PlasMa

aka Phil's lights and switches Machine

Mini-Mainframe Simulator Project

PlasMaSim Simulator Manual

Phil Tipping

www.philizound.co.uk

Table of Contents

1. Introduction

The PlasMa machine is a free-standing unit with real lights and switches. Its aim is to provide a hands-on programming platform and rekindle the ‘lights and switches magic’ from the mainframe computer era. It is described in a separate Machine Manual.

This document is the Simulation Manual, which describes a program for simulating the Plasma machine[1]. It can be used for assessing the capabilities before building a machine yourself, or for developing programs[2].

The full version of the program is a single executable file named PlasMaSim3.exe which supports the 3 microcodes (instruction sets) used on the PlasMa machine, namely Toy-A, Toy-B and Advanced.

Limited versions are also available which only run subsets of these; PlasMaSim1.exe can only support microcode 1 (Toy-A), and PlasMaSim2.exe can only support microcodes 1 and 2 (Toy-A and Toy-B).

2. Acknowledgements

The Toy instruction set is used by kind permission of Robert Sedgewick and Kevin Wayne at Princeton university, and is described in their book 'Computer Science'. More details at:

https://introcs.cs.princeton.edu/java/home and https://introcs.cs.princeton.edu/java/60machine

Coursera course site: https://www.coursera.org/learn/cs-algorithms-theory-machines

Thanks also to Adrian Rawson for suggestions[3] and testing.

3. Documentation

These following documents are downloadable from the philizound.co.uk website or by contacting me directly (email address is at the bottom of the website page).

- PlasMa Machine Manual
- PlasMa Instruction Set - Toy-A
- PlasMa Instruction Set - Toy-B
- PlasMa Instruction Set - Advanced
- PlasMaSim Simulator Manual
- Plasm Assembler Manual

4. Installation and Running

The program runs under Microsoft Windows on a PC. There is no 'installation' as such; just move the relevant PlasMaSim executable file into a folder of your choice and run it from there. The program loads and saves various files in this folder, so the 'Program Files' folder is not recommended as this is usually write protected. One option is to create a PlasMa folder under Documents, and move the file there.

Running the program depends on your version of Windows, but typically you can either double-click the file, or right-click the file and 'send to desktop' to create an icon for future use.

---

1 The machine itself is a simulator, so is this a simulator of a simulator (sim^2 ?)... or should the term be emulators? Let me know which is correct.

2 Although developing *every* program on the simulator takes some of the fun out of PlasMa's hands-on features; the simulator user-interface has been left intentionally simplistic to discourage this.

3 After the initial 'barking mad' reaction died down!

5. Start up

When started, the program creates a text-based console window with a size of 124 x 43 characters. If it looks radically different to the above screenshot, confirm your PC can support this minimum size then try resizing the window manually as follows:

- Click the icon at the top-left of the console window and select Properties.
- Click the Layout tab.
- Set the following values:
  - Screen Buffer Size: Width=124,, Height=43
  - Window Size: Width=124, Height=43
  - Window Position: Left=0, Top=0 (or as required)
- Click OK to complete the operation.

The display shows most of the lights and switches corresponding to the real machine's front panel. The boxes on the left represent the timer and tty/oper screens, and the area below the registers is for the mag tape spool lights (only shown when active in the Toy-B and Advanced emulations).

When the machine is stopped, values for each register are displayed in hex and decimal (unsigned and signed), and the opcode's description is shown alongside the instruction register IR.[4]

The lower display shows the program title comprising version, microcodes available and which one is loaded, followed by a list of command keys with brief descriptions; see Commands for full descriptions.

The yellow line just above the title is used for prompts and status messages. If a message is displayed, all commands are disabled until it is acknowledged by pressing the Esc key.

6. Closing down

Press the Esc key when there is no outstanding prompt; the simulator will then prompt "Quit y/n?". Press 'y' (lower-case) to close the simulator. Any other key (including Esc) is treated as 'n'.

7. Commands

All commands use single key presses; upper and lower-case letters are treated differently. Most of the commands correspond to the control switches on the real machine, so refer to the PlasMa Machine Manual (or watch the PlasMa videos on Youtube) for more details.

The Machine Manual also contains full lists of instructions and I/O codes which are needed to write PlasMa programs. Some I/O functions may not be fully implemented on the simulator.

Commands shown with a ***** prefix are only available on the simulator, and have no equivalence on the real machine.

In this document, numbers prefixed with '$' are hex.

Nibbles are numbered the same way as bit numbers; from right to left, starting at 0. For example, nibble 7 on the load switches is the left-hand most significant (ms) end.

When the simulator is first started there is no microcode loaded (as with the real machine), so only a subset of commands are shown on screen. To access the full set of commands, see Loading a microcode.

4 None of these luxuries are present on the real machine; learning hex is all part of the journey.

7.1 Program control

s - Single step.

r / u / n - If FMP is reset, run at full/medium/slow speed (SMF status lights).

If FMP is set, load the microcode specified by nibble 7 (ms end) on the load switches, then halt.

q - Stop.

m / M - Set/reset FMP flag (force microprogram load) (FMP status light).

z - Zero (reset) all registers, clear the stop flags and reset system timer.

Z - Toggle speaker clicks on/off (SPK light).

This is only provided as a taster for the real machine as it impacts performance on the simulator, and may also lock up the Windows audio driver due to a bug in MS Windows, in which case all audio will stop until the next reboot.

***C** - Cycle register colours.

***Esc** - Quit the simulator.

7.2 Loading registers and memory

] / [- Increment/decrement PC.

Pg Up / Dn - Select the target register for manual loading via the Load switches.

The LOD light to the left of the target register will be lit; only one register can be the target at any one time.

l - Load the target register with the current value from the Load switches.

If the target is IR, the value is also written to memory at the current PC address.

L - Load & Inc; same as the ‘l’ command but only enabled if the target register is IR.

The value is written to IR and memory at the current PC address, and PC is incremented automatically.

With the Advanced emulation, ‘l’ and ‘L’ are the only ways of writing to the non-volatile fixed store (FST). The program simulates FST by creating a file PLASMA3.FST in the same folder as the PlasMaSim executable file and saving any changes to it. If the folder does not have write access, a 'fail to create' error message will be displayed.

7.3 Break points

Home / End - Select the target register for break-point comparisons.

The BRK light to the left of the target register will be lit; only one register can be the target at any one time.

x / X - Turn break-point checking on/off (BRK status light).

7.4 Peripheral control

These commands simulate loading and unloading media on peripherals in the real machine. The available peripherals depend on the emulation (microcode). If media is not loaded, the corresponding I/O instructions will return an error status to the program.

Media on the real machine is represented by files held on sd-cards plugged into the corresponding peripheral socket; these are described in the Machine Manual.

Media on the simulator is represented by files shown below for each peripheral type. They are all held in the same folder/directory as the PlasMaSim executable file.

Filenames are shown in upper-case as this is how they are defined in the simulator. Any files created by it will have upper-case names, but if you create files manually using lower-case, the simulator will still be able to read them as Windows does not differentiate.

See the Machine Manual for file formats.

7.4.1 Paper tape

Only valid in the Toy-B and Advanced emulations.

The reader and punch devices can be set independently to read and punch either 7 or 8-hole paper tapes. There are no checks; e.g. if a program reads a 7-hole tape with an 8-hole reader, the I/O instruction will receive garbage and/or read errors.

j / J - Load/unload paper tape on ptr tape reader; file PLASMA.PTR (PT status lights).

If no tape loaded, '**J**' toggles the device format between 7 and 8-hole (PTR status lights).

k / K - Load/unload paper tape on ptp tape punch; file PLASMA.PTP (PT status lights).

If no tape loaded, '**K**' toggles the device format between 7 and 8-hole (PTP status lights).

7.4.2 Mag tape

Only valid in the Advanced emulation.

WPR means 'write-*permit* ring', not 'write-*protect* ring', so it needs to be 'on' if you want to write to a tape.

g / G - Load/unload tape on deck mt0; file PLASMA.MT0 (MT0 status lights).

If no tape loaded, '**G**' toggles mt0 WPR on/off (MT0 status lights).

h / H - Load/unload tape on deck mt1; file PLASMA.MT1 (MT1 status lights).

If no tape loaded, '**H**' toggles mt1 WPR on/off (MT1 status lights).

7.4.3 Disk

Only valid in the Advanced emulation.

x / X - Load/unload disk on drive eds0; file PLASMA.ED0 (EDS0 status lights).

v / V - Load/unload disk on drive eds1; file PLASMA.ED1 (EDS1 status lights).

7.5 Test files

These commands are only available in the simulator; there is no equivalent on the real machine as files are accessed directly by PlasMa programs using I/O instructions.

In the simulator, all files must be in the same folder/directory as the PlasMaSim executable file.

Filenames are constructed from the current microcode number and nibbles 6-4 on the Load switches. The remaining nibbles are ignored, so you can leave nibble 7 holding the microcode number if you want.

The nibbles are interpreted as hex characters and converted to uppercase. The filename extension (.XXX) is specific to each command so is shown in the descriptions below.

***t** - Load a test .plh hex file to ram. The ram start addresses are defined within the hex file, starting from zero if no address is specified. See Hex file format for file format.

e.g. if the current microcode is 2 (Toy-B) and $3a47bf68 is set on the Load switches, pressing '**t**' will try and load a hex file named 2A47.plh

The following paper tape commands need additional information to define the ram start and end address limits. These are set with nibbles 7-4 and 3-0 respectively on the Break switches.

***o** - Load paper tape file to ram from start address onwards (the end address is ignored).

***O** - Dump ram from start address to end address to a paper tape file.

e.g. if the current microcode is 3 (Advanced), Load = $12cf56e8 and Break = $0010ffff, pressing '**O**' will create a file named 32CF.ptx containing ram contents from $0010 to $ffff, where x is 7 or 8 depending on the current tape punch format; see Peripheral control.

The '**O**' command creates the file in the same folder as the PlasMaSim executable file. If the folder does not have write access, a 'fail to create' error message will be displayed.

If you load or save paper tapes while a program is currently processing one, the result is undefined.

See the Machine Manual for paper tape file format.

7.6 Latching switches

These command keys simulate the 3 rows of latching switches on the real machine. The display shows all the switch states but they can only be set one nibble at a time.

***left**, **backspace**, **right**, **up**, **down** - Select switch row and current nibble.
The current nibble is shown within square brackets.

***0**, **1**, **2** ... **7**, **8**, **9**, **a**, **b**, **c**, **d**, **e**, **f** - Set current nibble to this hex digit value.
The next nibble becomes the current nibble.

The most significant mem address switches (on the 3rd row) are disabled in the two Toy emulations as these only need the lower 8 switches to address their 256 words of memory.

7.7 Keypad

These command keys simulate the hex keypad on the real machine. The display shows a single nibble representing which of its 16 buttons is pressed.

***up**, **down** - Select keypad. The nibble is shown within square brackets.

***0**, **1**, **2** ... **7**, **8**, **9**, **a**, **b**, **c**, **d**, **e**, **f** - Set nibble to this hex digit value.

The keypad and displayed value are only active at certain time depending on the emulation.

With Toy-A, the keypad is only enabled when the program halts due to a special type of load instruction. When this occurs, the kpad light turns on, and each keypad button then inserts a hex nibble into the target register. Pressing '**r**' (or any of the other run or step commands) resumes the program.

Toy-B has a similar feature, but uses a specific I/O function called 'TTY Read'.

Both Toy-B and the Advanced emulations also support a polling variation so the machine doesn't need to stop. See the Machine Manual for full details.

8. Loading a microcode

Full details for the real machine are in the Machine Manual (or see video #2), but here's a summary of the equivalent key-presses for the simulator:

z - reset.

m - set FMP; confirm FMP light is on.

cursor keys - move nibble highlight to Load switches nibble 7 (ms end).

number key - microcode number required.

r - run; wait for Halt light to turn on.

z - reset.

M - reset FMP; confirm FMP light is off.

9. Hex file format

These are simple text files where each line contains either an address or a data value using the following format:-

- The letter '**m**' or '**M**' followed by two or more hex digits. This defines a new start address for subsequent data values; the default start address is 0 (note ramifications for the Advanced emulation below).
- Four hex digits defining the data value to be loaded into ram at the current address. The address is incremented automatically so consecutive data lines will be loaded to consecutive addresses.

Comments can be added by prefixing them with a ';' character. Everything from that character to the end of line will be ignored. Blank lines are also ignored.

All hex digits can use upper or lower case.

In the Advanced emulation, the FST memory can only be written using the Load switches, so attempts to write to addresses $00 to $0F from a test file will be ignored.

9.1 Example hex file for Toy-A (hand-assembled)

- Copy and paste the lines below into a plain text file called 1901.plh in the same folder as the PlasMaSim executable file.
- Start the simulator: PlasMaSim*N* (where *N* depends on your simulator variant; all variants can load the Toy-A microcode).
- Load microcode 1 (Toy-A); see Loading a microcode.
- Set nibbles 6-4 on the Load switches to 901 (nibbles are numbered 0-7 from right to left).
- Press '**t**' to load the program, then press '**Esc**' to acknowledge the prompt.
- Press '**z**' to reset all registers.
- Press '**r**' to run the program.

```
;add 2 numbers from memory and display result
m00
C010  ;00 jz r0 to L10; branch to start[5]

;start of data
1234  ;01 L01: num1
5678  ;02 L02: num2

;start of code
m10
8101  ;10 L10: ld: r1 = num1
91FF  ;11 st: mem[$FF] = r1; display num1
8302  ;12 ld: r3 = num2
93FF  ;13 st: mem[$FF] = r3; display num2
1513  ;14 add: r5 = r1 + r3
95FF  ;15 st: mem[$FF] = r5; display sum
0000  ;16 halt
```

5 R0 is always zero in Toy-A, so this is effectively an unconditional jump.

9.2 Example hex file for Toy-A (auto-assembled)

- Copy and paste the lines below into a plain text file called 1902.pls in the same folder as the PlasMaSim executable file.
- Open a command window and run the assembler: Plasm*N* -h 1902 (where *N* depends on your assembler variant; all variants can assemble Toy-A programs).
  This should create a hex file called 1902.plh
- Start the simulator: PlasMaSim*N* (where *N* depends on your simulator variant; all variants can load the Toy-A microcode).
- Load microcode 1 (Toy-A); see Loading a microcode.
- Set nibbles 6-4 on the Load switches to 902 (nibbles are numbered 0-7 from right to left).
- Press '**t**' to load the program, then press '**Esc**' to acknowledge the prompt.
- Press '**z**' to reset all registers.
- Press '**r**' to run the program.

```
;add 2 numbers from memory and display result
%s 1  ;Toy-A source code
%m 0  ;set memory address
      jz r0 start
%d    ;start of data section
      ;numbers are decimal unless prefixed with '$'
.n1   1234  ;decimal
.n2   5678  ;decimal
%c    ;start of code section
.start
      ld r1 n1
      st $ff r1  ;display n1
      ld r3 n2
      st $ff r3  ;display n2
      add r5 r1 r3
      st $ff r5  ;display sum
      hlt
```