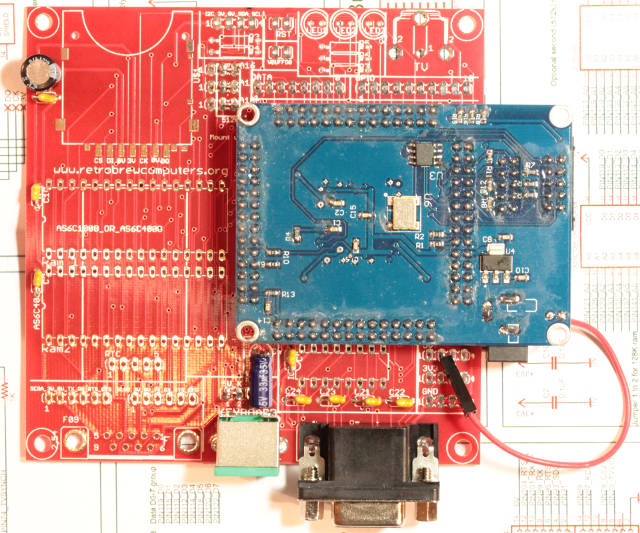
**Multicomp Cyclone II-C**

Portions of the hardware and designs on this page are based on Grant Searle's original work, which was published with the following license:

“By downloading these files you must agree to the following: The original copyright owners of ROM contents are respectfully acknowledged. Use of the contents of any file within your own projects is permitted freely, but any publishing of material containing whole or part of any file distributed here, or derived from the work that I have done here will contain an acknowledgement back to myself, Grant Searle, and a link back to this page. Any file published or distributed that contains all or part of any file from this page must be made available free of charge.” -<http://searle.hostei.com/grant/Multicomp/index.html>, retrieved 12/14/15.

**Introduction**

The Multicomp Cyclone II-C (v1.10) is a motherboard which provides peripheral connectivity for a Cyclone II FPGA development board commonly available on eBay. The board was designed “by committee” in the retrobrewcomputers forum during December 2015, with James Moxham doing all the actual work (schematic capture and PCB layout). One of the prime motivators for this board was to fit a 10cmx10cm footprint because the manufacturing cost of PCBs of this size is particularly attractive. It provides connectivity for the following peripherals:

* VGA Screen with PS/2 Keyboard interface
* Two RS-232 serial ports:
  + Serial A uses on-board level-shifter and DE-9F connector *or* external level shifter via 6-pin connector
  + Serial B uses external level shifter via 6-pin connector
* SD Card connector
* Two RAM sockets to support up to 1 MB of RAM.
* 5-pin connector for external DS1302 real-time clock and backup coin cell
* 3V3 or 5V power option for PS/2 Keyboard, with voltage clamp diodes.
* 3 LEDs
* RCA Video connector
* Jumpers for reset and for boot control
* Debug/Expansion headers for data bus and uncommitted FPGA pins

Due to the header installation on the Cyclone II module, the module is required to be mounted inverted onto the motherboard.

The photo in this section shows a minimum build configuration using PS/2 keyboard and VGA output; no SDcard, no external RAM, and no serial ports.

**Compatibility**

The FPGA pin assignment for the V110 board is a superset of the pin assignment used in Grant's original design. Therefore, his original design files should work on this design with no modification. There are two caveats:

* To use Grant's design files, the three memory links must be set correctly (see [Construction (Part 2)](https://www.retrobrewcomputers.org/doku.php?id=boards:sbc:multicomp:cycloneii-c:start#construction_part_2))
* Grant's original SD controller only works with SD standard capacity cards (upto 5GBytes in size). The 6809 code presented here uses a modified design which supports both standard capacity and high-capacity (SDHC) cards.
* The CycloneIIc-Z80 (18 Aug 2016) project below currently only works with SD standard capacity cards. To enable use with SDHC cards, copy the sd\_controller.vhd file from the [mc-2g-1024.zip](https://www.retrobrewcomputers.org/lib/exe/fetch.php?media=builderpages:rhkoolstar:mc-2g-1024.zip) project and overwrite the existing file in \Components\SDCARD and recompile.

**Parts List**

| **Qty** | **Ref** | **Description** | **Note** |
| --- | --- | --- | --- |
| 1 |  | Cyclone II FPGA Development/Breakout Board. |  |
| 1 |  | Cyclone II V1.10 PCB |  |
| 4 |  | 2 row x 14 contact 0.1“ pitch female header connector |  |
| 1 |  | VGA D15 right-angle connector (male shroud, female pin) | 1 |
| 3 | R10,12,14 | 470R resistor | 1 |
| 3 | R11,13,15 | 680R resistor | 1 |
| 1 | R18 | 470R resistor | 2 |
| 1 | R19 | 1k resistor | 2 |
| 1 |  | PCB-mounted RCA socket | 2 |
| 1 |  | Surface-mount SD card socket |  |
| 1 |  | RTC module | 3 |
| 2 |  | 2-pin male header |  |
| 2 | C3,4 | 470uF 6V3 Electrolytic |  |
| 3 | C12,2,10 | 0.1uF capacitor |  |
| 5 | C9,21-24 | 0.1uF capacitor | 4 |
| 1 |  | PS/2 keyboard socket | 1 |
| 2 | D1,2 | 3V6 zener diode | 1 |
| 2 | R1,2 | 10k resistor | 1 |
| 2 | R3,4 | 10k resistor | 7 |
| 2 |  | 32-pin 0.6” DIL IC sockets |  |
| 2 |  | 1 AS6C1008 128Kx8 or (preferred) 1 or 2 AS6C4008 512Kx8 |  |
| 3 | R5,6,7 | 4K7 resistor |  |
| 3 | LED1,2,3 | Red LED, 0.1“ pitch through-hole | 6 |
| 1 |  | D9 right-angle connector (male shroud, female pin) | 4 |
| 1 | IC9 | SP3232E RS232 level shifter; 14-pin 0.3” DIL | 4 |
| 1 |  | 14-pin 0.3“ DIL IC socket | 4 |
| - |  | wire for jumpers | 5 |

Notes:

1. Only if you want PS/2 keyboard and VGA output
2. Only if you want mono TV output
3. Only if you want a real-time clock
4. Only if you want RS232 to 9-way D
5. Off-cuts from resistor leads will do fine for this
6. These are the current-limiting resistors for the LEDs. The schematic shows a value of 5K6 but that seems a bit large/makes the LEDs too dim. Your mileage may vary. Draft v1.11 board specifies 1K.
7. Only if you are using I2C header.

Other connectors and components optional for adding external expansion connectors:

* 6-pin 0.1” connector for Serial B external level shifter
* 6-pin 0.1“ connector for Serial A external level shifter
* 4-pin 0.1” connector and 2x 10K resistor for external I2C
* 10-pin 0.1“ connector for GPIO connection
* 8-pin 0.1” connector for Data Bus connection

You will also need:

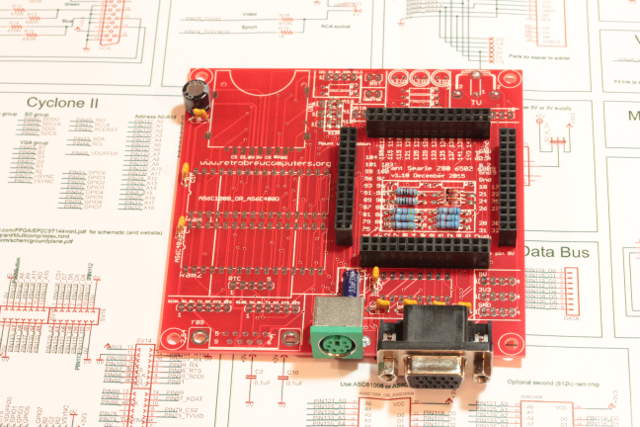
* Altera USB Blaster programming adaptor
* A computer with Altera's (free) Quartus software installed

Component sourcing:

* The FPGA board is available on <http://www.ebay.com> – search for *ALTERA Cyclone II EP2C5T144 CPLD FPGA Development Board*.
* The PCB files are freely available so you can choose a vendor, upload the files and have a batch made. Once a couple of people have done this, there should be boards available through the forum (since few people are likely to want all 10 boards from a minimum order). I got my boards from <http://www.dirtypcbs.com> and they turned out well. The design is “shared” there so that anyone can order the board without the need to upload design files (You can go directly to the design at this link: <http://dirtypcbs.com/store/designer/browse/9663> or you can select “PCB Store” then use your browser's page search facility to locate “Cyclone2v110”. I recommend changing the board thickness from the default of 1.2mm to 1.6mm).
* I couldn't find 2 row x 14 contact 0.1“ headers, but I found some longer ones on ebay and cut them down with a hack-saw.
* The SDcard socket is for a full-size card. There are at least 2 variants with different footprints (I had to do some minor mechanical mods to get mine to fit)
* The VGA and PS/2 connectors ought to be source-able from an old PC motherboard. Otherwise, ebay.
* Ebay again for the RAMs. Be sure to get 0.6” DIL parts. The AS6C4008 is only slightly more expensive than the AS6C1008. The Z80 system can make use of extra memory (paged RAM disk) and that capability is in progress for the 6809 system.
* The SP3232 is made by Exar. The Maxim MAX3232 is an alternative. Apparently some of the MAX3232 parts on Ebay are fakes so you might want to use Farnell/CPC/RS or some other reputable source.
* If you have no RS232 connector on your host machine you can use a serial-to-USB adaptor. Once again, ebay. Be sure to get one that can run on 3.3V power. Search for *USB to serial 3.3v*
* The RTC module is available on.. ebay – search for *Arduino DS1302 RTC*
* Surface-mount SD card socket is Sparkfun part number PRT-12769

In all cases look at the photos here to check that the component you are choosing matches the footprint on the board.

**Construction**

There are various build options, as indicated in the notes of the parts list above. For now, these notes assume that you will use the PS/2 keyboard and VGA display connections. You can start by assembling the bare minimum system and go through the process of programming the FPGA board with a pre-built configuration and making sure that everything is working. Then you can go on to add the remaining components and reprogram the FPGA board. When you have assured yourself that everything is working correctly with the pre-built configurations you will have a proven stable base to go forward with either hardware or software experimentation.

**Minimum Build**

For a minimum build, you need only fit:

* Decoupling capacitors (four 0.1uf and the two electrolytics). In the photo, the four 0.1uF caps for the RS232 level shifter are also fitted, but they don't need to be.
* The PS/2 connector
* The VGA connector
* The 6 VGA resistors
* The 2 PS/2 pullup resistors
* The 2 zener diodes (see “PS/2 keyboard power” below)
* The PS/2 power link (see “PS/2 keyboard power” below)
* The 5V flying lead (see “PS/2 keyboard power” below)
* The 4, 2×14 connectors for attaching the FPGA board

When fitting the 2×14 connectors, attach the FPGA board to them before soldering them into position; that will ensure that they are aligned correctly.

Have a copy of the schematic to hand while assembling the board. Because the board is so dense, some of the silk-screen references are a little hard to read. The schematic should help resolve any ambiguity.

With the minimum build completed, program the FPGA using one of the pre-built minimum configurations below.

**PS/2 keyboard power**

Grant's original design used 3.3V power for the PS/2 keyboard. Not all PS/2 keyboards will run at this voltage so the board provides the option of supplying either 3.3V or 5V power to the keyboard, using a wire jumper just behind the PS/2 connector.

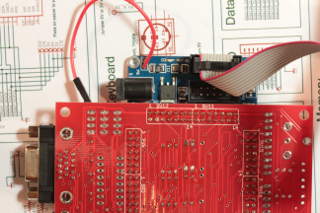
Solder a wire jumper from the middle connection to either the 5V or the 3V connection.

The two zener diodes prevent 5V from the keyboard from over-driving the FPGA inputs. They are optional but are strongly recommended if you configure the board for 5V keyboard. With a 3V keyboard they are not needed (you can still fit them if you want; they will do no harm).

With a 5V keyboard, you also need to route 5V to the board from the FPGA board, where it is tapped off directly from the input power socket. The easiest way to do this is with a short flying lead from one of the four marked 5V pads close to the VGA connector. On the FPGA board there is a 2-pin 0.1“ connector/footprint in the corner, next to the input power socket. Use a meter to buzz out and see which pin is 5V (the other is 0V).

Theoretically, you can connect a USB keyboard using a passive USB-to-PS/2 adaptor. If anyone has done that successfully, please add a note here.

**FPGA Programming**

The Cyclone II FPGA is configured through internal RAM cells, and therefore its configuration is not maintained across a power cycle. The appearance of non-volatile operation is achieved by programming the configuration into a EEPROM (which is also on the FPGA breakout board) which is automatically loaded by the FPGA at power-up.

The FPGA breakout board has 2 separate 10-pin programming sockets. One is marked JTAG and is used for programming the FPGA directly (configuration endures until a power cycle). The other is marked AS (the “Active Serial” programming port) and is used for programming the EEPROM. Each connector is keyed, and the keying is rotated 180degress from one connector to the other. Programming through the JTAG port requires a .sof file. Programming through the AS port requires a .pof file. Usually, the run through Quartus will create both .sof and .pof files.

**Pre-built FPGA images**

Pre-built images are available for some configurations; you can use them “as-is” or as a starting point for your own modifications. See [the Downloads section](https://www.retrobrewcomputers.org/doku.php?id=boards:sbc:multicomp:cycloneii-c:start#downloads) below.

**Configuring the USB Blaster for use under Linux**

The USB-Blaster works fine under Linux. The user guide tells you to create the file /etc/udev/rules.d/51-usbblaster.rules and containing the text (on a single line):

BUS=="usb", SYSFS{idVendor}=="09fb", SYSFS{idProduct}=="6001",MODE="0666", PROGRAM="/bin/sh -c 'K=%k; K=$${K#usbdev}; printf /proc/bus/usb/%%0

3i/%%03i $${K%%%%.\*} $${K#\*.}'", RUN+="/bin/chmod 0666 %c"

I found that, on my Kubuntu system, that line did not work and I used this instead:

SUBSYSTEMS=="usb", ATTRS{idVendor}=="09fb", ATTRS{idProduct}=="6001",MODE:="0666"

Both incantations are attempting to do the same thing: to change the protection on the USB Blaster automatically when it is plugged in.

**Programming**

You can program the FPGA breakout board when it is still connected to the V110 PCB. You do not need to supply power whilst programming: the programming cable provides the necessary power.

Programing is very quick; it takes about 4s.

https://www.retrobrewcomputers.org/lib/exe/fetch.php?tok=f89205&media=http%3A%2F%2Fwww.retrobrewcomputers.org%2Flib%2Fimages%2Fsmileys%2Ffixme.gifdescribe in detail from within Quartus

**Software**

It goes without saying (I hope..) that if you have configured the FPGA breakout board for a 6809 processor design, you can only use 6809 software. However, for the initial bring-up, everything you need is in ROM.

**Bringup/debug of minimal system**

The description here assumes that you have built a minimal system and programmed it with the minimal 6809 image, and that the programming completed successfully.

Connect the FPGA breakout board to the V110 PCB, and connect the 5V flying lead (if applicable). Connect the PS/2 keyboard and VGA monitor. Power-up the monitor. Connect power to the FPGA breakout board and power-up.

You should be greeted with a message in white 40-column text on the VDU:

6809 CamelForth v1.0 25 Apr 95

If not, and assuming nothing went *bang* investigate a little more:

* The keyboard caps-lock LED should be illuminated. It should toggle when you press the caps-lock key repeatedly. This is implemented in hardware and has nothing to do with processor choice or code ROM.
* The monitor should show a small flashing cursor - either after the message, or (if there's no message) in the top left-hand corner.

Here are some debug suggestions:

| **Effect** | **Possible cause** |
| --- | --- |
| kbd LED toggles OK, message on VDU | Everything seems to be working fine |
| kbd LED toggles OK, no message on VDU, no cursor | faulty monitor or cable. Bad joint on VGA connector, R10-R15 missing or faulty |
| kbd LED toggles OK, no message on VDU, flashing cursor | processor not executing code. Should not occur with “proven” pre-built image.. |
| no kbd LED, message on VDU | Bad kbd, missing or incorrect kbd power jumper, R1-R2 missing or faulty, D1-D2 faulty or wrong orientation |
| no kbd LED, no message on VDU, flashing cursor | Bad kbd, missing or incorrect kbd power jumper, R1-R2 missing or faulty, D1-D2 faulty or wrong orientation, processor not executing code. Should not occur with “proven” pre-built image.. |
| no kbd LED, no message on VDU, no cursor | Bad kbd, missing or incorrect kbd power jumper, R1-R2 missing or faulty, D1-D2 faulty or wrong orientation, faulty monitor or cable. Bad joint on VGA connector, R10-R15 missing or faulty, bad power supply, incorrectly programmed FPGA |

Once the keyboard and VDU are working as expected, try pressing a few keys; they should be echoed on the screen. The processor is running a FORTH intepreter/compiler. I think that FORTH makes a great debug monitor. If you don't know any FORTH I suggest you spend a few minutes reading the [introduction and tutorial to Gforth](https://www.complang.tuwien.ac.at/forth/gforth/Docs-html/Introduction.html#Introduction).

A couple of things to note about CamelForth:

* It is case sensitive, so keep caps-lock on.
* There is (currently) no (easy) way to save any code, so don't write anything too big.

The final piece of testing on the minimal system is to see the colour output on the VDU. Type in this program. Be careful with case (everything is in upper case except the m near the end) and with spaces:

: DIG DUP 10 / 48 + EMIT 10 MOD 48 + EMIT ;

: GR 27 EMIT ." [" DIG ." ;" DIG ." m" ;

Now try this:

44 31 GR

42 93 GR

0 30 GR

0 41 GR

0 0 GR

31 40 GR

32 40 GR

34 40 GR

47 34 GR

45 94 GR

1 34 GR

0 0 GR

1 37 GR

: GREET ." HELLO WORLD " ;

32 40 GR GREET 44 31 GR GREET

You should see coloured output on the display. Notice the two levels of intensity (visible in the white). If your output looks different, check that R10-R15 are fitted and the correct values.

Consult Grant's site for a complete list of the ANSI escape codes supported by his VDU design.

You can leave the board like this, if you want, and explore CamelForth further. Alternatively, continue with the construction; in particular, fitting an external memory.

**Construction (Part 2)**

Once you have some signs of life from the board, you can go ahead with the remainder of the construction. Fit:

* The RAM and RAM address links (notes below)
* The on-board RS232 (connector, caps and
* An external RS232/USB adaptor
* The three LEDs and their current-limiting resistors
* The SDcard socket. The footprint for this component is not standard so you might need to improvise a little.
* The Real-time clock module. Check the pin connections on the schematic; it should wire 1-to-1 from the module down to the PCB. You can mount it on top of the RAM chip with a double-sided sticky foam pad.
* 2-pin jumpers for RESET and VDUFFD0
* Any other connectors you want to use either for debug or expansion

**RAM selection and configuration**

You can fit one or two RAM devices. If you only fit 1, it should be the device closest to the SDcard socket – three, 2-way links control the high-order addresses on that device.

To use Grant's original design (which does not drive the high-order address lines) wire all three links pin1-pin2. Otherwise, wire all three links pin2-pin3 (marked “512K” on the PCB silk-screen).

The latest Z80 and 6809 designs can access more than 64Kbytes of memory through a simple memory-mapping unit.

**Bringup/debug of full system**

Use the same FPGA configuration (internal RAM) to complete the testing. Connect and power-up the board as before. It should boot as before and display the CamelForth message.

Test the reset. There is a reset button on the FPGA breakout board; once you get used to where it is, you can reach around and press it even though you can't see it! Alternatively, you can fit a switch or a jumper to the “RST” pins near the LEDs. Note that the VDU logic is not reset by reset and so the screen is not cleared by reset; the processor is reset and so code execution restarts (so you will see the CamelForth message after each reset). If you use a jumper or paper-clip to short the “RST” pins you will probably see a very bouncy reset; multiple resets in quick succession.

Test the VDUFFD0 jumper. When fitted, it swaps the hardware decode of the VDU/keyboard and the SerialA port. Fit the jumper and press reset. There should be no output to the VDU. Pressing the Caps-Lock key should continue to toggle the Caps-Lock LED.

Test the Serial A RS232 port. Use a “straight” male-to-female cable (wire 2-2, 3-3, 5-5 as a minimum) to connect to a VT100.. or a terminal emulator. Configure the terminal for 115200baud, 8 data bits, 1 stop bit, no flow control. Press reset with the VDUFFD0 jumper fitted; the CamelForth message should be displayed on the monitor, and you should be able to interact with CamelForth using the terminal.

I made the mistake of fitting a male connector on my PCB, to match my female-to-female null-MoDem cable. The fix to this is to wire a female (F) connector back-to-back with a male (M) connector, wired F1-M5, F4-M3 F3-M2

Test the external memory. CamelForth uses RAM internal to the FPGA, in the address range 0x0000-0x07FF. The ROM and I/O occupy address range 0xD000-0xFFFF. All the rest of the memory space is RAM.

Run a single pass memory test like this (note PASS - singular):

HEX

1000 1FFF PASS

Or 16 (0x10) passes like this (note PASSES - plural):

HEX

1000 1FFF 10 PASSES

If this works OK, test the whole memory:

HEX

0800 CFFF 10 PASSES

Here are the definitions for the memory test routines (you don't need to type these in; they are already in ROM):

simple memory test

: STUP

start-addr end-addr -- start-addr end-addr

2DUP SWAP DO I DUP ! CELL +LOOP ;

: STUPI

start-addr end-addr -- start-addr end-addr

2DUP SWAP DO I NEGATE I ! CELL +LOOP ;

: TSTUP

start-addr end-addr -- start-addr end-addr

2DUP SWAP DO I DUP @ <> IF

." Error@" I U. THEN CELL +LOOP ;

: TSTUPI

start-addr end-addr -- start-addr end-addr

2DUP SWAP DO I NEGATE I @ <> IF

." Error@" I U. THEN CELL +LOOP ;

: PASS STUP TSTUP STUPI TSTUPI ; ( do 1 pass)

: PASSES 0 DO ." ." PASS LOOP ; ( do N passes)

Even if you don't know FORTH you should be able to see how definitions build on earlier (older) definitions. STUP does an incrementing memory fill where each pair of bytes contain their address (and TSTUP checks that pattern and reports errors); STUPI does an incrementing memory fill where each pair of bytes contain the 1-s complement (inverse) of their address (and TSTUPI checks that pattern and reports errors).

If the memory test fails, try reading and writing a small group of locations and try to infer what's wrong:

HEX ( THIS IS A COMMENT - NO NEED TO TYPE IT)

1000 20 AB FILL ( FILL FROM 1000 FOR 20 BYTES WITH VALUE AB)

1000 30 DUMP ( DISPLAY MEMORY FROM 1000 FOR 30 BYTES)

( REPEAT THE FILL/DUMP LINES BUT REPLACE "AB" WITH THESE NUMBERS IN TURN: 0 1 2 4 8 10 20 40 80 55 AA FF)

This should allow you to see if there are any data bus problems (shorts/opens). To look for addressing problems try this:

HEX ( THIS IS A COMMENT - NO NEED TO TYPE IT)

1000 2000 STUP ( FILL FROM 1000 TO 2000 WITH AN INCREMENTING PATTERN )

1000 30 DUMP ( DISPLAY MEMORY FROM 1000 FOR 30 BYTES)

( LOOK FOR DUPLICATE VALUES IN THE DUMP. REPEAT THE DUMP, CHANGING THE RANGE. TRY USING STUPI TO FILL WITH INVERSE VALUES)

You can also use STUP and DUMP to fill/inspect memory. For example:

OK 1000 1080 STUP 1000 80 DUMP

1000 10 00 10 02 10 04 10 06 10 08 10 0A 10 0C 10 0E ................

1010 10 10 10 12 10 14 10 16 10 18 10 1A 10 1C 10 1E ................

1020 10 20 10 22 10 24 10 26 10 28 10 2A 10 2C 10 2E . .".$.&.(.\*.,..

1030 10 30 10 32 10 34 10 36 10 38 10 3A 10 3C 10 3E .0.2.4.6.8.:.<.>

1040 10 40 10 42 10 44 10 46 10 48 10 4A 10 4C 10 4E .@.B.D.F.H.J.L.N

1050 10 50 10 52 10 54 10 56 10 58 10 5A 10 5C 10 5E .P.R.T.V.X.Z.\.^

1060 10 60 10 62 10 64 10 66 10 68 10 6A 10 6C 10 6E .`.b.d.f.h.j.l.n

1070 10 70 10 72 10 74 10 76 10 78 10 7A 10 7C 10 7E .p.r.t.v.x.z.|.~

OK 1000 1080 STUPI 1000 80 DUMP

1000 F0 00 EF FE EF FC EF FA EF F8 EF F6 EF F4 EF F2 ................

1010 EF F0 EF EE EF EC EF EA EF E8 EF E6 EF E4 EF E2 ................

1020 EF E0 EF DE EF DC EF DA EF D8 EF D6 EF D4 EF D2 ................

1030 EF D0 EF CE EF CC EF CA EF C8 EF C6 EF C4 EF C2 ................

1040 EF C0 EF BE EF BC EF BA EF B8 EF B6 EF B4 EF B2 ................

1050 EF B0 EF AE EF AC EF AA EF A8 EF A6 EF A4 EF A2 ................

1060 EF A0 EF 9E EF 9C EF 9A EF 98 EF 96 EF 94 EF 92 ................

1070 EF 90 EF 8E EF 8C EF 8A EF 88 EF 86 EF 84 EF 82 ................

OK

(typing the commands in this way will leave extra stuff on the stack, which you can ignore or clean up with ABORT. It makes the examples self-contained).

If the RAM appears to fail catastrophically, try removing it and see if you get the same behaviour. If you do, what you're seeing is just capacitance. Either the RAM is faulty or, more likely, you have set the jumpers wrongly (Ahem, I did exactly that and it took me ages to realise my mistake).

To save you trying it, I can assure you that fitting the RAM the wrong way around is a \_bad\_ thing to do; the board will not power up (in my case, my carelessness was not punished; the board and RAM worked correctly when I turned it around.

The most likely RAM problems are a short or open or a bent pin or bad jumpers. If you get stuck, go to to the forum and ask for help.

If you have a RAM problem, you might suspect a faulty PCB. Although this is possible, it's very unlikely. At least 12 batches of these boards have been manufactured, by multiple companies, with no apparent problems and the boards themselves are individually electrically tested.

Test the “Drive” LED. With no SDcard fitted, type this:

FLEX

LED1 should light and stay on (LED1 is the LED closest to the “RST” jumper). You need to press the reset button to recover (LED1 should go off).

Test the SDcard controller. Power-off the board and fit an SDcard (one that you don't mind corrupting). There is no card-detect mechanism so you always need to power-cycle to change the card. Type this:

HEX

200 80 DUMP

0 200 SDRD

200 80 DUMP

The 2nd line inspects the current contents of memory from 0x200-0x27F. The 3rd line loads block 0 (512 bytes in size) from the SDcard to RAM at address 0x200. The 4th line shows the final contents of memory. Assuming the SDcard is not filled with 0, you will see the memory contents change between the first and the second DUMP.

If the system hangs when you press return after the SDRD, the SDcard is not recognised or there is a connection problem.

Once everything is working, you can reprogram the FPGA with the full image - this uses the external RAM, which frees up internal RAM on the FPGA so that the VDU can be configured for 80-column operation.

If the system does not boot with the full image, go back to the minimal image and test the memory. This is really the only thing that can be causing a problem

**Preparing an SDcard (6809 system)**

For the 6809 system, the SDcard does not contain anything that would generally be recognised as a partition table or filesystem. Rather, it is a concatenated set of binaries and floppy disk images. To prepare the SDcard you need a tool like dd (for Linux or Mac OS). For Windows, there is a program on sourceforge called “Win32 Disk Imager” that claims to perform the equivalent task. FIXMEI haven't tried it. Please update this if you have, or if you can recommend a suitable tool.

**Linux Detailed Instructions**

Download multicomp09\_sd.img (see [Downloads](https://www.retrobrewcomputers.org/doku.php?id=boards:sbc:multicomp:cycloneii-c:start#dowloads) below) to your host computer and plug in the SDcard. *Do not mount the SDcard*. Use “tail /var/log/syslog” to identify the device associated with the SDcard

If you get this wrong you could corrupt your hard drive so be careful.

The device will be something like sdb (it may be reported as “sdb sdb1”). Copy the file to the device like this (change the device as appropriate)

sudo dd if=multicomp09\_sd.img of=/dev/sdb

When the copy completes, remove the SDcard (it has no filesystem so you cannot “unmount” it).

**Using the SDcard (6809 system)**

Power off the V110 and insert the SDcard. When you power on it should boot to CamelForth as usual. The table below lists the commands used to start the different software environments that are on the SDcard image:

| **Command** | **Description** |
| --- | --- |
| BASIC | Load and start Microsoft 8K BASIC. This is the image created by Grant which is used as the boot ROM on his original design. |
| FLEX | Load and start the FLEX operating system. There are 4 virtual drives. |
| BUGGY | Load and start Lennart Benschop's 6809 debug monitor. The killer feature of this debug monitor is its built-in line assembler/disassembler. It also has a breakpoint capability and a single-step capability (using the single-step hardware in multicomp09). |
| CUBIX | (then type b and press return) Load and start Dave Dunfield's CUBIX operating system. There are 4 virtual drives. |
| NITROS9 | Load and start NITROS9, an open-source re-engineered OS9. There are 4 virtual drives. |

**Where next?**

Although there is a “canned” software download here, the latest versions and all the component parts/source code and other notes are in the [multicomp6809 repository on github](https://github.com/nealcrook/multicomp6809). There is additional documentation on the [associated wiki](https://github.com/nealcrook/multicomp6809/wiki).

**A little bit more FORTH**

Regardless of how much memory you have fitted, the ROMed FORTH uses RAM in the address range 0x0000-0x03FF for its workspace. Definitions that you create go into RAM from address 0x0000 upwards. There are some variables stored at the top of the RAM then a region assigned for the return stack and another region assigned for the data (parameter) stack.

Here's a simple example that allows you to see how definitions are stored in memory. Reset the system and enter these commands:

: GREET ." Hello World" ;

: HI CR GREET ;

HEX 0 80 DUMP

The dump shows how the two definitions are laid out in memory. You can see the complete list of definitions built in to CamelForth by typing WORDS:

6809 CamelForth v1.0 25 Apr 95

WORDS UINIT #INIT NITROS9 FLEX BUGGY BASIC CUBIX CDTOCD EFTOCD MMUMAP PIVOTRST P

IVOT piv SDRD256n SDRD256 SDRDn SDWR SDRD SDIDLE SDLBA10 SDLBA2 PASSES PASS TSTU

PI TSTUP STUPI STUP DUMP .A .B H. .HHHH .HH .H COLD .S WORDS ENVIRONMENT? DEPTH

MOVE WITHIN POSTPONE ['] IMMEDIATE REVEAL HIDE ] [ RECURSE CREATE ( [CHAR] CHA

R ' ABORT" ?ABORT ABORT QUIT EVALUATE INTERPRET ?NUMBER>NUMBER ?SIGN DIGIT? LIT

ERAL FIND IMMED? NFA>CFA NFA>LFA WORD>COUNTED /STRING SOURCE C, , ALLOT HERE HE

X DECIMAL . U. SIGN #> #S #>DIGIT <# HOLD UD\* UD/MOD ." S" (S") TYPE ACCEPT UMA

X UMIN SPACES SPACE CR COUNT 2OVER 2SWAP 2DUP 2DROP 2! 2@ MIN MAX \*/ \*/MOD MOD /

/MOD \* FM/MOD SM/REM M\* DABS ?DNEGATE DNEGATE ABS ?NEGATE S>D R0 L0 PAD S0 LP H

P LATEST 'SOURCE DP STATE BASE>IN U0 TIB TIBSIZE BL CELL LOOP LOOP ENDLOOP LEA

VE DO L>>L REPEAT WHILE AGAIN UNTIL BEGIN ELSE THEN IF USER VARIABLE CONSTANT ;

: DOES> (DOES>) !DEST ,DEST ,BRANCH ,EXIT !COLON ,CF !CF COMPILE,>BODY CHARS C

HAR CELLS CELL ALIGN ALIGNED SCAN SKIP CMOVE> CMOVE S= FILL UM/MOD UM\* J I ( L

OOP) (LOOP) UNLOOP (DO) ?BRANCH BRANCH U> U<> <<> = 0<0= RSHIFT LSHIFT ! 2/

2\* 1- 1><INVERT XOR OR AND NEGATE - M C@ @ C! ! RP! RP@ SP! SP@ R@ R>>R P

ICK TUCK NIP ROT OVER SWAP DROP ?DUP DUP EXECUTE LIT EXIT EMIT KEY? KEY

OK

(that output was wrapped at 80 columns, the width of my terminal, and may not be the latest version)

**Even more FORTH**

As of the June 2016 release, CamelForth includes support for block words, for loading/storing binary blobs and has (stored as part of the SDcard image) a screen-based block editor. Here is a quick getting-started guide.

The file storage uses a traditional “block” format. Each block is 1Kbyte is size and a block can hold anything you choose: txt, source code, data or binary data. There is no management of what is stored where; that is the programmer's responsibility.

The default block file contains 256 blocks (and is therefore 256KByte in size), with source code stored in the low-order block numbers. Blocks are numbered from 0 (but, you cannot load code from block 0 for various arcane historic reasons, so block zero is just use for comments)

0 LIST ( DISPLAY BLOCK 0)

1 LIST ( DISPLAY BLOCK 1)

1 LOAD ( LOAD BLOCK 1.. WHICH IN TURN LOADS SOME OTHER BLOCKS)

Once the load has completed (which takes about 10s) some new capabilities are available:

WORDS ( WORDS NOW COLOUR-CODES THE WORDS, AND HAS A BUILT-IN PAGER - PRESS SPACE FOR THE NEXT SET)

16 LIST ( DISPLAY THE COMMANDS FOR THE SCREEN EDITOR)

1 EDIT ( START THE SCREEN EDITOR)

( ctrl-N FOR NEXT SCREEN, ctrl-P FOR PREVIOUS SCREEN)

( ctrl-E TO EXIT.. LOOK AT BLOCK 16 FOR THE COMPLETE LIST)

Also, when the code was compiled it was stored back to a region of the block file in compiled form (in a format that I call a “blob”). After a reset or power-cycle you can load it back in like this:

HEX 800 ALLOT 64 RDBLOB

For more details read follow these two links:

[CamelForth](https://github.com/nealcrook/multicomp6809/wiki/camelforth) [CamelForth save/load](https://github.com/nealcrook/multicomp6809/wiki/CamelForthSaveLoad)

**Downloads**

| **Link** | **For processor** | **Updated** | **Description** |
| --- | --- | --- | --- |
| [cyclone2v110.pdf](https://www.retrobrewcomputers.org/lib/exe/fetch.php?media=boards:sbc:multicomp:cycloneii-c:cyclone2v110.pdf) | any | xxDec2015 | PDF schematic of the V110 printed ciruit board |
| [cyclone2v110.zip](https://www.retrobrewcomputers.org/lib/exe/fetch.php?media=boards:sbc:multicomp:cycloneii-c:cyclone2v110.zip) | any | xxDec2015 | Eagle design database of V110 printed circuit board |
| [ep2c5t144mini.pdf](https://www.retrobrewcomputers.org/lib/exe/fetch.php?media=boards:sbc:multicomp:cycloneii-c:ep2c5t144mini.pdf) | any | xxJul2018 | Leon Heller's schematic of the EP2C5T144mini FPGA board |
| [multicomp09\_sd.zip](https://www.retrobrewcomputers.org/lib/exe/fetch.php?media=boards:sbc:multicomp:cycloneii-c:multicomp09_sd.zip) | 6809 | 09Jan2016 | SDcard image for 6809 |
| [multicomp09\_sd.zip](https://www.retrobrewcomputers.org/lib/exe/fetch.php?media=boards:sbc:multicomp:cycloneii-c:multicomp09_sd_jun2016.zip) | 6809 | 11Jun2016 | OBSOLETE SDcard image for 6809. Updates: CamelForth utilities, single-step now works in BUGGY |
| [multicomp09\_sd.zip](https://www.retrobrewcomputers.org/lib/exe/fetch.php?media=boards:sbc:multicomp:cycloneii-c:multicomp09_sd_nov2016.zip) | 6809 | 24Nov2016 | OBSOLETE SDcard image for 6809. Updates: Major additions to FLEX, added FUZIX |
| [multicomp09\_sd.zip](https://www.retrobrewcomputers.org/lib/exe/fetch.php?media=boards:sbc:multicomp:cycloneii-c:multicomp09_sd_apr2017.zip) | 6809 | 22Apr2017 | SDcard image for 6809. Updates: Replaced NITROS9 Level 1 with NITROS9 Level 2. |
| [multicomp09\_iram.zip](https://www.retrobrewcomputers.org/lib/exe/fetch.php?media=boards:sbc:multicomp:cycloneii-c:multicomp09_iram.zip) | 6809 | 09Jan2016 | 6809 system with 2Kbytes of internal RAM. Works with or without external RAM. VDU runs at 40×25, colour. Use this image for debug/board bring-up |
| [multicomp09.zip](https://www.retrobrewcomputers.org/lib/exe/fetch.php?media=boards:sbc:multicomp:cycloneii-c:multicomp09.zip) | 6809 | 09Jan2016 | 6809 system with no internal RAM. Requires at least 1 external RAM. VDU runs at 80×25, colour |
| [multicomp09.zip](https://www.retrobrewcomputers.org/lib/exe/fetch.php?media=boards:sbc:multicomp:cycloneii-c:multicomp09_jun2016.zip) | 6809 | 11Jun2016 | 6809 system with no internal RAM. Requires at least 1 external RAM. VDU runs at 80×25, colour. Updates: UART reset on reset. ESC key now generates 0x1b. CamelForth fixes and additions: FUZIX boot, BLOCK words, load/store pre-compiled code blobs, BLOCK Editor |
| [multicomp09.zip](https://www.retrobrewcomputers.org/lib/exe/fetch.php?media=boards:sbc:multicomp:cycloneii-c:multicomp09_nov2016.zip) | 6809 | 05Nov2016 | 6809 system with no internal RAM. Requires at least 1 external RAM. VDU runs at 80×25, colour. Updates: VDU now handles line insert and delete correctly. |
| [cycloneIIc-Z80](https://www.retrobrewcomputers.org/lib/exe/fetch.php?media=boards:sbc:multicomp:cycloneii-c:cycloneiic-z80.zip) | Z80 | 18Aug2016 | Z80 System for Cyclone IIc PCB. Supports 2 serial ports, VGA, SD card and up to 1MB RAM. Default serial port speed is 9600. Mini-Monitor replaces the Multicomp monitor. Boots ROM-MJS |
| [mc-2g-1024.zip](https://www.retrobrewcomputers.org/lib/exe/fetch.php?media=builderpages:rhkoolstar:mc-2g-1024.zip) | Z80 | 24Dec2017 | Z80 System for Cyclone IIc PCB (or equivalent) Provides 6 CP/M variants as well as Grant's NASCOM “ROM” Basic. Also included are 10 extra volumes with popular applications. Updates: SD card module now uses a 250 kHz SPI clock when initializing. Keyboard improvements when using built-in RCA or VGA terminal. Fix bug with setting memory disk size in 512Kb systems. Correction to install instructions. See [rhkoolstar's builder page for more details](https://www.retrobrewcomputers.org/doku.php?id=builderpages:rhkoolstar:mc-2g-1024) |
| [cgasanserif\_multicomp\_fonts.zip](https://www.retrobrewcomputers.org/lib/exe/fetch.php?media=boards:sbc:multicomp:cycloneii-c:cgasanserif_multicomp_fonts.zip) | any | 24Nov2018 | Alternate font ROM. |

**Rebuilding a Quartus project from source**

In general, the pre-build FPGA bundles in the downloads section do *not* contain everything that you need to rebuild the project in Quartus. The directory structure looks like this:

project1/

project2/

project3/

ROMS/

Components/

Where the ROMS and Components directories are common to all of the projects. The FPGA bundle will contain the files for (eg) “project1” but only a skeleton/subset of the files needed for the “ROMS” and “Components” directories.

If you want a complete tree, start by downloading the zip from Grant's site and then overlaying it with files *either* from the Z80 design *or* from the 6809 design (because the Z80 and 6809 designs have each modified some of Grant's original files, you are advised to keep independent trees of files if you are working with both)

The additional/latest 6809 files can be downloaded from [github](https://github.com/nealcrook/multicomp6809/tree/master/multicomp).