HD6309 Single Board Computer

Description

The HD6309 SBC is an attempt to build an SBC around the "ultimate 8 bit CPU"

Specifications

- 3 MHz Hitachi HD63C09 microprocessor w/ external clock
- 64KB RAM (2x 32KB SRAM)
- 32KB EPROM (27C256)
- Parallel Interface & Timers (Zilog Z8536 CIO)
- Dual Serial Channels (Zilog Z85C30 SCC)
- Realtime Clock (~~Epson RTC7301~~ Maxim DS3232)
- micro-SD memory card
- USB slave (FTDI FT230XS) connected to SCC Ch. A

Electrical Details

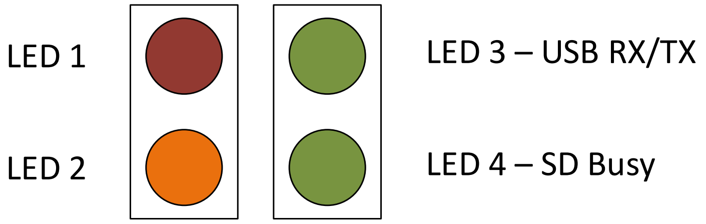
Power is provided either via USB-B connector, or via 2.1mm coax connector J9. **5V INPUT VOLTAGE**, center pin is positive, shield is ground. There is a reverse biased 1A diode across J9 to prevent reverse polarity from damaging the SBC.

Mechanical Details

SBC form factor is 160mm x 100mm which fits nicely within a Hammond 1558 extruded aluminum enclosure. I had the PCB fabricated by OSH Park with the usual, wonderful, purple results.

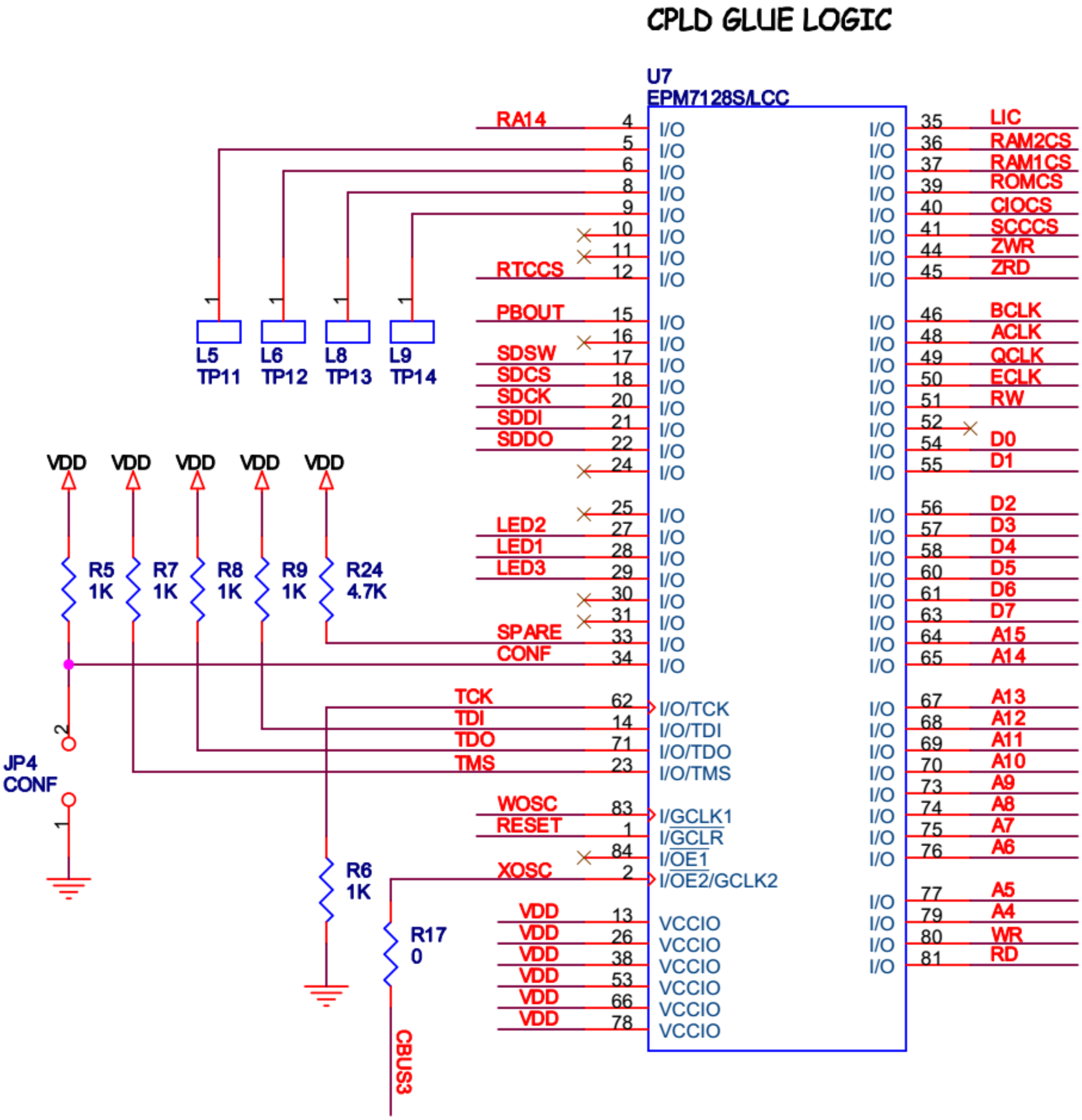
There are two pushbuttons, SW1 and SW2. SW1 is a hardware RESET. SW2 is readable by the CPU and can be used for user programs. Each is 'debounced' by a ZXCM205 supply voltage monitor that generates a ~200ms long pulse.

There are four LEDs, LED1-LED4 – looking at the edge of the PCB:



CPLD Glue Logic

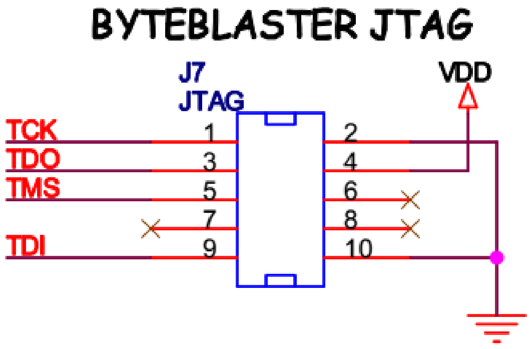
A 128-macrocell CPLD EPM7128 (PLCC 84 package) is used extensively for glue, SPI, clock logic. This is 5V I/O flash CPLD.



TP11-14 are provided to help with debugging of the CPLD. Presently (CPLD V1.3, V1.4) they are mirrors of the SPI signals to the SD card, to allow easier probing.

CONF is a jumper that determines the state of the EPROM pin 27 (A14 on a 27C256) after reset. The level on CONF is latched and output on RA14, which in turn is connected to the A14 input of the 27C256 EPROM. Therefore, it can be used to select between two different boot behaviors. The RA14 signal can be changed during runtime, allowing access to the entire EPROM space.

CPLD JTAG programming is possible via J7, the standard 10-pin JTAG connector.



CPU Clock Generation

Y1 is a 24 MHz oscillator and is the master clock for the CPLD and CPU. Alternately, the CBUS3 output of the FT230XS can provide a 24 MHz clock; populate R17 and remove Y1 to use this option. **Note: If the FT230XS is the source, it ONLY provides a clock when USB is connected!**

A small FSM divides the clock by 2 (12 MHz state clock) and nominally uses 4 states for one CPU clock cycle. The FSM has provisions to add either 1 or 3 "wait states" depending on the RDY1 and RDY3 signals, which are internally derived from the address decoder. The Q and E signals are driven onto QCLK and ECLK outputs on the CPLD, respectively.

```
-- CPU clock generation
-----------------------
-- HD6309 needs quadrature phase clock (here with 0WS)
--
--        ___________            __________               __________
--  Q ___/           \__________/          \_____________/
--
--
--              ___________             ___________             _______
--  E _________/           \___________/           \___________/
--
--
--                ___________             ___________             _______
--  D-E _________/           \___________/           \___________/
--
-- ST 0     1     3     2     0     1     3     2     0     1     3
--
```

```
--
-- (here with 1WS)
--
--        ___________            __________                    ___________
--  Q ___/           \__________/          \__________________/           \_
--
--
--              ___________             _________________             ______
--  E _________/           \___________/                 \___________/
--
--
--    ___________________________________        ____________________________
-- R1                   \\\\\\\\\\\\\\\\\\______//////////////////////
--
-- ST 0     1     3     2     0     1     3     6     2     0     1     3
--


--
-- (here with 3WS)
--
--        ___________            __________
--  Q ___/           \__________/          \______________________________
--
--
--              ___________             _____________________________
--  E _________/           \___________/                             \_____
--
--
--    ___________________________________        ____________________________
-- R3                   \\\\\\\\\\\\\\\\\\______/////////////////////////////
--
-- ST 0     1     3     2     0     1     3     4     5     6     2     0
--
--
-- States 4 and 5 and 6 are wait states for slow peripherals and are inserted
-- if READY3 is not asserted before end of state 3. READY3 is asserted during
-- accesses to the CIO and SCC (slow Zilog peripherals).
--
-- A shorter, 1 wait state, sequence is used if READY1 is not asserted before
-- end of state 3 (only the extra state 6 is inserted).
```

Peripheral I/O Space Decoding

The CPLD performs the address decoding of the H6309's address space. All peripherals are memory mapped, and all are mapped within the 0xE000 – 0xE1FF range.

0xE000 - 0xE00F	Zilog CIO
0xE010 - 0xE01F	Zilog SCC
0xE020 - 0xE02F	~~Epson Real Time Clock**~~ Soft I2C register
0xE030 - 0xE03F	SD Card Interface
0xE040 - 0xE040	SYSTEM CONFIG register
0xE050 - 0xE05F	IO CONFIG register
0xE060 - 0xE06F	VERSION register

RAM and ROM Mapping

RAM is always mapped from 0x0000 - 0xBFFF

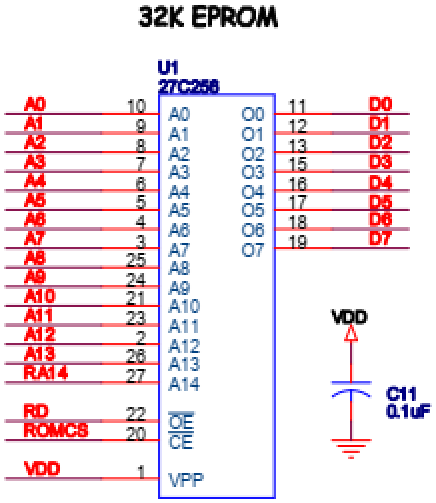
RAM is mapped from 0xC000 - 0xDFFF when ROMSEL = 0
ROM reads are mapped from 0xC000 - 0xDFFF when ROMSEL = 1

RAM is mapped from 0xE200 - 0xFFFF when ROMSEH = 0
ROM reads are mapped from 0xE200 - 0xFFFF when ROMSEH = 1

ROM writes ALWAYS map to the underlying RAM!

Flash EPROM Usage

A 27C256 EPROM (32KB) can be used without restrictions. A14 is controlled by CONF jumper at bootup, so it's possible to have two 16K images stored and boot from one or the other.



A 27C512 12V Flash (e.g. Winbond 27C512) can be used with the understanding that really only 32K of its 64K address space is usable. High voltage Flash memory devices with A15 connected to pin #1 (permanently tied high) result in only the UPPERMOST 32K of space in the Flash being available.

CPLD Special Function Registers

There are four SFR realized in the CPLD V1.4. They are:

0xE020	I2C Bit Bang Port
0xE040	SYSTEM CONFIG register
0xE050	IO CONFIG register
0xE060	VERSION register

The **I2C Port** is for communication with the Maxim DS3232M Realtime clock:

0xE020		I2C Bit Bang Port
bit 7	r/w	SDA drive (1 = float, 0 = sink)
bit 6	r/w	SCL drive (1 = float, 0 = sink)
bit 3	r	SDA monitor
bit 2	r	SCL monitor

all other bits are unused and return 0’s when read.

The **SYSTEM CONFIG register** has the following functionality:

0xE040		SYSTEM CONFIG register	
bit 4	r	CONF	CONF jumper staus (1 = open, 0 = shorted)
bit 3	r/w	ROMP27	Control A14 of 27C256 / 27C512 Flash device This takes the value of CONF at reset!
bit 2	r/w	ROMSEH	map ROM into $E200-$FFFF when 1, otherwise RAM This takes the value of 1 at reset
bit 1	r/w	ROMSEL	map ROM into $C000-$BFFF when 1, otherwise RAM This takes the value of 1 at reset
bit 0	r/w	ROMWS	add 1 WS to ROM accesses when 1, otherwise 0WS This takes the value of 1 at reset

all other bits are unused and return 0’s when read.

The **IO CONFIG register** has the following functionality:

0xE050		IO CONFIG register	
bit 7	r	SDSW	sense the SD card present switch (1 = present)
bit 6	r	SDBUSY	SPI interface is busy when 1
bit 5	r/w	SDCLK	select CLK for SPI (1 = XOSC, 0 = OSC/8)
bit 4	r/w	SDCS	control CS line to SD CARD
bit 3	r	PB	pushbutton input (1 = pressed)
bit 1	r/w	LED2	control LED #2 (1 = illuminate)
bit 0	r/w	LED1	control LED #1 (1 = illuminate)

bit 2 is unused and returns 0 when read.

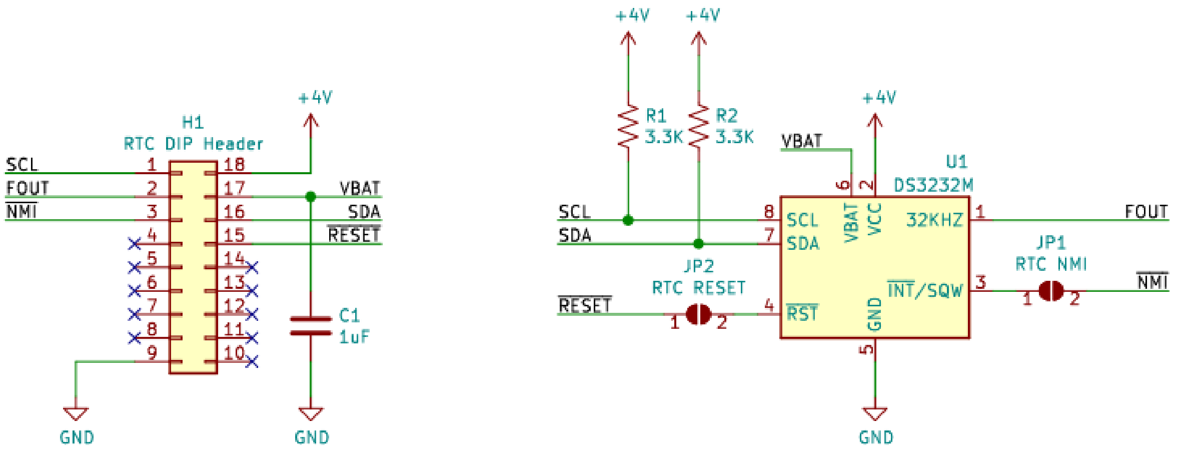
The read-only **VERSION register** has the following functionality:

0xE060	VERSION register	
bits 7:4	MSD	version most significant digit (CPLD 1.4 = '1')
bits 3:0	LSD	version least significant digit (CPLD 1.4 = '4')

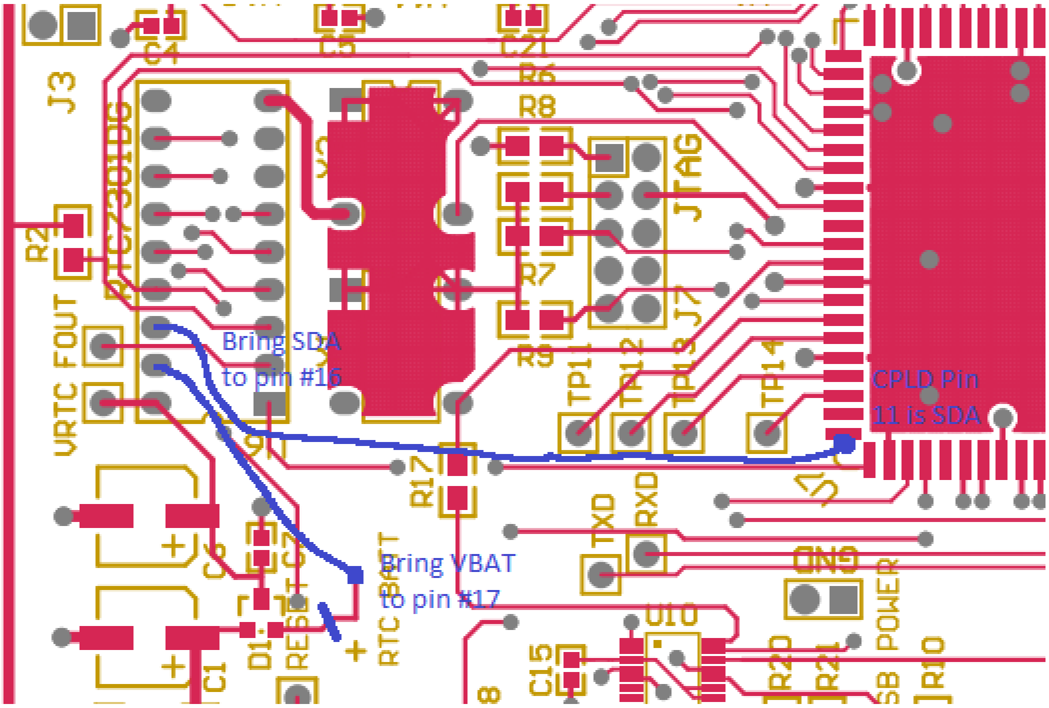
Real Time Clock Interface

The Epson RTC7301 Real Time Clock was originally chosen, because it was the cheapest RTC with built-in xtal. However, as of late 2019, the RTC7301DG is obsolete and very hard to find! There are other 4-bit wide devices from Epson, but they are not compatible and lack a lot of the features of the RTC7301DG. The preferred solution is a socket adapter and a bit-banged I2C interface to a Maxim DS3232M Realtime Clock with built-in MEMS resonator and nonvolatile SRAM.

An 18-pin DIP plug carrier can hold the RTC, pullups, and bypass cap. The Epson RTC has a common VBAT/VCC node, while the DS3232M has separate terminals for these two supplies - so the VBAT is seperated from VCC feed. The series diode in the VCC line is helpful, because the DS3232M has a 4.5V maximum supply voltage (!). The Epson RTC had pins 16 and 17 as no-connects, so these are re-purposed for SDA and VBAT, respectively: The RTCCS (pin #1) is re-purposed for SCK signal.



Some minor surgery is required to route the SDA and VBAT signals to the DIP carrier, and separate the VBAT from VCC. SDA is taken from CPLD pin 11 (a corner pin) because there is no other signal from CPLD to RTC to 'steal' for this.



Bring SDA to pin #16

CPLD Pin 11 is SDA

Bring VBAT to pin #17

Finally, the CPLD is revised to V1.4 remove the external 4 bit I/O port and replace with an internal port to control the open-drain SDA and SCL bidirectional lines.

Zilog Peripheral Interface

The Zilog Z8536 CIO and Z85C30 SCC are slow, and therefore need a **wait state** to transfer data within spec. Moreover, the write transfer (from CPU to CIO) takes place on the **falling edge of WR**. Therefore, the gate for WR cannot simply be E, but needs to be delayed a bit to allow the CPU to present the proper data onto the data bus.

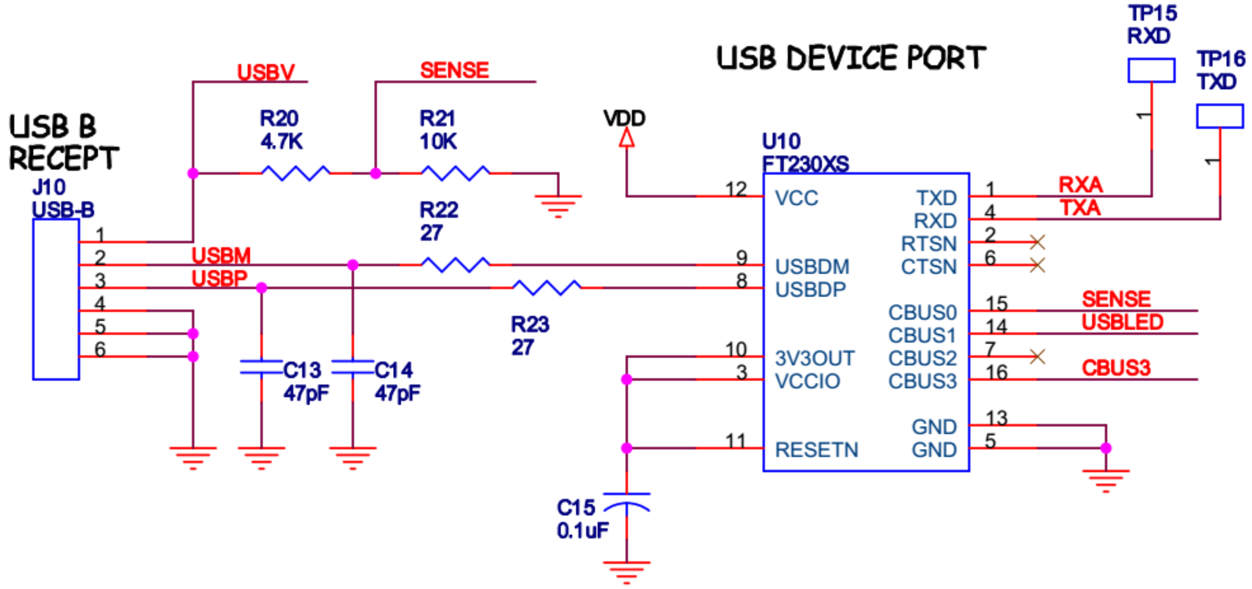
```
-- Read-Write signal generation
-------------------------------
-- ZRD is RW without any qualification
-- ZRW is RW qualified by last portion of e-clock (state 2)
-- both ZRD and ZWR are asseted simultaneously during RESET
zrd_n <= '0' when ( RESET = '0' or RW = '1' ) else '1';
zwr_n <= '0' when ( RESET = '0' or (RW = '0' and zwgate = '1') ) else '1';
```

SCC Baud Rate Clock Generation

Each of the UART channels of the SCC requires a 16x baud clock. SCC channel A is fixed at 115.2kbps baud rate on SCC channel A, so an "ACLK" signal of 1.843 MHz (16 x 115.2kbps) is required. SCC channel B is intened to allow standard baud rates, so a "BCLK" signal of 3.686 MHz is required. Both ACLK and BCLK are derived from 14.745 MHz oscillator Y2, by dividing by 8 and 4, respectively.

USB Interface

I used a FTDI FT230XS bridge chip between the SCC Port A and the host PC. This chip is very nice in that it requires only a small handful of external components (R/C on the USB lines and a core VREG bypass cap). It has on-chip EEPROM and has configurable "CBUS" pins to allow (re)mapping of their functions. Of the four LED, I've mapped one each to RX activity and TX activity. A third CBUS pin is used for voltage sensing, in the event that the USB 5V is NOT used to power the board.



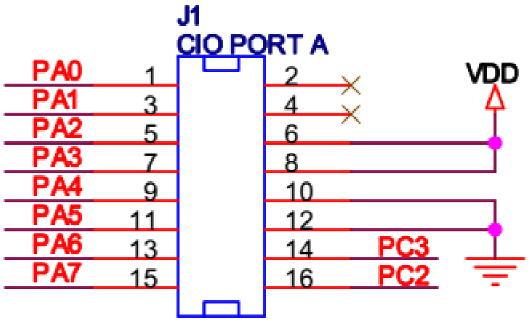
A ‘fresh’ FT230XS device requires configuration before it can be used correctly on the SBC. Use the FTDI FTPROG utility to configure it:

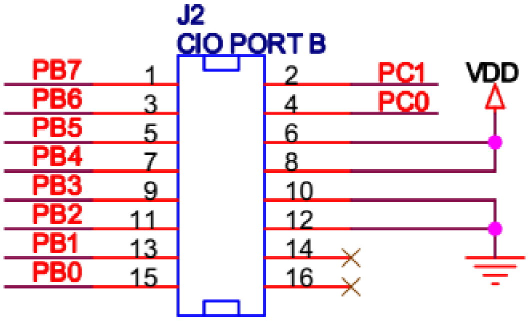
USB Config Descriptor
- bmAttributes = Self Powered
- MaxPower = 500mA

Hardware Specific
- CBUS Signals
  - C0 = VBUS_Sense
  - C1 = TX&RXLED
  - C2 = Drive_0
  - C3 = CLK24MHz
- IO Pins
  - CBUS Drive = 8mA

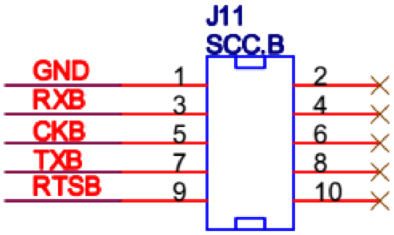
Input-Output Connectors

I brought the CIO GPIO pins pins out to dual-row headers along one edge of the PCB.



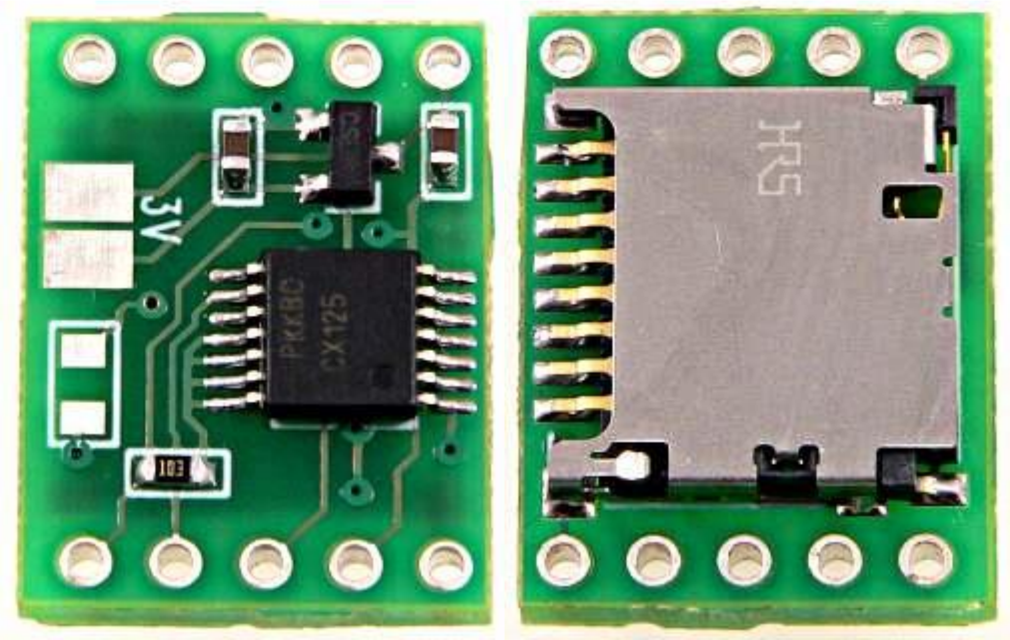


The SCC Channel B is also brought out to a dual-row connector - so far, this is useful for running the NoICE debugger.



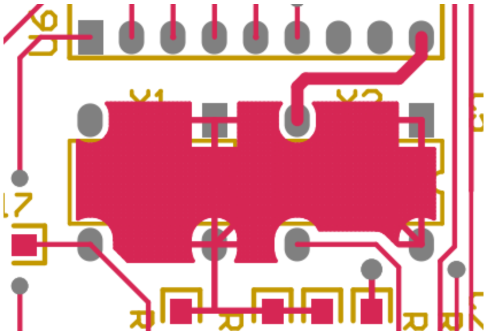
SDcard Interface

PJRC (maker of 8051 Paulmon) offers a nice little Micro SD card adapter - has level shifter (3.3V/5V) and vreg right on board! It's only $8 and so I ordered a few. Works great! The EPM7128 CPLD has (just enough) space for a simple SPI master, so the SDcard is accessed in SPI mode via a real hardware SPI.

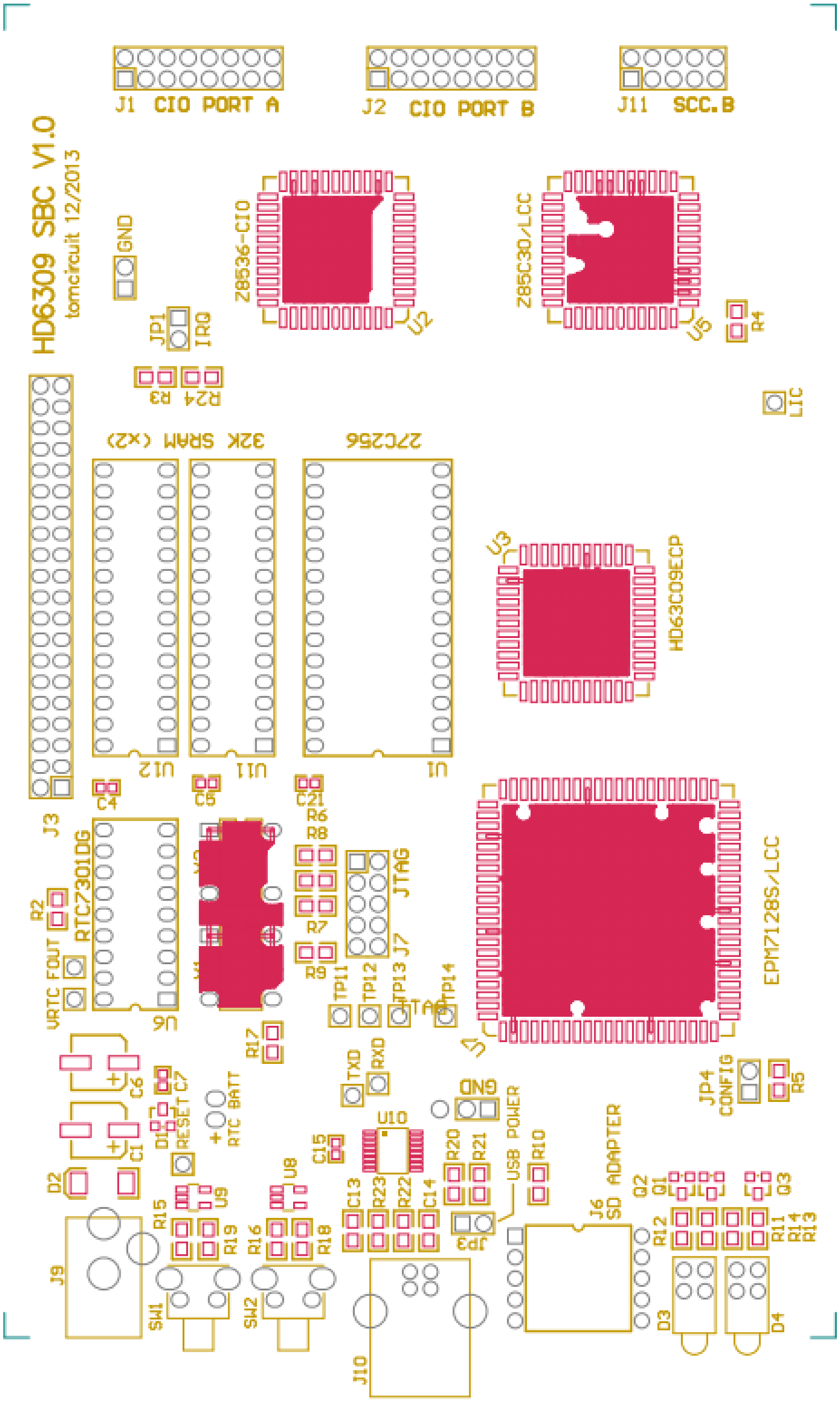


Bugs!

1) The first issue noted is that the two oscillators are a bit close together - so one has to be raised up with a socket.



2) Not really a bug, but a limitation. The provision for using the internal 24 MHz from the FT230XS works, but the catch is that it only works when USB is enumerated. So, it's best to include the 24 MHz oscillator for Y1.


HD6309 SBC V1.0
tomcircuit 12/2013
J1 CIO PORT A
J2 CIO PORT B
J11 SCC.B
GND
JP1
IRQ
Z8536-CIO
U2
Z85C30/LCC
U5
R4
R3
R24
LIC
32K SRAM (x2)
27C256
U12
U11
U1
U3
HD63C09ECP
J3
C4
C5
C21
R6
R8
R7
R9
JTAG
J7
EPM7128S/LCC
R2
RTC7301DG
VRTC
FOUT
U6
X1
X2
TP11
TP12
TP13
TP14
JTAG
U7
R17
TXD
RXD
GND
USB POWER
JP4
CONFIG
R5
C6
C7
C1
D1
RESET
+ RTC BATT
U10
C15
R20
R21
R10
D2
U9
U8
R15
R19
R16
R18
C13
R23
R22
C14
JP3
J6
SD ADAPTER
Q2
Q1
Q3
R12
R11
R14
R13
J9
SW1
SW2
J10
D3
D4

Conclusions

It was fun to work with this venerable architecture. I published it on Hackaday and got lots of hits. Once again I used the monitor from Dave Dunfield. It doesn't support all the 6309 opcodes, but it's still quite nice. I used William Astle's lwtools 6809/6309 assembler toolchain. I actually submitted an improvement to allow generation of HEX and S-record output formats.

- Using the CPLD for the glue logic is glorious! The Quartus II software from Altera is a dream, and the Terrasic USB blaster works great. It's so nice to be able to alter memory maps, wait states, etc. with just a few changes to the VHDL and an upload.

- The USB bridge chip I chose for this design is great! RS232 is dead, and the FT230XS device in SSOP is not that hard to solder down and provides a lot of flexibility. Power from the USB is very convenient, also.

- The old Dave Dunfield tools work really well, even if they have to be run in DOSbox to function :-(

Grant Searle did a bang-up job of porting 6809 Microsoft BASIC to his SBC, and so I leveraged off of that to bring it to the HD6309SBC. Wow! How nice!