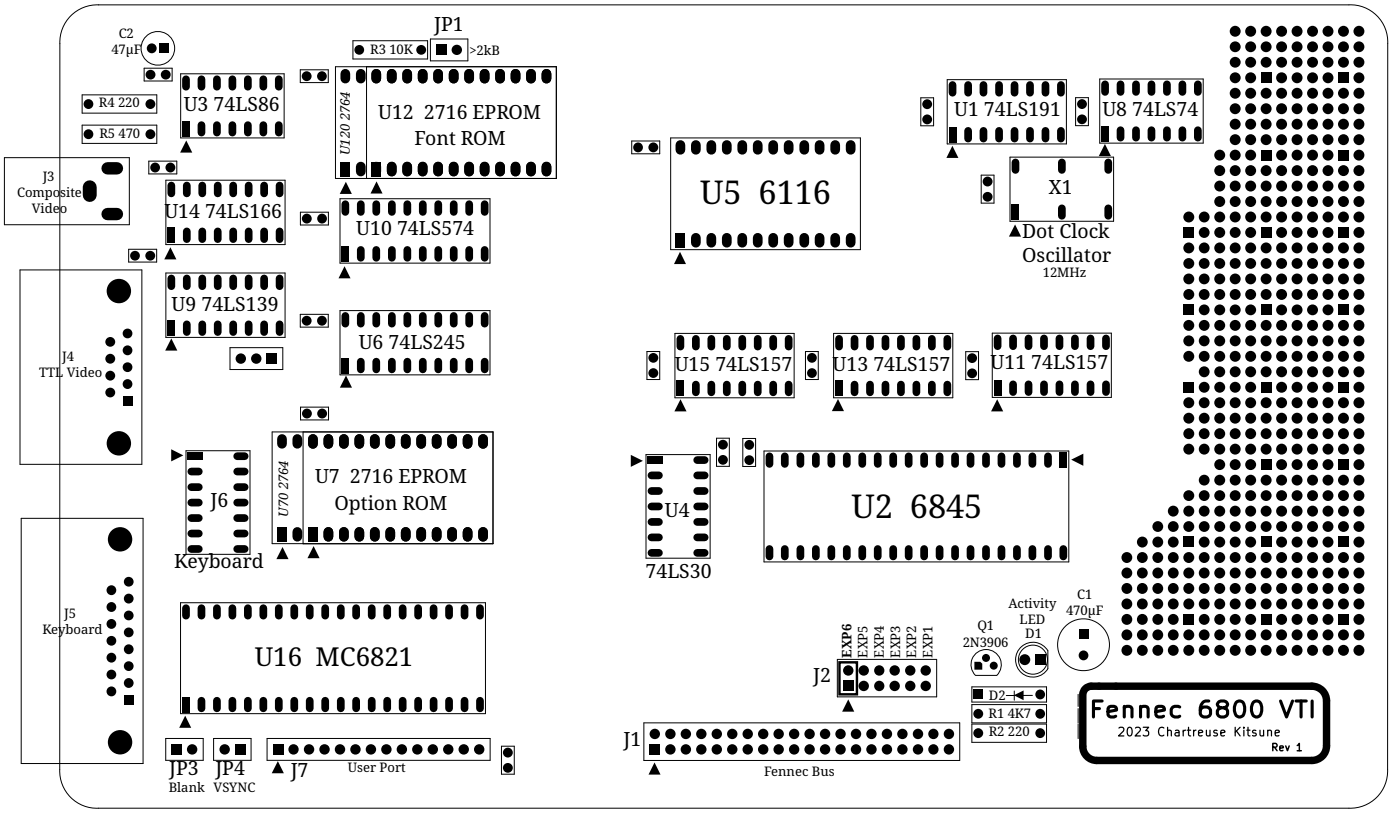
- 1x 2n3906 PNP Transistor in TO-92 package with staggered center pin

Integrated Circuits

- 1x Motorola MC6845 CRTC, any speed grade should be fine
- 1x 12MHz DIP-14 (full can) or DIP-8 (half can) oscillator
- 1x Motorola MC6820/6821 or MOS 6520/6521 or compatible PIA
- 2x 2716 EPROM or 28C16 EEPROM or compatible. 2764 may also be used (larger will work but only first 2kB of option ROM and 8kB of font ROM usable)
- 1x 6116 or compatible (eg. D4016) 2k x 8 SRAM
- 1x 74LS86 quad XOR
- 1x 74LS166 8-bit parallel to serial shift register
- 1x 74LS139 dual 2-4 selector
- 1x 74LS245 octal bus transciever
- 1x 74LS574 octal d-flip-flop
- 1x 74LS30 8 input NAND gate
- 3x 74LS157 2:1 multiplexers
- 1x 74LS191 4-bit universal counter
- 1x 74LS74 dual D-flip-flop

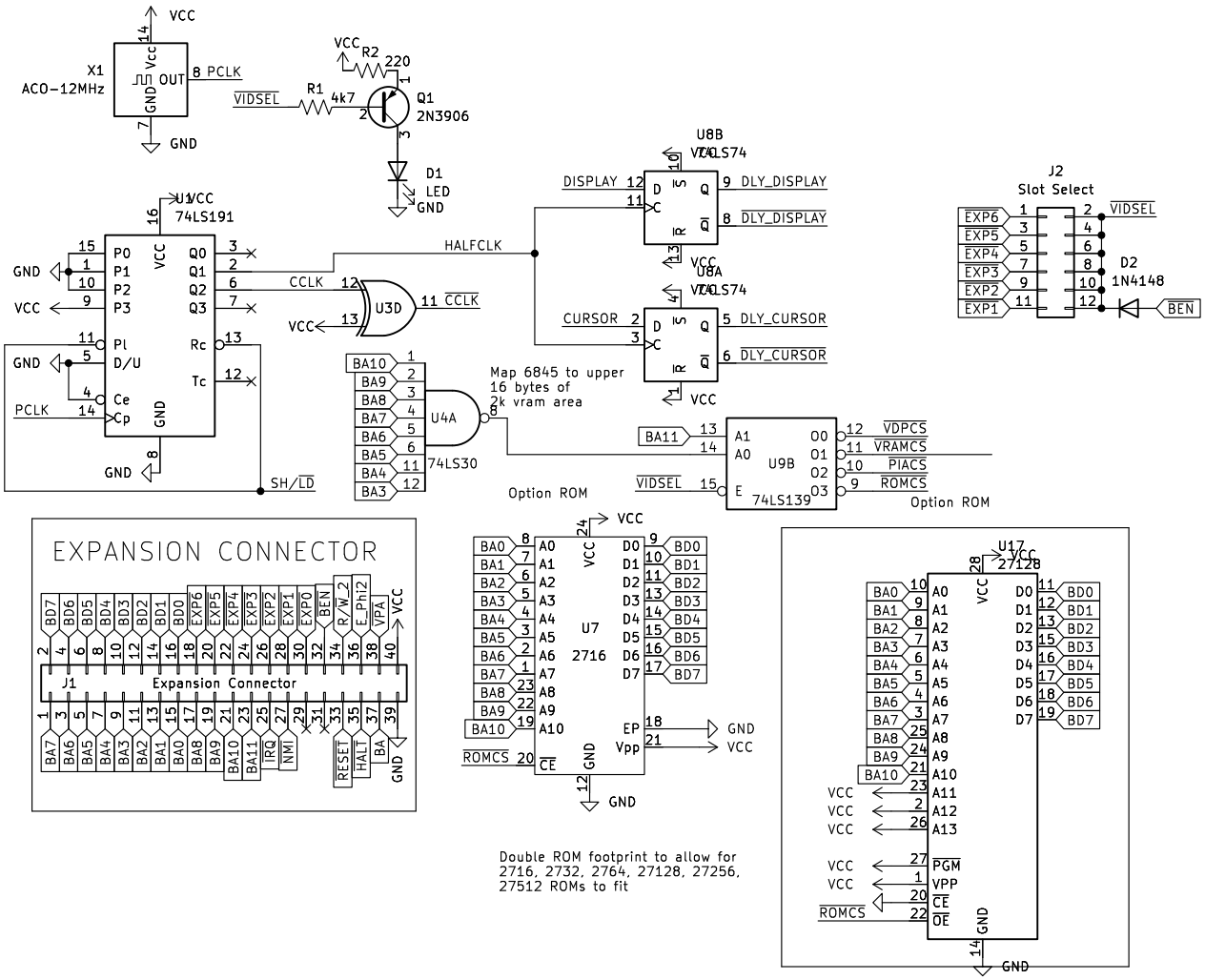
BOM Table

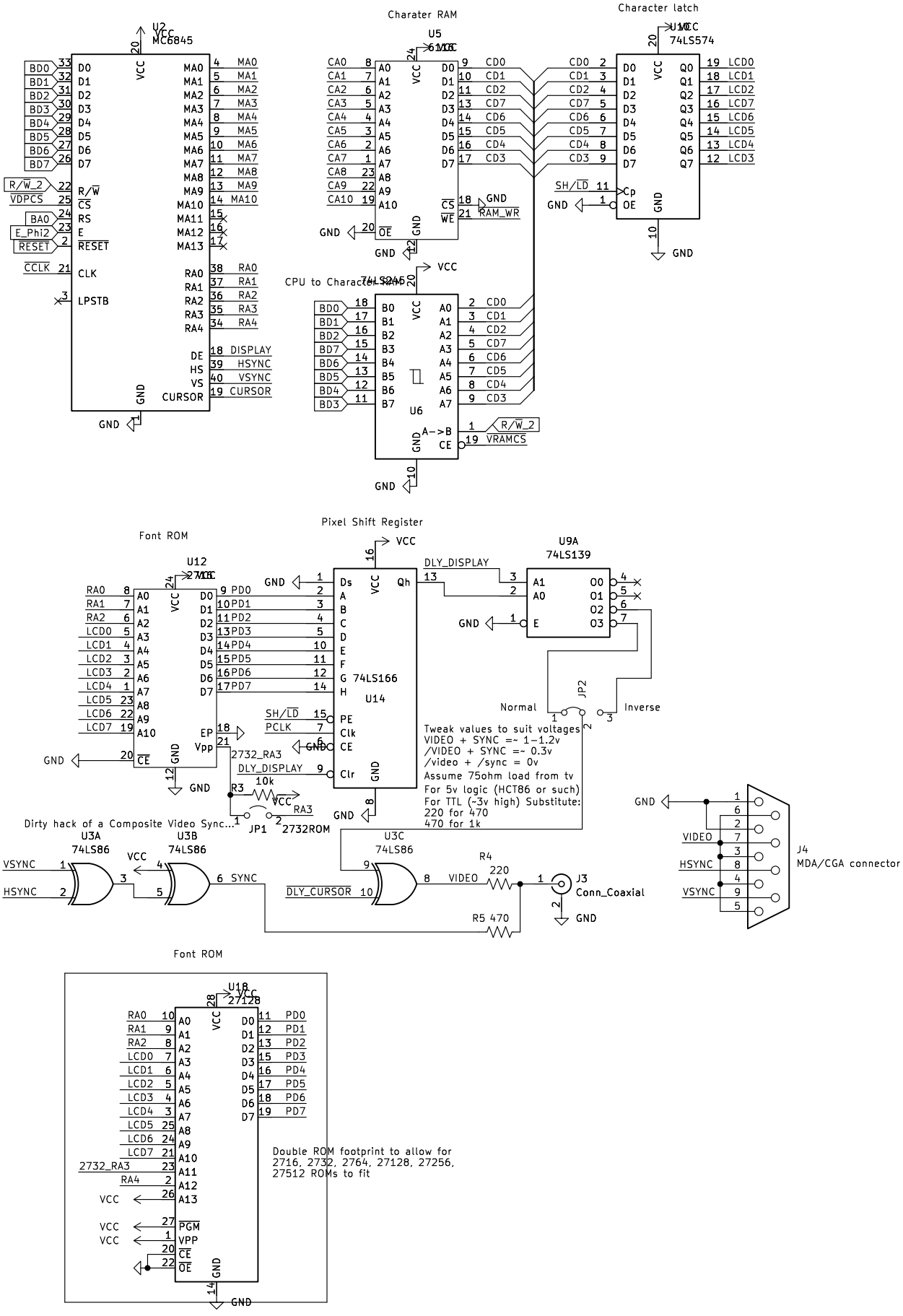
IC Layout Diagram

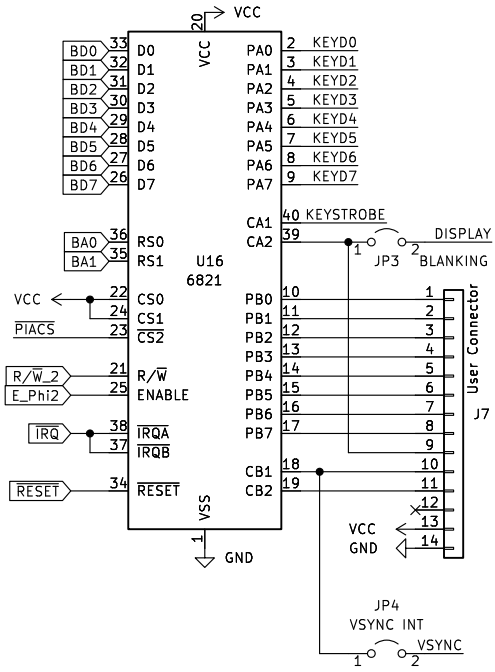


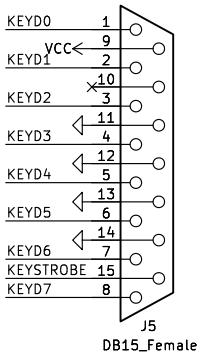
Schematic

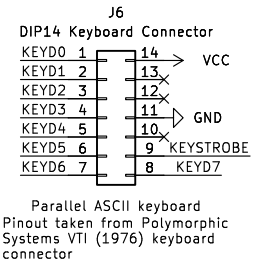
For the most up to date schematics, check out the KiCAD project in the repository. These schematics are up to date as of REV 1 or REV 0 with bodges applied.

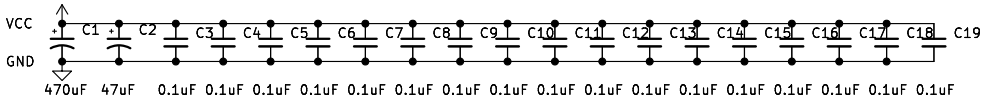

X1
ACO-12MHz
VCC
GND
8 PCLK
VIDSEL
R1 4k7
R2 220
Q1
2N3906
D1
LED
U1
74LS191
P0
P1
P2
P3
Q0
Q1
Q2
Q3
Pl
D/U
Ce
Cp
Rc
Tc
CCLK
SH/LD
HALFCLK
U3D
U4A
74LS30
BA10
BA9
BA8
BA7
BA6
BA5
BA4
BA3
Map 6845 to upper
16 bytes of
2k vram area
U8B
74LS74
DISPLAY
DLY_DISPLAY
U8A
CURSOR
DLY_CURSOR
J2
Slot Select
EXP6
EXP5
EXP4
EXP3
EXP2
EXP1
VIDSEL
D2
1N4148
BEN
U9B
74LS139
BA11
VIDSEL
VDPCS
VRAMCS
PIACS
ROMCS
Option ROM
Option ROM
EXPANSION CONNECTOR
J1
Expansion Connector
BD7
BD6
BD5
BD4
BD3
BD2
BD1
BD0
EXP6
EXP5
EXP4
EXP3
EXP2
EXP1
EXP0
BEN
R/W_2
E_Phi2
VPA
VCC
BA7
BA6
BA5
BA4
BA3
BA2
BA1
BA0
BA8
BA9
BA10
BA11
IRQ
NMI
RESET
HALT
BA
GND
U7
2716
A0
A1
A2
A3
A4
A5
A6
A7
A8
A9
A10
D0
D1
D2
D3
D4
D5
D6
D7
EP
Vpp
CE
ROMCS
Double ROM footprint to allow for
2716, 2732, 2764, 27128, 27256,
27512 ROMs to fit
U17
27128
A11
A12
A13
PGM
VPP
OE
Address Mux
74LS157
U11
U13
U15
IOa
I1a
IOb
I1b
IOc
I1c
IOd
I1d
Za
Zb
Zc
Zd
S
E
MA0
MA1
MA2
MA3
MA4
MA5
MA6
MA7
MA8
MA9
MA10
CA0
CA1
CA2
CA3
CA4
CA5
CA6
CA7
CA8
CA9
CA10
RAM_WR
VRAMCS

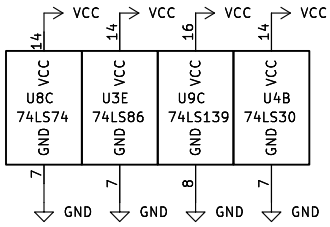

U2
MC6845
Charater RAM
U5
Character latch
U1
74LS574
CPU to Character RAM
74LS245
U6
R/W_2
VRAMCS
RAM_WR
Font ROM
U12
27128
Pixel Shift Register
U14
74LS166
U9A
74LS139
Normal
Inverse
JP2
Tweak values to suit voltages
VIDEO + SYNC =~ 1-1.2v
/VIDEO + SYNC =~ 0.3v
/video + /sync = 0v
Assume 75ohm load from tv
For 5v logic (HCT86 or such)
For TTL (~3v high) Substitute:
220 for 470
470 for 1k
Dirty hack of a Composite Video Sync...
JP1
2732ROM
U3A
U3B
U3C
74LS86
R4
220
R5 470
J3
Conn_Coaxial
J4
MDA/CGA connector
Font ROM
U18
27128
Double ROM footprint to allow for
2716, 2732, 2764, 27128, 27256,
27512 ROMs to fit


VCC
20
BD0 33
BD1 32
BD2 31
BD3 30
BD4 29
BD5 28
BD6 27
BD7 26
D0
D1
D2
D3
D4
D5
D6
D7
VCC
PA0 2 KEYD0
PA1 3 KEYD1
PA2 4 KEYD2
PA3 5 KEYD3
PA4 6 KEYD4
PA5 7 KEYD5
PA6 8 KEYD6
PA7 9 KEYD7
CA1 40 KEYSTROBE
BA0 36
BA1 35
RS0
RS1
U16
6821
CA2 39
JP3
DISPLAY
BLANKING
VCC 22
CS0
24
CS1
PIACS 23
CS2
R/W_2 21
R/W
E_Phi2 25
ENABLE
IRQ 38
IRQA
37
IRQB
RESET 34
RESET
VSS
1
GND
PB0 10
PB1 11
PB2 12
PB3 13
PB4 14
PB5 15
PB6 16
PB7 17
CB1 18
CB2 19
User Connector
J7
VCC
GND
JP4
VSYNC INT
VSYNC


KEYD0 1
VCC 9
KEYD1 2
10
KEYD2 3
11
KEYD3 4
12
KEYD4 5
13
KEYD5 6
14
KEYD6 7
KEYSTROBE 15
KEYD7 8
J5
DB15_Female


J6
DIP14 Keyboard Connector
KEYD0 1
KEYD1 2
KEYD2 3
KEYD3 4
KEYD4 5
KEYD5 6
KEYD6 7
14 VCC
13
12
11 GND
10
9 KEYSTROBE
8 KEYD7
Parallel ASCII keyboard
Pinout taken from Polymorphic
Systems VTI (1976) keyboard
connector


VCC
GND
C1
C2
C3
C4
C5
C6
C7
C8
C9
C10
C11
C12
C13
C14
C15
C16
C17
C18
C19
470uF
47uF
0.1uF
0.1uF
0.1uF
0.1uF
0.1uF
0.1uF
0.1uF
0.1uF
0.1uF
0.1uF
0.1uF
0.1uF
0.1uF
0.1uF
0.1uF
0.1uF
0.1uF


VCC
VCC
VCC
VCC
14
14
16
14
U8C
74LS74
U3E
74LS86
U9C
74LS139
U4B
74LS30
VCC
GND
7
7
8
7
GND
GND
GND
GND