

Parallax Propeller® based IBM 1130 emulator notes

As of July 16, 2012

Contents copyright © 2012 by Walter T. Moss crop. You may use these notes and the accompanying source code in any way but I will not be responsible for errors, omissions, or any losses (financial or other).

1. Concept:

- a. This project began in early 2010. For many years I have wanted to emulate an IBM 1130 computer, which was the first “real” computer that I used in high school. There are already emulators for the 1130 that run on PC’s and FPGA’s, but I was more interested in seeing if I could do the emulation on a microcontroller. After all, if you have billions of bytes of memory alone, how hard is it to emulate a machine with a 1MB disk?
- b. The 1130 that I learned on consisted of the following equipment:
 - i. A 16-bit hardwired CPU supporting all 1130 instructions (A variant of the 1130, called the 1800, had additional instructions that most other emulators support).
 - ii. 8K words (16K bytes) of parity core memory.
 - iii. A 1MB disk cartridge. It could take well over a second for a seek to complete.
 - iv. A 1442 card reader/punch; 300 cards read per minute; 50 cards punched per minute.
 - v. An 1132 line printer. 60 lines per minute. On a good day, downhill off a cliff.
 - vi. A modified Selectric typewriter mechanism for use as a console printer. 15 characters/second.
 - vii. A modified keypunch keyboard for console input.
 - viii. 16 “sense” switches and various buttons, used for loading data or directing program execution.
 - ix. Lots and lots of blinking lights.

2. Project goal:

- a. In high school, I had written a blackjack (21) game program where the computer acted as the dealer. The program was written in Fortran, with a few assembler subroutines. I learned a lot from this program... random number theory (had to shuffle the deck)... I/O (the “cards” were stored in a disk file)... and so on.
- b. The emulator needed to be able to play the blackjack game as closely as possible to the original.
- c. I have yet to actually get the game up and running on the emulator... but I have gotten several Fortran & assembler programs to compile and run.

3. Emulator status:

- a. CPU (hopefully) fully emulated. Some nasty stuff where the “index” registers are maintained in core memory was a big headache.
- b. Core memory fully emulated.

- c. Disk emulated on a 2M x 8 flash chip (Microchip SST25VF016B, available from Mouser for about \$3 each). There is a limit to how many sector rewrites can be performed to a disk before it has to be erased and reloaded, but that has not yet been an issue, and the design allows for an upgrade that will allow unlimited rewrites.
- d. A 2501 card reader in place of the 1442.
 - i. The 1442 generated an interrupt for EACH card column. That's 80 interrupts per card. The 2501 has just an "operation complete" interrupt. Much faster and easier to emulate.
 - ii. I have no need for emulating a card punch.
 - iii. The input comes from a file on the SD card.
 - iv. Due to all 8 cogs being in use, I can only open one file at a time. The card file is opened and copied to the top 16K of a 64K EEPROM (the same EEPROM used to boot). This does limit the input file to 16K.
- e. A 1403 printer in place of the 1132.
 - i. Like the 1442, the 1132 generated an interrupt for each character to be printed as the drum rotated. 48 interrupts per line vs. 2 interrupts. Much faster, much easier.
 - ii. Output to file on SD card.
- f. Console typewriter—RS-232 interface to a generic printer. I have an electronic typewriter that can act as such a printer (hacked into keyboard scan lines w/a propeller-based driver).
- g. Console keyboard—I have yet to get the electronic typewriter accepting keypresses (future project).
- h. Sense switches and buttons—interfaced via 74HC165 chips and a 128x64 LCD.
- i. Blinking lights—interfaced via 74HC595 chips. Run status, keyboard select status, interrupt status, and accumulator display.

Please note that items f through i are optional; in fact, only 6 cogs are needed if not driving the LCD and the switches/lights.

4. Setup and use:

- a. All spin source files are specified in *italics*.
- b. The flash chip must be initialized with *FlashInit*.
- c. The dms.dsk image on the SD card must be copied onto the flash chip with *DiskInit*.
- d. To run a job:
 - i. Edit 2501.txt to contain the "card deck".
 - ii. Run *Run_1130*.
 - iii. When the job is complete, the 1403.txt file on the SD card needs to be closed. The emulator currently looks for the closure of one of the sense switches; this can easily be changed to a tact switch or whatever you like.
 - iv. As mentioned above, the flash chip will eventually fill up with rewritten sectors. Your mileage may vary, but I'd expect at least 10 runs before this happens.

1. You can use *FlashStatus* to determine how many empty sectors remain for use.
 2. Repeat steps a-c to restore the flash chip for another round.
- e. IBM 1130 documentation (and PC-based emulator) available at www.IBM1130.org.
5. Included files.
- a. I have drawn (heavily) from the propeller libraries and OBEX. It has been my intent to always give credit where credit is due. In some cases I've modified the objects for my own purposes and in the process may have accidentally removed the original credits and/or author information.
 - b. Some of these files include schematics. Only the schematic in Schematic.spin should be trusted at this time.

74HC165_74HC595	Initializer for and interface to blinking lights and switches
74HC165_74HC595_Graphics	Initializer for and interface to LCD
74HC165_74HC595_Graphics_Saver	Saves 74HC165_74HC595_Graphics PASM to EEPROM
74HC165_74HC595_IO	Wrapper around 74HC165_74HC595 (probably redundant at this point)
74HC165_74HC595_Saver	Saves 74HC165_74HC595 PASM to EEPROM
1130_CPU	Emulates the 1130's CPU
1130_XIO	Emulates the 1130's I/O subsystem (takes its name from the XIO instruction)
1130_XIO_Saver	Saves 1130_XIO PASM to EEPROM
1403_XIO	Emulates the 1403 printer (Note: doesn't use cog loader at this time)
2501_XIO	Emulates the 2501 card reader
2501_XIO_Saver	Saves 2501_XIO to EEPROM
Cog_Loader	From OBEX; used for retrieving various cogs
Common_XIO	XIO constants
Disk_Constants	Disk constants
Disk_XIO	Emulates the disk subsystem
DiskInit	Copies dms.dsk (disk image) to flash
DiskStatus	Displays sectors remaining on flash
EEPROM_Constants	EEPROM constants
EEPROM_IO	From OBEX; used for various EEPROM activities
FlashInit	Erases and reinitializes flash
FlashStatus	Displays flash usage count
fsrw	From OBEX; SD card I/O
Fsrw_small	From OBEX; SD card I/O (limited)
IO_Pin_Constants	Pin constants
Mask_Constants	Mask constants
Memory_Constants	Memory constants
PasDebug	From OBEX; PASM debugger
Run_1130	Initializes, boots, and runs the emulated 1130
Safe_spi	From OBEX; flash I/O
Schematic	Schematic of emulator circuit

sdspi	From OBEX; SD card I/O
Simple_Serial	From OBEX; serial I/O
SPI_Asm_Disk_Util	From OBEX; Used by utilities for disk I/O