Computer History Museum

IBM 1620 Jr.
Project Status
Report #5 (12/31/2018)

This is the fifth status report on the IBM 1620 Jr. project. An overview of the project is detailed in *IBM 1620 Jr. Project Description, Version 1.3 (1/12/2018)*. The project is structured as several sub-projects, each with a set of volunteers working on it. Note that this is only a summary of the work done to date. Details are available in email & data files.

General (Team)

A major goal for the team was completing the necessary work on the IBM 1620 Jr. (front panel, console typewriter, simulator, and software library) to exhibit at VCF West XIII in early August 2018. The plan was to demonstrate the historic BBC Baseball Simulation program and other programs that would highlight an operational console typewriter. The goal was reached and the IBM 1620 Jr. received Best in Show and 2nd place in the Restorations & Replicas category. A separate writeup – *IBM 1620 Jr. VCF West XIII Report* (9/1/2018) – details the show.

The development of the console typewriter and the associated software changes to the simulator were done independently and 300 miles apart. The first time the IBM 1620 Jr. and the console typewriter were physically together was July 6th, one month before VCF. Not all firmware features were implemented and a few problems were discovered, but for the most part it worked and the Powers of 2 demo program was successfully run.

A GitHub repository for the IBM 1620 was set up (https://github.com/IBM-1620) and a few of the console typewriter interface documents were added. It is the team's intention to put all of the IBM 1620 Jr. documents and source code in the repository. Other files related to the IBM 1620 and the original restoration project will also be added.

In September, the team welcomed a new member, John Bohn, a retired software developer living in Michigan. John learned Fortran and assembly language programming on the IBM 1620 as a senior in high school. He is currently writing a test program for the IBM 1620's floating point instructions.

Work on the project slowed down after VCF for several team members due to family commitments. This is a part-time, all-volunteer project, and yet the amount of progress in the past 2 years has been remarkable. Things are expected to pick up with the new year.

The team is planning to again exhibit at VCF West in August. The new addition this year will be an emulated Card Read-Punch device. There is a possibility that the IBM 1620 Jr. will be used during the annual Baseball Hack-A-Day in March. Completing a number of "loose ends" in several sub-projects is also planned for this year.

Front Panel (Steve Casner, Joe Fredrick)

The front panel sub-project consists of all physical and electrical work done on the spare IBM 1620 front panel, the Raspberry Pi 3 (RPi), and the interface circuitry connecting the two.

The majority of work on the front panel was completed early in the project and it has been operating without problems for over 18 months. A few miscellaneous items were done for VCF West XIII – a temporary

plexiglass back was added to the case, a power distribution box was added, and an I/O distribution panel was added.

Still to be done - install an amplified speaker, add a locking latch and carrying handles to the cabinet, label currently unused lights and switches for missing Level F features, paint or skin the cabinet to match IBM 1620 textured gray, add a permanent back cover, and a more sturdy name sign.

The end of September an IBM 1620 Model 1 Level F/G front panel was available on eBay. It would have been a perfect replacement for the current IBM 1620 Jr. front panel, eliminating the need to add labels for the additional lights and switches. [Note that even with the additional labelling, the current IBM 1620 Jr. front panel won't exactly match a Level F machine's panel because some lights will be in the wrong place and the new labels will look a little different.] Unfortunately, we were outbid.

Console Typewriter (Joe Fredrick, Steve Casner, Dave Babcock)

The console typewriter sub-project consists of all physical and electrical work done on the console I/O device.

The majority of work done prior to VCF was on the console typewriter. This includes:

1. The design of the typewriter interface board was finalized and the board laid out, fabricated, and assembled.

2. An illustrated guide, *IBM/Lexmark Wheelwriter 1000 Teleprinter Adaptation*, was written.

3. The project's two Wheelwriters (a primary and a spare) were converted to IBM 1620 Jr. console typewriters following the guide. Adapting a Wheelwriter costs approximately $100 in parts and takes about 2 hours.

4. The firmware for the interface board was revised and enhanced to implement the wire protocol documented in *IBM 1620 Jr. Console Typewriter Protocol, Version 1.7 (7/8/2018)*.

5. A standalone test program was written in Java and used to exercise the IBM 1620 Jr. console typewriters via the wire protocol. [A subset of the test functionality will be added to the simulator to make future console typewriter maintenance easier.]

At VCF the team was given the name of a company which makes custom keycaps for Wheelwriter typewriters. Following that lead, a realistic set of IBM 1620 console typewriter keycaps was designed and fabricated. The two project typewriters were retrofitted with the new keycaps and the interface board firmware is being modified to match.

It was the project's goal that Jr.'s console typewriter would mimic the real IBM 1620 console typewriter in look, feel, functionality, and typed output as closely as possible. Now that the keyboard layout has been changed, the operation of the typewriter better matches the IBM 1620. One exception is the setting/clearing of margins/tabs which was done mechanically on the IBM 1620 console typewriter and via special key sequences on the Wheelwriter.

In general, the Wheelwriter-based console typewriter keeps up with an authentic 10 cps print speed. However, due to the overprinting required for some IBM 1620-specific characters, lines which contain a large number of special symbols, do not meet the average rate of 10 cps.

Still to be done – update the firmware for the new key positions, complete the functionality of the firmware, fix a few bugs, and develop a standalone typewriter version of the firmware for off-line typewriter

maintenance. A related item, but not formally part of the IBM 1620 Jr. project, is to develop an "ASCII" version of the firmware. This has been requested by some hobbyists with other vintage computers in need of a teleprinter device.

Card Read-Punch Device (Dave Babcock)

The card read-punch device sub-project consists of emulating a punched card unit so that IBM 1620 programs can be "read" and "punched". The majority of the IBM 1620 Software Library that the Computer History Museum has is card based.

The current plan is that it will be a scale model of the IBM 1622 Card Read / Punch with a touch screen in front. The screen will usually show an [animated] image of the front of the 1622 with its control buttons, card hoppers, and card stackers. There will be a USB port on the left side for "blank cards" to punch and one on the right side for already "punched cards" to read. When USB memory sticks are inserted in either or both sides, their content will be examined.

On the read side, if there is only one "*.crd" file, it will be opened, the number of cards counted, and an image of an appropriate number of cards will appear in the read hopper. If the memory stick contains more than one "*.crd" file, the touch screen will list them for the user to select from. If there are no "*.crd" files or no memory stick is present, then a READ CHECK will occur if the IBM 1620 Jr. attempts to read a card. However, if a "*.crd" file is open, then RNCD and RACD instructions will read "cards" from the file and animate them moving from the hopper to the correct stacker, with appropriate sound effects.

On the punch side, a new "*.crd" file will be created and opened on the memory stick for writing. The touch screen will show a number of "blank" cards in the punch hopper. As the IBM 1620 Jr. executes WNCD and WACD instructions, the touch screen will show cards moving from the hopper to the correct stacker, with appropriate sound effects. No memory stick would trigger a WRITE CHECK when attempting to punch cards.

Pressing the 1622 control "buttons" on the touch screen will function as expected.

The emulated card read / punch device will contain its own microcomputer and be connected with the IBM 1620 Jr. via USB and a protocol similar to the console typewriter protocol. The entire IBM 1620 Software Library will be stored on a USB memory stick or in the emulated card read / punch with a means for selecting which card deck to read via the touch panel.

This sub-project is just getting started. Photographs and measurements of an IBM 1402 card reader / punch have been made. The IBM 1622 card read / punch was [almost] physically identical to the IBM 1402. The only difference was the read-side card hopper – the read hopper on the IBM 1622 was the same as its punch hopper, whereas on the 1402 it was much longer and at a shallower angle.

Next steps – define the protocol, add card I/O support to the IBM 1620 Jr. simulator, build a 3D model of the IBM 1622, record the sounds of the museum's IBM 1402, build a prototype of the device, develop firmware for the microcomputer, test, refine, repeat.

Simulator (Dave Babcock, Steve Casner, John Bohn)

The simulator sub-project consists of developing a new IBM 1620 cycle-level simulator, coded in C, that interfaces with the front panel hardware. It will simulate an IBM 1620 Model 1 Level F punched card system with 60,000 digits of memory and the automatic divide, indirect addressing, MF/TNF/TNS, and floating-point processor options.

Support for the remaining core instructions and the console typewriter I/O instructions was added to the simulator for VCF. Next to be implemented is indirect addressing and the divide, MF/TNF/TNS, floating-point, and punched card I/O instructions. The software library contains the official IBM diagnostic programs for all of this except for the floating-point and punched card I/O instructions. Testing the five punched card I/O instructions is simple and straight-forward, but testing floating-point is very complex.

A robust program which tests the eight floating-point instructions (FADD, FSUB, FMUL, FDIV, FSL, FSR, TFL, and BTFL) is under development. Written in SPS assembly language, it will exercise all instructions and conditions with short and long, valid and invalid data. It verifies the numeric result of floating-point arithmetic instructions by the use of equivalent integer arithmetic. All checks with valid data are now coded and tested using the SIMH IBM 1620 simulator. The program will be used to validate the IBM 1620 Jr. simulator's implementation of the floating-point instructions.

To support the development of the floating-point instruction test program, possible use by the Baseball Hack-A-Day team, and future use by the CHM Education department, a portable SPS cross-assembler (SPS1620) was written. The assembler is coded in Java and runs on Windows, Mac, and Linux systems. It supports all of the features of the IBM 1620 Model 1 & 2 systems and standard peripherals producing object code files that can run on the IBM 1620 Jr., real IBM 1620 systems, the SIMH IBM 1620 simulator, and the [possible] future IBM 1620 replica. It also generates a symbol cross-reference table in addition to a full source and object code listing.

Several aspects of the IBM 1620 SPS assembler were missing from the manuals, ambiguous, or incorrect. An SPS test program was written to explore several dozen corner cases of the assembler and was run thru the IBM 1620 SPS card assembler. [The test program will also be run thru the paper tape and disk versions of the IBM 1620 SPS assembler to see if they do anything different.] The results were very surprising. A few items produce reasonable results and are considered omissions from the IBM manuals. Most of the items are error conditions, that is things that violate the definition of the SPS assembly language. Some of these produce no error message and most generate unexpected binary output. Several cases crash the assembler. The test program and all results are being documented.

Still to be done – implement the remaining instructions, add machine sounds, and add support for displaying console terminal output on an external HDMI monitor.

Software Library (Dave Babcock)

The software library sub-project consists of converting all of the museum's IBM 1620 software collection to a format usable by the IBM 1620 Jr., assembling the associated documentation, and testing the programs on the IBM 1620 Jr.

Two demo programs – BBC Baseball Simulation and Tic-Tac-Toe – were extracted from the software library and run on the IBM 1620 Jr. for VCF. Since VCF, two versions of the card-based IBM 1620 SPS assembler were extracted and run successfully on the SIMH IBM 1620 simulator to investigate corner cases. The complete set of IBM 1620 Monitor 1 installation programs was also extracted and run, but most failed due to what might be a disk I/O implementation problem in SIMH. This will be investigated and fixed as the SPS II-D disk assembler needs to be tested for how it handles the SPS corner cases. To test the paper tape SPS assembler, a utility program needs to be written to convert punched card images to paper tape format. Finally, the source code of several SPS programs were extracted from the library and successfully run through SPS1620 to validate its operation.

The bulk of the software library still needs to be extracted, post-processed, organized, combined with their documentation, and tested.

Operations Guide (Dave Babcock)

The operations guide sub-project consists of writing a brief document on the operation of the IBM 1620 Jr.

Notes for the Operations Guide are being kept as the project is being developed.

Website (Dave Babcock, Team)

The website sub-project consists of selecting, configuring, and populating a dedicated website on the IBM 1620 which will document the IBM 1620, the CHM IBM 1620 restoration project, and the IBM 1620 Jr. project.

An IBM 1620 GitHub repository has been created for storing project files.

Development of the website itself has not started.