An 80-column adapter project for the CoCo 1 and 2

Increasing Character Display

By Tony DiStefano
Rainbow Contributing Editor

CoCo 1 and 2 users, remember when the CoCo 3 came out? One of the good things built right into the CoCo 3 is an 80-character-by-24-line display screen. I guess you must have felt left out in the cold. If you have a CoCo 1 or 2 with a Multi-Pak and are using OS-9, your luck will change; you too can now have an 80- by 24-character display.

I say this is a big project not because it is hard to build, but because the overall project will take a bit of time and some hardware considerations. For instance, if you want an 80-by-24 display, you must have a monitor capable of displaying 80-by-24 characters. That requires an RS-70 compatible monitor of about 20 MHz resolution. You will also need a Multi-Pak and software. (You know what I think of software — leave it to the programmers.) CRC will send you OS-9 Level I Version 2 software to drive this display for about $10.

In the old days, an 80-by-24 character display required many chips, starting with the display chip. For many years, the most common display chip was the Motorola MC6845, back then a very powerful chip. It had a bunch of registers and counters that would divide a high-frequency clock into two lower frequencies. The higher of the two was the horizontal frequency, usually about 15 KHz; the other was the vertical frequency, at about 60 Hz. Also coming from this display chip was a character address. Part of this address went directly to a character-generator ROM. Now, a character-generator ROM is nothing more than a ROM with bit-mapped graphics of what letters and numbers are made of.

The other part of the character address went to the address lines of one side of some dual-ported RAM, which was ordinary RAM with some extra circuitry allowing two devices to read and write data to the same RAM. Also included in this circuitry was a circuit to switch between the two.

The other side of the dual-ported RAM usually was connected to a CPU, like the MC6809 CPU that is in all CoCos. The data lines of the RAM fed the rest of the address lines of the character ROM, and the ROM's data lines fed into a parallel-to-serial shift register. This shift register is the dot pattern that flows out of the display adapter and onto your screen. This dot pattern is mixed with the vertical and horizontal frequencies, called sync signals, into one signal that is called composite video.

Sound a little complicated? It might be at first, but read it again a couple of times and you'll understand it. Think of it like this: The CPU writes data characters into RAM, one character per byte of memory. The display chip, along with its support circuitry, reads this data and converts it into a stream of dots. With these dots come the signals necessary to control your monitor's circuitry to keep these dots in sync so that, to you, they look like characters such as letters and numbers.

What I have described requires a display chip (such as the MC6845), a character ROM, a RAM chip, about 20 other TTL support chips like the shifter and dual port circuitry. That makes quite a big job to design, let alone to do the board space and the wiring. But in the past, that's the way that technology was used.

Today things are different. Super LSI (Large Scale Integration) chips are here: The CoCo 3 is proof of that. The GIME chip is a video display adapter, a memory map decoder and a memory management adapter all in one. Chips like that contain thousands of TTL equivalents.

To make this 80-column display, I will use an LSI chip made by Standard Microsystems Corporation (SMC), the CRT 9128. This chip by itself does most of the work I described above. It has built-in character ROM, all porting circuitry, and all decoding and shifting chips. The only support chip it requires is RAM. Add that and a couple of decoding chips, and you are done.

As you can see from the diagram in Figure 1, there are not many components in this project. U1 is the SMC chip, U2 is the RAM chip. A 6116 is a 2K-by-8-bit RAM chip. U3 is used for decoding, and U4 is used for the video output and mixing to form composite video. In the diagram all the pins of U1 are numbered and have abbreviated names. Most of the names are self-explanatory; the ones that are not are listed below:

Pin No.	Name	Function
7	VID	serial video data
18	HS	horizontal sync
19	VS	vertical sync
20	CS	composite sync, (HS and VS combined)
8	INT	intensity pin

Before I get into the project's construction, a little information on the SMC chip in needed. The SMC chip has internal registers that control the many aspects required to display characters on the screen. In order to talk to this chip, we must know where it is in the memory map below:

Location	Direction	Function
$FF54	Write	Writes to data register
$F554	Read	Reads from data register
$FF55	Write	Writes to address register
$FF55	Read	Reads status register

The CPU communicates with the video chip via seven registers. Access to these registers is made by first storing the register number in the address

Tony DiStefano is a well-known early specialist in computer hardware projects. He lives in Laval Ouest, Quebec. Tony's username on Delphi is DISTO.

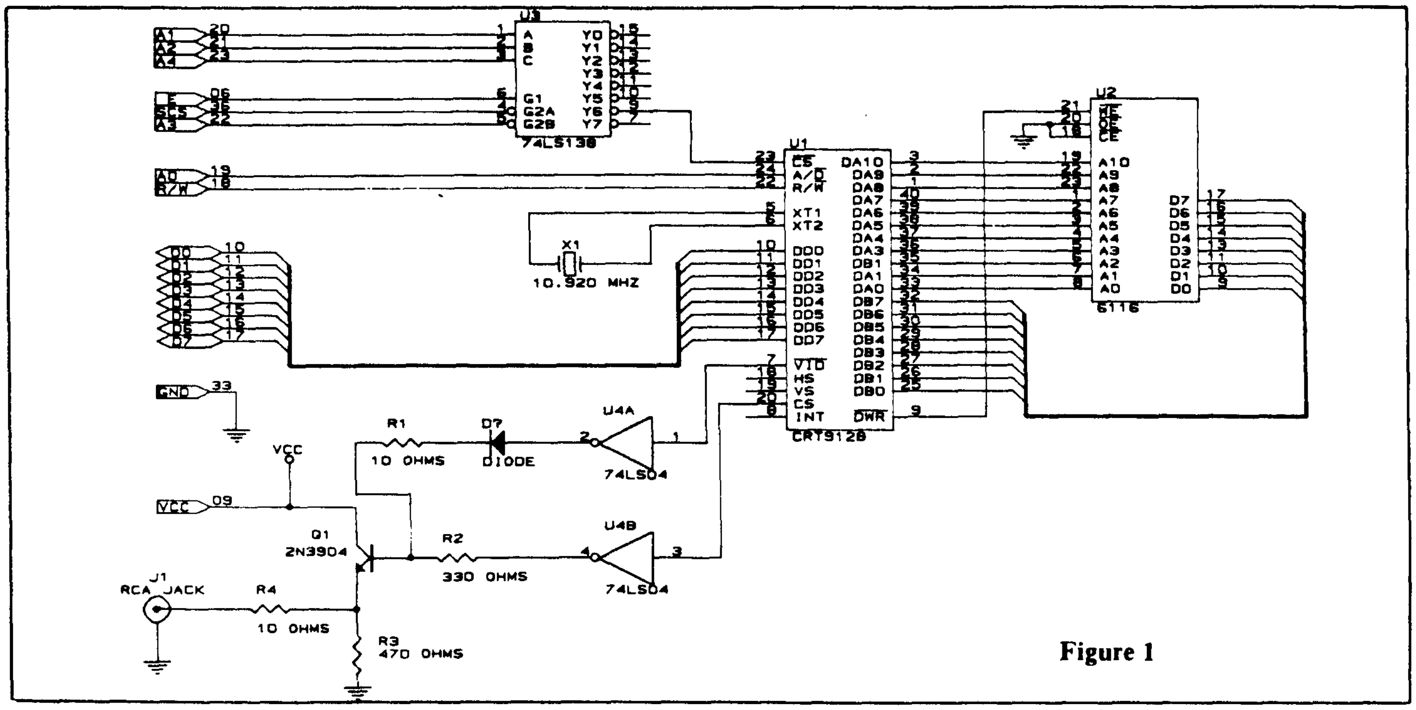

Figure 1

register and then accessing the register's data through the data register. The following is a list of the addresses and functions of available registers.

Address	Register Function
$6	Chip Reset
$8	TOS Add
$9	CUR Lo
$A	CUR Hi
$B	Fil Add
$C	ATT Dat
$D	Character
$E	Mode Register

For example, if you want to address the CUR Lo register, store the value $9 at $FF55 then store the CUR Lo byte at $FF54. Each of the seven registers has a specific function:

Chip Reset — The first thing done after powering up the chip. Stores $6 in $FF55 then stores a 0 value in the data register.

TOS Add — TOS stands for Top Of Screen. Top of screen address bits are DA10 to DA4, for D6 to D0, respectively. DA3 to DA0 are internally set to 0, forcing the first address at the beginning of each row to be 00, 16, 32 and so forth.

CUR L0 — Cursor low address position of flashing cursor. This is the first eight bits of the cursor address.

CUR Hi — Cursor high address position of flashing cursor. Bits D2 to D0 are DA10 to DA8, respectively. Other bits set to 0.

Fil Add — Fills address locations starting from cursor position to the fill address. Bits D6 to D0 are addresses from DA10 to DA4, respectively. As with TOS, the least three bits are always 0.

ATT Dat — Attribute Data, a register that changes the way things appear on the screen. The attribute byte:

D7 = 1 Enables block graphics
   = 0 Enables Alpha Mode
D6 = 1 Disables cursor (Invisible)
   = 0 Enables cursor (Visible)
D5 = 1 Underlines cursor
   = 0 Blocks cursor
D4 = 1 White screen and black characters
   = 0 Black screen and white characters
D3 = 1 Enables video suppress
   = 0 Allows character blinking
D2 = 1 Hi intensity
   = 0 Lo intensity
D1 = 1 Character underlined
   = 0 Character not underlined
D0 = 1 Character in inverse video
   = 0 Character in normal video

Character — Register where the ASCII character is placed to appear on the screen. If Bit D7 is set, the attributes described in the above byte (bits D3 to D0) will take effect on that character.

Mode — Auto increment mode. If Bit D7 is set to 1, the cursor address will automatically increment by one every time a byte is written to the character byte. If D7 is set to 0, the auto increment is disabled.

The basics for this display chip ought to be enough to get you started. If you want more detail, contact SMC at 35 Marcus Blvd., Hauppauge, NY 11788.

For this project you will need all the parts shown in Figure 1 and sockets for all the chips. The following is a list of socket sizes and the pin numbers for +5V and ground:

Chip No.	Socket Size	+5V	GND
U1	40	21	4
U2	24	24	12
U3	16	16	8
U4	14	14	7

You can get the SMC chip, project board, OS-9 software driver and RAM chip from CRC. Call (514) 383-5293 for prices.

There is one more interesting thing about the project. If you happen to have a Disto Super Controller or Disto Super Controller II, you can wire this project to the MEB connector. Two changes to the diagram in Figure 1 are necessary: Instead of A4, connect Pin 3 of U3 to VCC. Then, instead of A3, connect Pin 5 of U3 to GND. The rest of the connections appear on the bus. Instead of a project board, you can use just about any double-sided PC board. You will need, however, a 17-pin single inline female header. This way, you will not need a Multi-Pak.

Regarding the Multi-Pak, remember that when using the addresses from $FF40 to $FF5F, you must do a slot swap to whatever slot your hardware is, and swap back after you are finished. If you are in a multitasking environment remember to turn off the interrupts before swapping slots, and turn them back on again afterward.