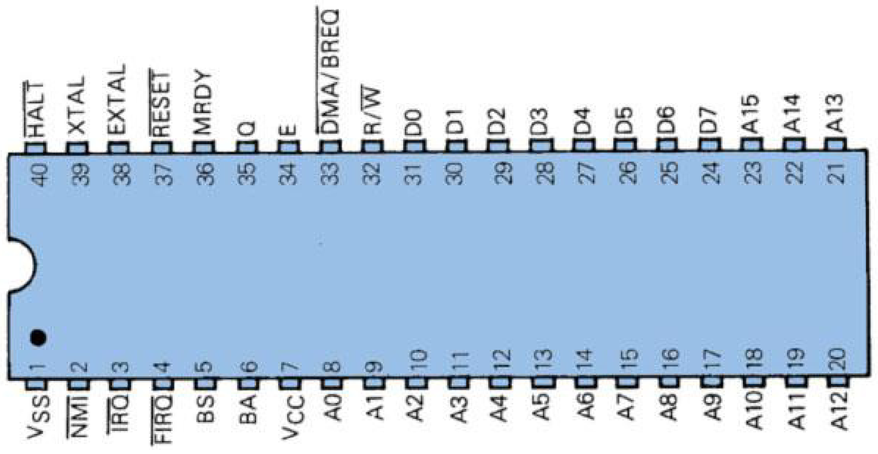


Grant's 6-chip 6809 computer

(only 5 chips if using a USB to TTL serial cable)

- a fully operational 6809 computer running BASIC can't get simpler than this!

VERIFIED WORKING
RE-BUILT FROM INFO PROVIDED HERE

by Grant Searle

For news and updates, follow me on Twitter:

Follow

Last update: 17th November 2013

*[FOR A **Z80** VERSION OF THE MINIMAL COMPUTER, CLICK HERE]*
*[FOR A **6502** VERSION OF THE MINIMAL COMPUTER, CLICK HERE]*

[FOR A VERSION ON A LOW-COST FPGA BOARD CLICK HERE]

Please note that you are NOT allowed to reproduce any of this page elsewhere on the Web without my permission.

Specification

16K ROM
32K RAM
68B09 Processor with a 7.3728MHz crystal (1.8432MHz clock)
115200 Baud serial interface, RS232 specification voltage levels
Power consumption - 200mA
Microsoft Extended BASIC, as used in the Tandy Coco 2 (modified for the SBC with all I/O via serial. Commands not applicable for the SBC have been removed)
Minimal possible component count - 6 ICs and a small number of discrete components

Memory Map

0000-7FFF 32K RAM
8000-9FFF FREE SPACE (8K)
A000-BFFF SERIAL INTERFACE (minimally decoded)
C000-FFFF 16K ROM (BASIC from DB00 TO FFFF, so a large amount of free space suitable for a monitor etc)

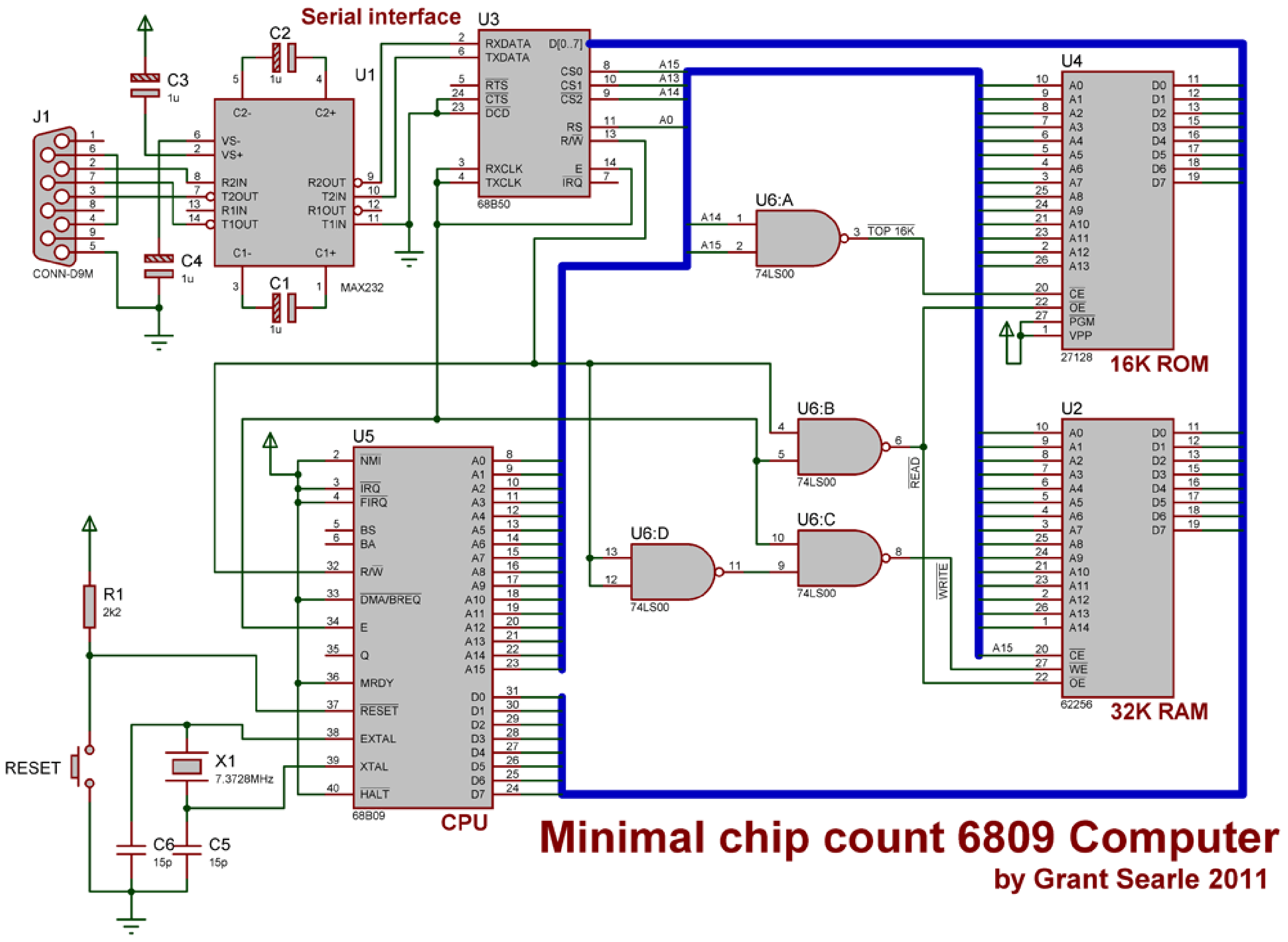
Circuit diagram

The purpose of this computer is to create the simplest possible machine with a high speed interface, good amount of RAM and also a good implementation of BASIC.
The design that I produced is shown here, and is probably the simplest 6809 circuit that can be done to fulfil what I need. This can be used as the basis of more complex machines.

The ACIA is very minimally decoded. If a more refined decoding is needed then it is straightforward to add additional gates or a 74LS138 decoder. However, even with this simple arrangement, there is still 8K of address space free for interfacing as needed.

Note: Chip selection - All of the standard 6850 chips that I have worked perfectly at the speeds required for this circuit even though the circuit requirement is faster than their specification. Therefore, you are unlikely to have any issues. However, I would recommend you buy the "B" speed grade ACIA ie. 68B50 if possible.

Power supply pins and any (optional) decoupling capacitors are <u>not</u> shown and need to be connected to the appropriate power rails.

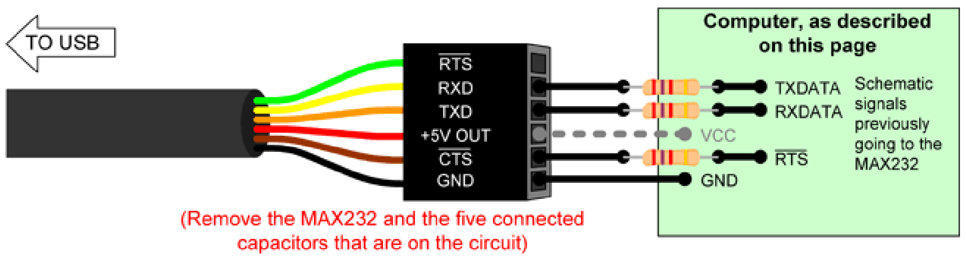


A MAX202 could be used instead of the MAX232, in which case C1-C5 should be 0.1uF.

Alternative serial (and power) connection

The circuit assumes a connection to a standard RS232 port. However, this could also be connected to a USB to TTL (5V) serial cable instead. If doing this, the MAX232, capacitors C1 to C5 and the serial connector are not needed. This therefore reduces the chip count by 1.

Using a USB to 5V (TTL) serial cable instead of an RS232 interface


TO USB
RTS
RXD
TXD
+5V OUT
CTS
GND
Computer, as described on this page
TXDATA
RXDATA
VCC
RTS
GND
Schematic signals previously going to the MAX232

(Remove the MAX232 and the five connected capacitors that are on the circuit)

Resistors are present to safely limit current along the cable if the USB or computer is not powered. These are not needed if using +5V OUT to VCC to power the computer - connect directly instead.

By using a USB to serial cable the board can also be powered from the USB port down the same cable (a powered hub could be used if higher current needed), eliminating the need for a separate power supply for this board. Just connect the +5V out on the cable to the Vcc

supply on the board, as shown on the dotted line.
IMPORTANT - only connect Vcc to the USB cable if there is no other power supplied to the board.
The 2k7 resistors are present in the diagram because the board may be powered but not plugged in to the USB cable, or the USB cable could be plugged in and active without power to the board. These limit the current to avoid power being drawn through the interface pins. If always powering the board via the cable then no resistors are necessary.

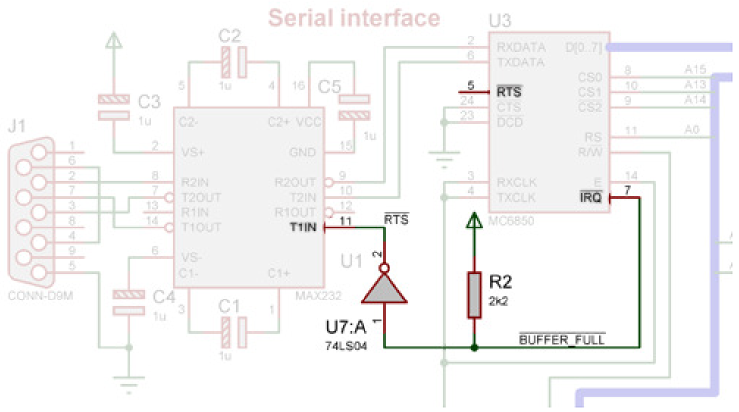
Enabling hardware handshaking

By adding one more chip *(a 74LS04 (NOT gates), alternatively, 74LS00 (NAND gates) or 74LS02 (NOR gates))* the RTS line on this board can be used to control the data flow. This will completely eliminate missing characters and the terminal software will no longer need to have delays at end of lines of between characters when transferring to this board.

To ensure data transfer is stopped when the 6809 is not ready for it (ie. the previous byte hasn't been read) the 6850 receive interrupt is enabled. As soon as a character is received, the IRQ on this chip will go low. This is connected, via an inverter, to the /RTS serial connection. The /RTS will therefore go high as soon as a character is received. It will remain high until the character is read (a read of the data resets the IRQ status). Once it has been read, the /IRQ will go high, so the /RTS will go low, allowing the PC to resume transfer. This is repeated as each character is received, controlling the data transfer until the 6809 is ready for it.

The latest version of the ROM code on this page must be used if making this modification.

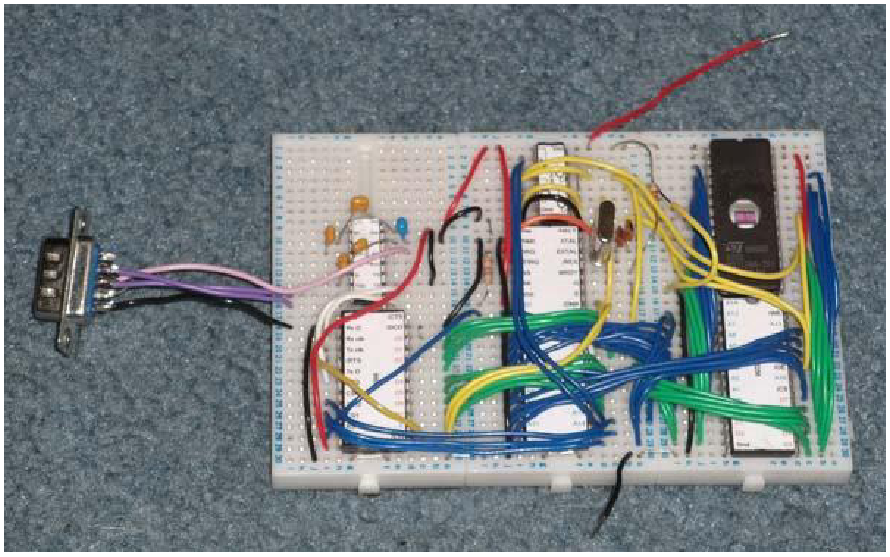
The changes are shown below
U7:A and R2 are the new components
Pin 11 (T1IN) on U1 is no longer connected to pin 5 (/RTS) on the 6850 (U3). Instead, the /IRQ signal on the 6850 is inverted by the new gate and this becomes /RTS which is then connected to T1IN on the MAX232. The pullup resistor is needed because the /IRQ output is open-collector.



(This also applies if using the TTL cable instead of a MAX232)

Prototype

The circuit shown on this page is built here. As you can see, I have attached labels (download HERE) to the chips to allow the wiring to be very straightforward and error-free. Blue wiring is the address lines, green is the data lines, red and black are power lines and yellow are the other connections.



ROM BASIC assembly listing

Update 14th August 2013 - tidy up of source code to remove hard-coded values for tokens. Unused tokens removed. ACIA will lower the interrupt pin when character received to allow hardware flow control. Removed vectored extensions (no need as the source can be changed directly).

AS9 was used to assemble the BASIC listing.
I took the listing from "Color BASIC Unravelled II" and updated it to assemble correctly under AS9.
The resulting ROM dump was compared to the actual ROM dump to ensure it matched perfectly before proceeding.
I then removed all code and commands that were specific to the screen, graphics, keyboard and sound.
I wrote a serial handler to control the text I/O, along with suitable Control-C break handling.
Unused variable locations were removed.
The command/function tables were updated as needed for the updated command set, and pointer values adjusted accordingly.
Once complete, I put it onto ROM to ensure all commands for the "Standard" BASIC ROM were working properly.

To allow quick-checks to any changes I made, I needed an emulator. However, I couldn't find any suitable, so I wrote my own in Visual Basic which emulates the complete 6809 CPU architecture and handles the I/O needed for this SBC.

Once working perfectly, I then added commands from the "Extended" BASIC ROM (from the "Extended BASIC Unravelled II" book), merging it with the existing code and updating command tables etc. as needed.

The ROM in the SBC is fully working with all the commands in the Standard and Extended ROMS that are applicable to the SBC.

Here is the zip containing the ASM and the LST file. This is a merge of the standard and extended ROMs as mentioned on my page.

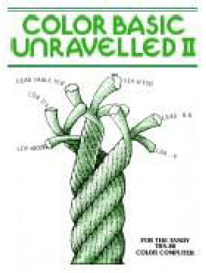
It was assembled using as9 (see link later on this page) with the following command...

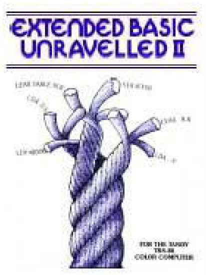
as9 exbasrom.asm -now l s19

"-now" suppresses warnings
"l" produces a lst file
"s19" produces an s19 hex file

If you use as9, you will probably get a "Null pointer assignment" message when it completes. This can be ignored, and is a known bug with as9 that doesn't affect operation (according to the readme file that came with it)
I actually maintain the listing in Excel, as I have standard, enhanced and commented code colour-coded and a macro will then produce the asm file that is in the zip. Using excel and my macros, I can comment-out code just by changing the colour to grey - any grey code is not exported to the ASM file.

The following files are available in many places on the internet. I have included them here for completeness. Click on the image to download the PDF.


COLOR BASIC UNRAVELLED II
FOR THE TANDY TRS-80 COLOR COMPUTER


EXTENDED BASIC UNRAVELLED II
FOR THE TANDY TRS-80 COLOR COMPUTER

ROM dump and source code

The zip file of the HEX dump of the ROM and source code is here.

Note that the first part is "FF" values, because BASIC starts at DB00 but the ROM starts from C000. There is a lot of space for inclusion of monitor programs etc.

ROM BASIC - details of what has been included/excluded

INCLUDED TOKENS

(From the Color BASIC ROM)
FOR, GO, REM, ELSE, IF, DATA, PRINT, ON GOSUB, ON GOTO, INPUT, LINE INPUT, END, NEXT, DIM, READ, RUN, RESTORE, RETURN, STOP, POKE, CONT, LIST, CLEAR, NEW, EXEC, TAB, TO, SUB, THEN, NOT, STEP, +, -, *, /, ^, AND, OR, >, =, <

(From the Extended BASIC ROM)
DEL, DEF, LET, RENUM, FN, USING, &, &H, TRON, TROFF, EDIT

Secondary functions

(From the Color BASIC ROM)
SGN, INT, ABS, USR, RND, SIN, PEEK, LEN, STR$, VAL, ASC, CHR$, LEFT$, RIGHT$, MID$, INKEY$, MEM

(From the Extended BASIC ROM)
ATN, COS, TAN, EXP, FIX, LOG, SQR, HEX$, VARPTR, INSTR, STRING$, MID$ (MODIFICATION), POS

EXCLUDED TOKENS

The following are not in this SBC version of the ROM as they are not applicable...

(From the Color BASIC ROM)
CLOAD, CSAVE, OPEN, CLOSE, LLIST, SET, RESET, CLS, MOTOR, SOUND, AUDIO, SKIPF, OFF

(From the Extended BASIC ROM)
LINE, PCLS, PSET, PRESET, SCREEN, PLCEAR, COLOR, CIRCLE, PAINT, GET, PUT, DRAW, PCOPY, PMODE, PLAY, DLOAD,

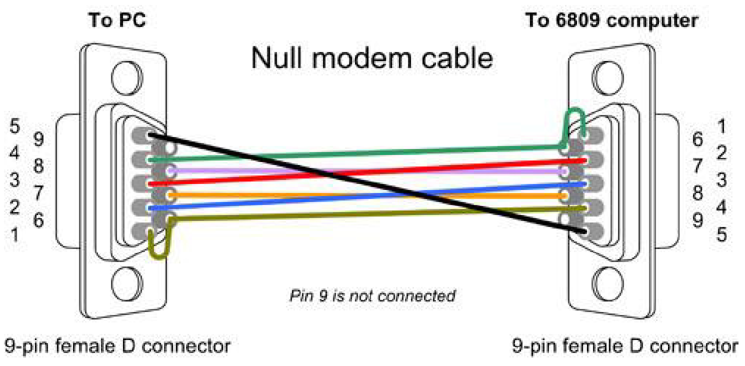
Secondary functions

(*From the Color BASIC ROM)*
EOF, JOYSTK, POINT, TIMER, PPOINT

(From the Extended BASIC ROM)
none

Powering-up

You will need a suitable connection to a PC or other terminal display. You will probably need a null-modem cable to connect this to a PC serial port. Here is my recommended wiring.



Connect to a PC or similar terminal, with 115200 baud, 8 bits, no handshake, no parity, 1 stop bit.
If the handshaking modification is done, **must** set the terminal to have **hardware** handshake (RTS/CTS).

Power on the board and press the RESET button.
After a few seconds (memory test) the following will appear on the terminal:

6809 EXTENDED BASIC
(C) 1982 BY MICROSOFT

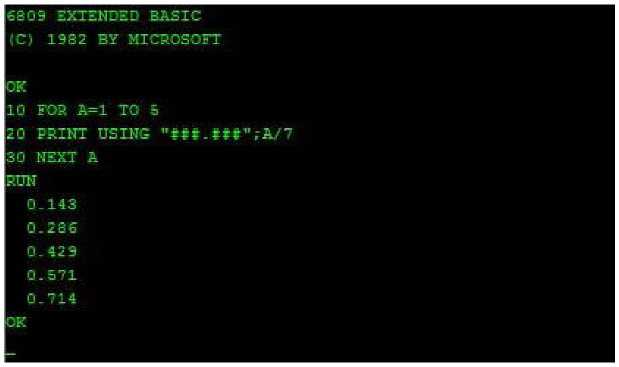
OK

The machine is now ready to use.

Typing
PRINT MEM
will give (current ROM version, may be slightly different with future releases)...
32016
OK
This shows that the 32K RAM has been recognised correctly.

...the rest is up to you.

A simple program entered and run on the terminal showing one of the more powerful "Extended" BASIC commands...



```
6809 EXTENDED BASIC
(C) 1982 BY MICROSOFT

OK
10 FOR A=1 TO 5
20 PRINT USING "###.###";A/7
30 NEXT A
RUN
  0.143
  0.286
  0.429
  0.571
  0.714
OK
_
```

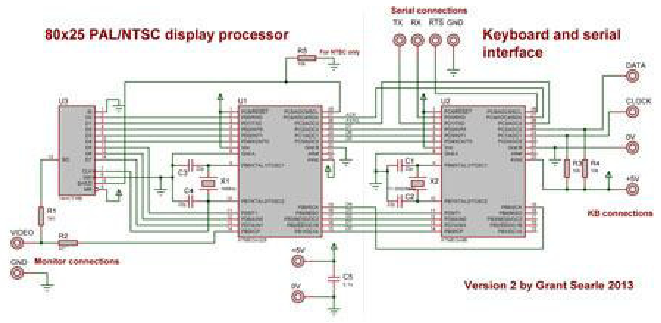
Loading and saving

Save
Type LIST but don't press enter.
In your terminal window on the PC, or whatever you are using to operate this board, enable the capture to a text file.
Press ENTER.
The listing will then be spooled to the text file on your PC.
Once the listing is complete, end the capture. The contents of the program will then be on the PC for later use.

Load
Most terminal programs have a method to pass a text file to the destination.
Ensure the board is ready to receive characters and then transfer the text file from the PC to the board using the terminal.
If the hardware handshake modification shown higher up on this page is **not** done then when transferring, you MUST have a delay after each new line, otherwise the BASIC interpreter will not be ready to accept the next line, so characters will be lost. You may possibly need a small delay after each character sent as well. This can be done in "HyperTerm" (for Windows) and may be available in other terminal emulators.

Using a keyboard and monitor instead of the PC interface

This board, as for my other boards, can use my monitor and keyboard interface here



Links

The following links were very useful to me...

Albert van der Horst - AS9 assembler

Arto Salmi - 6809 Emulation and links page

My other pages

CLICK HERE TO GO TO MY MAIN PAGE FOR MY OTHER PROJECTS

I hope this page has been useful.

Grant.

To contact me, my current eMail address can be found here. Please note that this address may change to avoid spam.