Manchester Baby Simulator
User Guide

By David Sharp
The University of Warwick
11th January 2001

Freddie Williams and Tom Kilburn in front of the Baby or one of it's close descendants.

Historical background

The Manchester Baby (also known as the Small Scale Experimental Machine) was the first working stored-program computer. That is a computer which stores and executes its program from an electronic storage device rather than for example reading it from a punched tape. It was built at the University of Manchester by Tom Kilburn, Freddie Williams and Geoff Tootill, successfully running its first program on June 21st 1948.

All three of the team had previously worked with radar technology at the Telecommunications Research Establishment (TRE) in Malvern and this had led the team to working with cathode ray tubes (CRT). Williams had visited the ENIAC project at the University of Pennsylvania in 1946 also reading the EDVAC report. He took away the stored-program concept as well as the need for a large high-speed electronic storage device, for which he saw CRTs as a prime candidate.

The primary purpose of the Baby was to test the ideas of Williams and Kilburn of using a CRT as a data storage device for a computer. Although they could store data on a CRT for long periods of time, this wasn't evidence of its suitability for a computer where the data was constantly changing at high speed and hence they had to build a computer to test it. To this end, Kilburn designed, "the smallest computer which was a true computer (that is a stored program computer) which [he] could devise"[1].

The Baby was built incrementally with many new features being added after June 1948, which have somewhat muddied the waters of history as to exactly the original features. The Baby led to the development of the intermediary version of the Manchester Mark 1, the final Manchester Mark 1 and the Ferranti Mark 1 computers. Patents filed by Williams and Kilburn led to what became known as the 'Williams Tube' which was licensed by IBM for their 700 series in the early 1950s. Later innovations by the team (who were soon joined by others) included adding an index register and then a secondary store in a magnetic drum. However it was after that event on 21st June 1948, as Williams later recalled that, "nothing was ever the same again"[2].

Introduction to the Baby's hardware

The Baby consisted of three cathode-ray-tube stores; the control, the accumulator and the store, only one of which was displayed at any one time on the display monitor. The store is made up of 32 rows (lines), each of 32 bits, notably displayed with the least significant bit on the **left**. The store is akin to what we might think of as memory and is accessed by line number.

The control contains two values, the control instruction (CI) and the present instruction (PI). The control instruction contains the line number of the instruction executed **previously**. The present instruction contains the line representing the instruction

[1] Kilburn, Tom, "From Cathode Ray Tube to Ferranti Mark 1", The Bulletin of the Computer Conservation Society, Volume 1, Number 2, Autumn 1990.
[2] Campbell-Kelly, Martin and Aspray, William, "Computer – A history of the information machine", BasicsBook, 1996, p.100.

currently being executed. It is important to note that the present instruction is 'transient' and only exists (and hence is only displayed) when the machine is running. The control instruction and present instruction (when in existence) are repeated on every line of the display.

The accumulator contains only one value, the accumulator itself which is the location for the result of arithmetic instructions, in a similar way to the accumulator on a modern microprocessor.

A store line contains 32 bits but only some of those are used. Bits 0-4 represent the operand line i.e. the number of the line that the instruction will operate on when executed. Bits 13-15 represent the function number (what we would know as the instruction opcode).

The available instructions were as follows:

Function Number	Modern name	Operation
0	JMP	Copy content of the specified line into the CI
1	JRP	Add the content of the specified line into the CI
2	LDN	Copy the content of the specified line, negated, into the accumulator.
3	STO	Copy the content of the accumulator to the specified store line.
4	SUB	Subtract the content of the specified line from the accumulator.
5	SUB	Exactly the same as for function number 4.
6	CMP	If the accumulator is less than 0 increment the CI.
7	STP	Halt the Baby and light the 'stop lamp'.

How to use

A simulation of the Baby would not be complete (and sadly many are not) without simulating the switch panels that were used to program the machine. The interface described initially in this user guide clearly has no historical basis and is present so that the user can quickly and easily use and program the simulator. The simulation of the 'switch panel' section later in this guide is the really interesting part of the simulation and gives the Baby its character. This user guide describes how to load and run programs and how to use the switch panel. If any extra help is needed, brief tool tips will pop up when the mouse pointer is over a feature.

Getting Started

The program has been written to run on Java v1.2 or later. Run the program by entering

```
java Baby
```

at the command line. The display window will appear, as shown below in Figure 1. A default program will be loaded automatically, see the end of the manual for details of other programs.

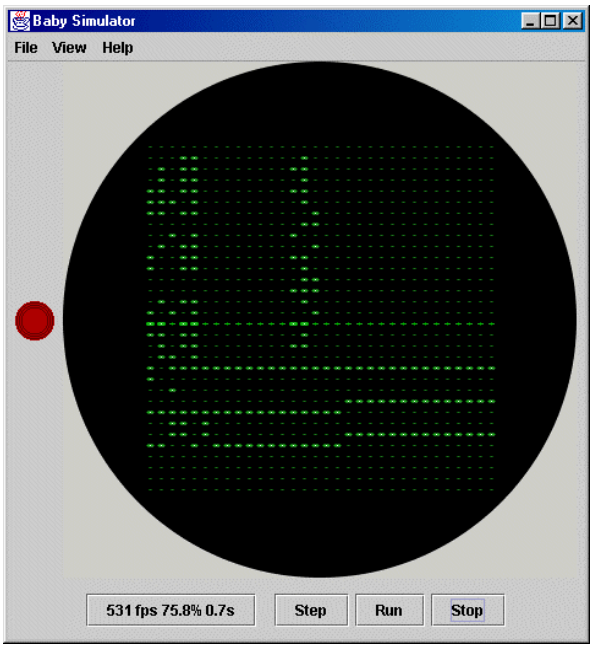

Figure 1: Picture of the simulator's display window

Loading a program

Historically programs were entered into the Baby's store (memory) via a 'typewriter', although this functionality is provided (see section 'Typewriter') there are easier ways to load programs on the simulator. Programs are available in two formats:

Assembly – a modern-style assembly language equivalent to the machine instructions of the baby, filename extension '.asm'. Single line comments are allowed prefixed by a semi-colon.
Snapshot – a representation of each bit of the Baby's store.

To load a program into the Baby, click on the 'File' menu and select the 'Load snapshot/assembly' item. A file selection window will open and you should select an assembly file (.asm) or snapshot (.snp) file provided and click 'Open'. The file type will be automatically detected and the program loaded into the display.

See the section 'Programs provided' for a list of programs for the Baby.

Saving a program

The state of the store can be saved to a snapshot or assembly file at any time by opening the 'File' menu and selecting the 'Save snapshot' or 'Save assembly' menu item. You can then enter a filename and select 'Save' to save the state or assembly.

Running the program

For authenticity programs should be run using the switch panel (see section 'Switch Panel') but for quick demonstration purposes, the buttons at the top of the display window, 'Step', 'Run' and 'Stop' can be used to control the Baby.

'Step' increments the control instruction, reads the line of the store denoted by the value of the control instruction and executes that instruction.
'Run' repeatedly performs the same as 'Step'.
'Stop' halts the running machine.

Unless stopped by the user, the Baby will keep on executing instructions until a STP instruction (function number 7) is executed.

Note, running the Baby simulation is very intensive and it is not recommended that you attempt to multitask too many other applications as it may affect performance of the display update.

Viewing the different CRTs

Only one of the three cathode-ray-tube stores could be displayed on the display monitor (the grid of green dots on a black background) at any one time, you can select which one should be displayed by choosing 'Store', 'Control' or 'Accumulator' from the 'View' menu.

Stop lamp

To the left of the display monitor is the Stop lamp. When a STP instruction is executed, the Baby stops executing instructions (so that the results of the calculation can be read off the display) and the stop lamp is lit.

No more instructions can be executed until the stop lamp has been extinguished. To clear the stop lamp, simply press the KC button on the switch panel (see later for details) or load another program.

Speed and timing

At the bottom of the display window is some information about the speed of the simulation. The real Baby ran at around 700 instructions per second, the number of instructions executed per second on the simulation is labeled as 'fps' (frames per second). The percentage speed of the real Baby that the simulator achieves is next to that (i.e. 100% would mean the simulator is running at the correct speed, 50% would mean it's running at half speed). To the right of this is the number of seconds of simulated time that have elapsed, to reset this to zero click on it.

The simulator attempts to adjust its speed so that it runs at around 100% speed though this is not possible to do perfectly because of the sporadic and unpredictable variations in processor usage.

Disassembling the store

To help understand the program and data in the store (reading binary in the reverse order to that which you're used to is not easy!) a disassembly feature is provided. To view a disassembly of the store to the modern assembly representation, select 'Disassembler' from the 'View' menu.

The disassembly window should be displayed; at the top are the values of the CI, PI and accumulator followed by every line of the store. A line might be:

```
12  SUB 23   ;=202540951
```

This means that the instruction at line 12 was to 'subtract the value at line 23 from the accumulator. It also means that if line 12 were to be loaded into the accumulator it would have the value 202540951.

The information in the disassembly displayed can be updated from the store by clicking the 'Load from store' button at the top of the disassembler window. The altered disassembly can then be saved back to the store by clicking the 'Save to store' button. This provides a simple Integrated Development Environment.

In disassembling the disassembler attempts to guess whether each line is program or data and will disassemble data to the assembler's pseudo-instruction, 'NUM', so that the correct value is restored to the store when it is updated. For programs which make use of the surplus bits in an instruction line for graphical purposes, this may not work correctly and will result in them being disassembled as data rather than instructions.

Switch Panel

To display the switch panel, select the 'Switch Panel' item from the 'View' menu.

Note: For realistic layout of the display, the display window should be positioned directly above the switch panel. A screen resolution of 1280 x 1024 pixels (or greater) has been found to be required to display the windows in this configuration.

Typewriter

At the top of the panel is the typewriter, this consists of 40 maroon-coloured buttons, numbered 0 to 39. These are used to affect the individual bits of a line in the store. Only buttons 0-31 are actually connected.

Staticisor

This is made up of the L-staticisor or Line switches are the 5 switches labeled 0 to 4. These are used to select a given line number. There is also the F-staticisor or Function switches which are labeled 13 to 15. These are used to select a given function number.

The final part of the staticisor is the automatic/manual selection switch labeled 'STAT'. If automatic is selected (the switch is down) then instructions are taken from the store and executed. If manual is selected (the switch is up) then the instruction encoded by the

function number of the F-staticisor switches is executed, acting on the line encoded in the L-staticisor switches.

For example, when the switches are set up as shown in figure 2, the Baby will be set up ready to execute the instruction STO 11 (function number 3).

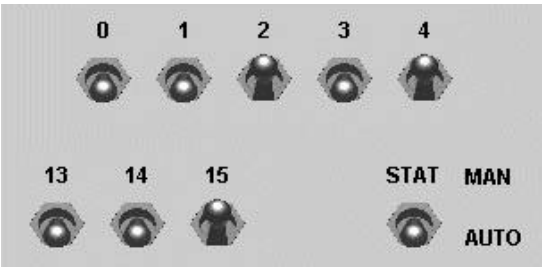

Figure 2: Example of setting up the staticisor switches.

Monitor Selector

The three red buttons labeled 'C', 'A' and 'So' are the monitor select switches. That is S subscript-zero, representing store zero with the intention of multiple stores being added later. Clicking on the selected switch displays the appropriate CRT-store on the display monitor.

CS (Completion Signal) Switch

This is the 'completion signal' switch (later known as the 'prepulse') where the signal which starts a new instruction was thought of as completing the previous instruction. When switched down the Baby repeatedly executes instructions, from either the store or from the encoded staticisor switches (depending on whether the staticisor is in automatic or manual mode). Flicking this switch to down is the equivalent of pressing the 'Run' button in the simple modern user interface (and in fact pressing 'Run' makes this switch flick down) and flicking it to up, the same as the 'Stop' button.

KC (Key Completion) Switch

This is effectively a single-step key and generates a single prepulse to execute a single instruction. Which instruction is run depends on the staticisor automatic/manual switch in the same way as the CS switch did. Flicking this switch down is equivalent to pressing the 'Step' button in the modern user interface. The KC switch will clear the Stop lamp if it is lit (this was the only way to reset it).

Clearing Keys

There are several keys that are used to clear different parts of the storage. However, some of the switches were not connected in the original Baby:
KLC – clears the line specified in the L-staticisor switches.
KSC – clears the store.
KAC – clears the accumulator.
KCC – clears the control, actually clears the CI, PI and accumulator.

KBC, KEC and KMC are all unconnected and were present for features to be added later.

Erase/Write Switch

This switch selects the affect that the typewriter should have on the store when its keys are pressed. If the switch is flicked up (to erase) then when a typewriter key is pressed, the corresponding bit in the line denoted by the L-staticisor is cleared to 0. On the other hand if this were set to write then the corresponding bit would be set to 1.

Programs provided

diffeqt.asm	Magnus Olsson's graph plotter for the solution to a difference equation.
flash.asm	By Ken Turner, one of the replica building team.
hcf.asm	Highest Common Factor by Geoff Tootill. The two numbers are 3142 and 2178, and they are co-prime.
hfr989.asm	Tom Kilburn's Highest Common factor for 989. Answer is 43. (This is loaded as default when the simulator is started.)
intdiv.snp	Brendan Campbell's integer division, UK schools prizewinning competition entry.
longdiv2.snp	Turing's long division.
medclock.snp	John Deane's Mediaeval clock, displays the hour as the number of zeroes counted from the left in location 30. The fraction of the hour is displayed in location 31 where each 0 from the left indicates 1/32 hour (about 2 minutes).
nightmare.snp	"Tom Kilburn's nightmare", a large baby chasing Tom Kilburn ad infinitum, a competition entry, author unknown.
noodletimer.snp	Yasuaki Watanabe's 3 minute noodle timer, the competition winner.
primegen.asm	Bas Wijnen's prime number generator, when the program stops, press KC to skip the stop and then re-run to get the next prime. A competition runner up.
slide9.snp	The first "modern" program written for the SSEM, by Keith Wood, one of the replica building team.
virpet.asm	Virtual Pet by Achut Reddy, a competition entry. See virpet.txt for details of operation.