CoCo 3 512K SRAM

Ciaran Anscomb <*www@6809.org.uk*>

A 512K RAM expansion design for the Tandy Colour Computer 3. The CoCo 3 is designed to accept such an expansion, so it's fairly straightforward. Plenty of boards have been produced at one time or another for this purpose. This design uses surface mount components and a single SRAM IC.

Fitting

These instructions are common to any 512K upgrade for the CoCo 3. First, remove the existing socketed DRAM chips (IC16, IC17, IC18, IC19). Cut capacitors C65 and C66. On a PAL CoCo 3, these capacitors are to the left of the GIME; on an NTSC machine they are closer to the DRAM sockets.

Looking at the motherboard with the ports to the rear, CN6 is the vertically aligned socket to the left of the original DRAM, with CN5 immediately below it effectively forming a single 24-pin socket. CN4 is below and to the right, running horizontally, just above the keyboard connector. Mount the expansion with J1 inserted into CN6/CN5, and J2 into CN4.

The rest of this document discusses the design of the expansion.

Interfacing with the CoCo 3

The RAM expansion sockets on the CoCo 3 comprise CN4, CN5 and CN6.

I couldn't find the pinouts of these connectors documented (which probably just means I didn't search hard enough), so here's what I traced.

Pin	CN4	CN5	CN6
1	GND	GND	GND
2	$\overline{\text{RAS}}$	D2	+5V
3	Z0	D3	D9
4	Z1	D1	D8
5	Z2	$\overline{\text{WE0}}$	D10
6	Z3	D0	D11
7	Z6	$\overline{\text{CAS}}$	D14
8	Z5	D7	D15
9	Z4	D5	D13
10	Z7	D4	D12
11	Z8	D6	$\overline{\text{WE1}}$
12	GND	GND	GND

Standard 2.54mm pitch header strip with 0.64mm^2 pins plugs into these sockets just fine; there's no need for machined pins.

SRAM choice

The CoCo 3 expects to address two banks of 256K x 8bit RAM and then externally latches 8 bits to the bus from one bank or the other for reads. For writes it provides two write-enable lines ($\overline{\text{WE0}}$ and $\overline{\text{WE1}}$) to select which bank is written to (the other will see a read cycle, but its data will be ignored).

The Cypress CY62146ELL[1] is a 256K x 16bit SRAM. It has a single write-enable line ($\overline{\text{WE}}$), but two more control lines: $\overline{\text{BHE}}$ and $\overline{\text{BLE}}$, which select whether the high or low eight bits are selected respectively. With a little external logic, these allow a single RAM chip to be used for the expansion.

For writes, $\overline{\text{WE1}}$ and $\overline{\text{WE0}}$ map directly to $\overline{\text{BHE}}$ and $\overline{\text{BLE}}$ on the chip (though we also want $\overline{\text{WE}}$ to be active if either $\overline{\text{WE1}}$ or $\overline{\text{WE0}}$ are). For reads, we want both banks to be selected. All of these signals are active low, so this is the behaviour we want to end up with:

$\overline{\text{WE1}}$	$\overline{\text{WE0}}$	$\overline{\text{WE}}$	$\overline{\text{BHE}}$	$\overline{\text{BLE}}$
0	0	0	0	0
0	1	0	0	1
1	0	0	1	0
1	1	1	0	0

This translates to some very simple logic:

- $\overline{\text{BHE}}$ = $\overline{\text{WE1}}$ *AND NOT* $\overline{\text{WE0}}$
- $\overline{\text{BLE}}$ = *NOT* $\overline{\text{WE1}}$ *AND* $\overline{\text{WE0}}$
- $\overline{\text{WE}}$ = $\overline{\text{WE1}}$ *AND* $\overline{\text{WE0}}$

Two inverters and three AND gates are required.

Address latches

Typically, a row address is strobed into DRAMs on $\overline{\text{RAS}}$ fall, then a column address on $\overline{\text{CAS}}$ fall. The SRAM devices we want to use do not multiplex their addresses like this even if they are arranged similarly internally. Instead, they have a pin for every address line. Therefore we need to latch the row addresses ourselves. The column address can be passed through.

The row addressing requires 9 bits of latch. I've chosen to use a 74'574 for the first 8 bits as its pinout works better on the board than using a '273. The tri-state capability of the '574 is not needed, so its $\overline{\text{OE}}$ line is tied low (always active).

For the ninth bit I've used a 74'1G79 to save board space.

Note that inverting versions of both ('573 and '1G80) should be perfectly fine if there's a cost or availability reason for picking those instead. The address lines presented to the SRAM just have to be *consistent* and *unique*.

All of these latch ICs operate on a rising clock, so we need an inverter for the $\overline{\text{RAS}}$ line.

Bill of materials

Three inverters are required. It would be natural to use a hex inverter IC ('04) and leave the other three unused, but it turns out they're needed at opposite ends of the board. I've instead chosen to use three individual inverters.

[1] There is also the CY62146ESL, which can operate at lower voltages, but handily is still fine with 5V.

In general I've chosen 74AHCT logic. It's lower power than the 74LS logic common in our era of computer, but will interface to TTL signals.

Component	Value	Package	Qty	Description
U1,U5,U6	74AHCT1G04	SOT-23	3	Single inverter
U2	74AHCT08	TSSOP-14	1	Quad AND gate
U3	74AHCT574	TSSOP-20	1	Octal D-type latch
U4	74AHCT1G79	SOT-353	1	Single D-type latch
U7	CY62146ESL	TSSOP-II-44	1	256Kx16bit SRAM
C1-C6	100n	0603	6	Bypass capacitors
J1	24-pin	Pin header	1	2.54mm pin header
J2	12-pin	Pin header	1	2.54mm pin header

Note: SOT-353 is also known as SC-70.

Similar products

There are many! As mentioned, the CoCo 3 was designed to accept such an expansion, so there were several early products using DRAM, for which you can find manuals on the TRS-80 Color Computer Archive[2].

There are also a few later ones doing the same job with SRAM:

- RETRO Innovations published a simple through-hole design called the HalfMegCoCo[3]. It's based around two SRAM ICs, one per 8-bit bank.
- Boyson Tech produced the Boomerang Classic 512K[4]. From the small picture, it seems like it also uses a single SRAM. All logic appears to be handled by a CPLD. As Boyson Tech have left the CoCo market, it's unclear who will now produce these.
- Cloud-9 makes the Triad 512K SRAM Upgrade[5]. It's a little larger, but surface mount and again based around two SRAM ICs.

Further development

Jim Brain from RETRO Innovations has observed that the logic could be simplified further. If you assume that $\overline{\text{WE1}}$ and $\overline{\text{WE0}}$ will never be active (low) at the same time, you get a slightly simpler truth table (where 'X' means "don't care"):

$\overline{\text{WE1}}$	$\overline{\text{WE0}}$	$\overline{\text{WE}}$	$\overline{\text{BHE}}$	$\overline{\text{BLE}}$	Action
0	0	X	X	X	Invalid
0	1	0	0	1	Write high byte
1	0	0	1	0	Write low byte
1	1	1	0	0	Read both bytes

Which means:

[2] `https://colorcomputerarchive.com/repo/Documents/Manuals/Hardware/`
[3] `https://github.com/go4retro/HalfMegCoCo.git`
[4] `https://boysontech.com/512k-memory/`
[5] `http://www.frontiernet.net/~mmarlette/Cloud-9/Hardware/512K.html`

- $\overline{\text{BLE}}$ = *NOT* $\overline{\text{WE1}}$
- $\overline{\text{BHE}}$ = *NOT* $\overline{\text{WE0}}$
- $\overline{\text{WE}}$ = $\overline{\text{WE1}}$ *AND* $\overline{\text{WE0}}$

Which now only requires one *AND* gate, and could be realised with a (smaller) 74'1G08. Furthermore, the whole thing can be implemented using *XOR* gates:

- $\overline{\text{BLE}}$ = $\overline{\text{WE1}}$ *XOR* 1
- $\overline{\text{BHE}}$ = $\overline{\text{WE0}}$ *XOR* 1
- $\overline{\text{WE}}$ = *NOT* ($\overline{\text{WE1}}$ *XOR* $\overline{\text{WE0}}$)

which is equivalent to:

- $\overline{\text{WE}}$ = $\overline{\text{WE1}}$ *XOR* (*NOT* $\overline{\text{WE0}}$)

where we can substitute *NOT* $\overline{\text{WE0}}$ with the above $\overline{\text{BHE}}$ providing:

- $\overline{\text{WE}}$ = $\overline{\text{WE1}}$ *XOR* $\overline{\text{BHE}}$)

Which means it can all be implemented using three gates from a quad *XOR* (74'86). This also leaves the last gate free to implement the inverter for $\overline{\text{RAS}}$ (albeit requiring that signal to be routed a little further). And, when using *XOR* gates to realize the logic, they actually take care of the invalid state. If $\overline{\text{WE0}}$ and $\overline{\text{WE1}}$ are both 0, not only will $\overline{\text{BHE}}$ and $\overline{\text{BLE}}$ both be 1 (inverted from the original signals), but $\overline{\text{WE}}$ will be 1 *XOR* 0, or 1, or inactive.

Thanks, Jim!

License

Revision history

2021-03-30 Updated text clarifying which signals are active low
2021-03-28 Traced expansion connector signals through to the GIME (prompted by Jim Brain)
2021-03-27 New *Further development* section (information from Jim Brain)