Simulator
JHB Note 1
24 Apr 84

MSIM: A Diagnostic Emulator for the Motorola M6800 and M6809

This note describes a diagnostic tool for use in debugging Motorola M6809 assembler programs. The program is written in IMP and is intended to be run on EMAS, VAX/VMS or APM. The program allows a full instruction-by-instruction trace, breakpoints, pseudo-I/O porting and so on. It is not yet possible to simulate asynchronous interrupts or events depending on timing. Timewasting loops should be patched out before running under emulation to avoid very long delays.

Invoking The Emulator

The emulator is invoked by the command line

```
     M9SIM <Control File>                    M6809
or   M0SIM <Control File>                    M6800
```

The Control File

This contains the basic commands needed to drive the emulation. This file defaults to TT: (.IN on EMAS) and may contain the commands below (one per line). A default extension of .CTR (#CTR on EMAS) will be assumed.

MAP <low> <high>
: This specifies a range of addresses which the program may use. MAP COMMANDS MUST PRECEDE ALL OTHER COMMANDS AND AT LEAST ONE MAP COMMAND MUST BE SPECIFIED. All addresses referenced by the program (code, stack and data) must be covered by MAP commands. Memory-mapped I/O ports need not. MSIM ignores the least significant byte of MAP addresses so MAP 0 0 is sufficient to map the whole of page 0. Mapping is used by the program to detect erroneous access out of range and to reduce the size of the MSIM working set. MAP 0 ffff will work fine but will slow the program down considerably.

OBJ <filename>
: This specifies an object module to be loaded. The object should be in the Motorola reload format output by the EUCSD M6809 assembler. The command file may contain more than one OBJ command. A default extension of .OBJ (#OBJ) will be assumed

TRACETO <filename>
: This nominates a file for trace output. Default is TT: (.OUT). A Default extension of .TRS (#TRS) will be generated

TRACEON <low> <high>
: This command defines absolute address bounds within which tracing should be turned on. <low> and <high> are in hexadecimal.

TRACEOFF <low> <high>
: This defines a range in which tracing is turned off. The program does not currently cope with the case where tracing bounds overlap. Tracing is turned off by default.

SET (<addr> | "R"<reg>) <val>
: Loads <val> into address <addr>. <val> and <addr> are in hexadecimal. This command is useful for patching out timewasting loops etc. A register may be specified (<reg> = a, b, c/cc, d, dp, p/pc, u/us, s/sp, x, y)

FIX <addr> <val>
: Fixes <val> into address <addr> such that any read on <addr> will always return <val>. Useful for holding I/O status bytes at known values.

ACIA <addr>
: Tells the program that location <addr>-<addr+1> represents a 6850 ACIA. Any store to this location will output to the default output device or the file nominated in a DEVICE command and any fetch will read from the default input or nominated input file.

PIA <addr>
: Nominates <addr>-<addr+3> as a 6820 PIA in the same manner as the ACIA command.

PTM <addr>
: Nominates <addr>-<addr+7> as a 6840 PTM.

BAUD <addr>
: Nominates <addr>-<addr+1> as a baud-rate generator.

DEVICE <addr> <devspec>
: Tells the program that input or output to or from the device at <addr> should be taken from the files specified in <devspec>.
<Devspec> is of the form

TO <output file> FROM <input file>

The TO and FROM components may be specified in either order. <input file> and <output file> default to TT: (.IN & .OUT on EMAS). <addr> should refer to an I/O port covered by a previous ACIA,PIA,PTM or BAUD command.

CTRLSON
: Control characters (ASCII 16 00 to 16 1F) will be sent as generated, and will also be sent in single-character mode. This mode aims to emulate as closely as possible a VDU attached to an M6809 kit. This mode is not yet available on EMAS.

CTRLSOFF
: Control characters other than carriage return and linefeed will be sent as their hex value in square brackets preceded by a newline. Output will be buffered by the system until a newline or a control character is output. This is the default mode.

DUMP <low> <high>
D <low> <high>
: Dumps from address <low> to <high> in hex to the console. If <high> is omitted it will take the value <low>.

REGS
: Gives a register dump to the console.

BREAK <no> <addr>

Sets break point <no> at address <addr>. <no> may be between 0 & 7. If <addr> is omitted or is FFFF, the breakpoint will be unset.

RESTART <addr>

Restarts the program from either the nominated address or (by default) the power-on restart location FFFE. A restart from FFFE will be assumed if no RESTART or GO directive is supplied.

GO <addr>

Starts or resumes execution of the program from either the nominated address or (by default) the next instruction. Registers other than PC are not altered.

STDSVCS <addr>

Instructs the program to intercept the SWI instruction (16 3F) and the following byte and invoke the appropriate CS2 microkit SVC (appendix 1). <addr> is the address of a previously defined ACIA and will be used for I/O by the SVCs.

STEP <no>

Step <no> instructions on from breakpoint

STOP
QUIT

End the simulation. This may also be achieved by planting an infinite loop (bra *) in the code.

Error Messages

Guarded Access NNNN

An attempt has been made to access unMAPped store location NNNN. Likely causes include not setting up stack pointers, not MAPping the restart address, overlaying code with data etc. This is a catastrophic error and MSIM will abort with the message MSIM Abort - cannot recover.

End of file on stream N

Program has read off the end of the indicated stream. Stream allocations are stated at the top of the trace.

MSIM Abort - No Page Map

No MAP commands were given at the start of the control file. MSIM regards this as a catastrophic error and aborts.

Illegal SVC

The STDSVC command only allows SVCs in the range 0-16.

Invalid Opcode

The simulator has been asked to execute an invalid M6809 instruction. If this is 16 55 this is probably unallocated store.

End of simulation

Normal end of simulation due to Stop or Quit command or loopstop.

Program Startup

Store is filled by 16 55 throughout by MSIM once the mapping definitions have been read. This is an invalid instruction so should quickly trap attempts to execute vacant store. Registers are set to 0 except for PC which is either specified directly (GO) or taken from a nominated address (RESTART). It is left to the program to set its user and system stack pointers and initialise devices and local variables.

The trace

This consists of one or more lines per instruction, giving the following information..

PC	Program Counter at start of execution of instruction
p	Instruction prefix (page 2 or 3 instructions only)
In	Instruction opcode
Args	Instruction arguments
a	A register
b	B register
x	X register
y	Y register (this is omitted in tne M6800 emulator)
us	User stack pointer
sp	System stack pointer (omitted in M6800 emulator)
cc	condition code register
-	Memory accesses

The register values are those after execution of the indicated instruction except for the PC.

Memory accesses shown are all fetches and stores except for those associated with fetching the instruction itself. They are displayed as

A '*' is placed in the trace file whenever the trace switches off.

F/AAAA: VV = Fetch from address AAAA (value VV returned)
S/AAAA: VV(WW) = Store at address AAAA (Value WW replaced by VV)

Access to emulator

Vax

The M6809 emulator lives at dr0:[jhb]m9sim.exe and does not require a DCL macro. Insert the statement "m9sim :== $dr0:[jhb]m9sim" into your login.com file. The M6800 version is in m0sim.exe.

EMAS

Insert ecsc62.m9sim#obj then call as for VAX except that terminal input should be referred to as .IN and output files as .OUT. The m6800 version lives in ecsc62.m0sim#obj.

Speed

Speed under absolutely ideal conditions is approximately 5000 M6809 clocks per VAX CPU second with tracing off and about 900 clocks/second fully traced. These timings will vary widely depending on the program as there is no direct relationship between the time taken to emulate an instruction and its actual execution time. Timings quoted are for the program..

```
        ldx #0
loop    leax -1,x
        bne loop
```

Appendix 1: CS2 Microkit SVCs

Quantities referred to are..

Registers: A, B, D (A+B), X, Y.
16-bit top of stack: This is the 16-bit quantity at the top of the system stack when the SVC is issued.
String: This is a set of up to 255 characters preceded by a length byte.

00	muld	Multiply D by the 16-bit top of stack and return the value to D.
01	divd	Divmod: Divide D by 16-bit top of stack. Quotient is returned to the top of the stack and remainder is left in D.
02	lshift	Left-shift 16-bit top of stack by A
03	rshift	Right-shift 16-bit top of stack by A
04	skips	Skipsymbol: Read a symbol and discard it
05	nexts	Nextsymbol: Return the value of the next symbol but don't read it
06	reads	Readsymbol: Read a symbol and return it in B.
07	readb	Read a decimal value <= 255 and return it in B.
08	readw	Read a decimal value v, -32768 <= v <= 32767 and return it in D.
09	prtsym	Printsymbol: Print the symbol in B.
0A	prthex	Printhex: Print B as a two character hexadecimal number.
0B	prtstr	Printstring: Print the string pointed at by X.
0C	writeb	Writebyte: Write B as a signed decimal number.
0D	writew	Writeword: Write D as a signed decimal number. ** Note difference from spec. ** First character will be - sign or space.
0E	space	Space: Print a space
0F	spaces	Spaces: Print B spaces.
10	newln	Newline: Print a linefeed (ASCII 16 0A)
11	newlns	Newlines: Print B linefeeds
12	stop	Halt execution and enter monitor
13	wait	Wait
14	disreg	Dump registers
15	dump	Dump store between X and Y
16	prompt	Change Prompt.

Appendix 2: Examples

1) A suitable control file for running a small program "fred#obj" which uses locations 0 - 16 400 and the standard SVCs is..

```
map 0 3ff                       Code area
map 7ffe 7fff                   Restart vector and system stack
traceto x                       Trace file will go to x#trs
traceon 0 ffff
obj fred                        Object file is fred#obj
acia 100a                       for use by SVCs
device 100b to y                All SVC output will go to y#out
stdsvcs 100a
fix 100a 03                     Set the ACIA to be permanently ready
restart 7ffe
```

The program should have a pointer to its start address in location 7ffe and should set its system stack to 7ff0. User stack is as 16 400

This produces in file x.trs the trace...

```
Object File = fred#obj
Control File = x#ctr                              16-APR-1984 13:47:55

Device Allocations
ACIA 100A-100B       From: 0 .IN              To: 0 y#out

Fix-list

Fix   100A 03

cc bits are: EFHINZVC.  Register push order is pc,u/s,y,x,dp,b,a,cc

pc   p In Args   a  b  x    y    us   sp   cc          memory accesses

                 00 00 0000 0000 0000 0000 00 S/FCAB: 39(CC) F/FFFE: FC
                                                        F/FFFF: 10
*
FCB4   F6 100A   1B 03 0000 0000 0200 03FC 00 F/100A: 03
FCB7   C5 02     1B 03 0000 0000 0200 03FC 00
FCB9   27 F9     1B 03 0000 0000 0200 03FC 00
FCBB   B7 100B   1B 03 0000 0000 0200 03FC 00 S/100B: 1B(00)
FCBE   39        1B 03 0000 0000 0200 03FE 00 F/03FC: FC      F/03FD: C3
*
FCB4   F6 100A   76 03 0000 0000 0200 03FC 00 F/100A: 03
FCB7   C5 02     76 03 0000 0000 0200 03FC 00
FCB9   27 F9     76 03 0000 0000 0200 03FC 00
FCBB   B7 100B   76 03 0000 0000 0200 03FC 00 S/100B: 76(00)
FCBE   39        76 03 0000 0000 0200 03FE 00 F/03FC: FC      F/03FD: C7
*
FCB4   F6 100A   1B 03 FC02 0000 01FD 03FA 00 F/100A: 03
FCB7   C5 02     1B 03 FC02 0000 01FD 03FA 00
FCB9   27 F9     1B 03 FC02 0000 01FD 03FA 00
FCBB   B7 100B   1B 03 FC02 0000 01FD 03FA 00 S/100B: 1B(00)
FCBE   39        1B 03 FC02 0000 01FD 03FC 00 F/03FA: FC      F/03FB: CC
*
```