Appendix C Motorola 68000 Simulator and Assembler

This appendix describes the Motorola 68000 simulator (`Sim68000`) and assembler (`68kasm`) included in the BSVC distribution.

Simulator

The Motorola 68000 simulator included in the BSVC distribution simulates the 68000 at the software level. This means the simulator does not understand what goes on in the 68000 at the hardware level. Instead the simulator performs a set of actions for each instruction that gives the same result.

Notes

- The simulator has a few areas in which it does not behave exactly like a 68000
  - Address and Bus errors do not generate a "complete" exception stack
  - Some illegal instructions may be mistaken for real instructions
- Double bus errors cause the 68000 to halt and require a system reset

Simulator Devices

The 68000 simulator provides two devices that can be attached to the microprocessor. These devices are the M68681 Dual UART and RAM. Each of these devices is described below.

M68681 Dual UART Device

The M68681 Dual Universal Asynchronous Receiver/Transmitter (DUART) device is modeled after the Motorola MC68681 DUART. The dialog box in Figure 15 allows you to customize the M68681 device when it's attached to the 68000. You can select the:

- Base address of the device
- Byte offset to the first register of the device[8]
- Offset between registers
- Interrupt level
- Port commands[9]
- Port input and output attachments[10]

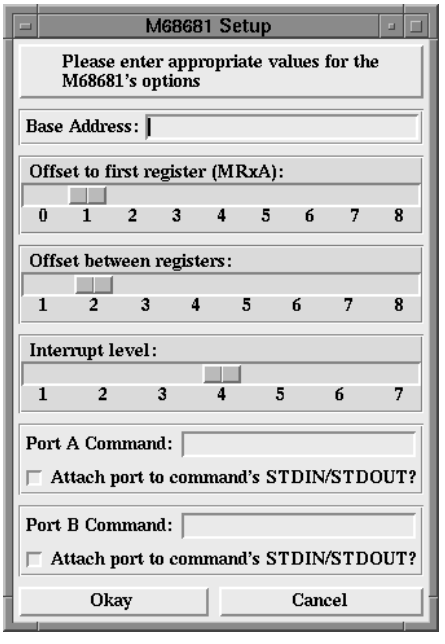

FIGURE 15 M68681 Device Setup Dialog

[8] This allows you to connect the port to the upper or lower 8 bits of the data bus.

[9] This is the Unix command to connect the port's I/O to (usually an xterm running xtermpipe). If the process buffers its I/O there will be problems.

[10] The port's I/O can be attached to the command's standard I/O or to file descriptors 3 and 4.

Features

The M68681 device supports the following features:

- Two, independent, full-duplex asynchronous Receiver/Transmitter ports
- Each port can be connect to a Unix process
- Independently programmable baud rate for each Receiver and Transmitter
  - 18 Fixed rates: 50 to 38400 baud
- Programmable data format allowing five to eight data bits
- Programmable channel modes
  - Normal (full-duplex)
  - Automatic echo
- Versatile interrupt system
  - Single interrupt output with four maskable interrupting conditions
  - Interrupt vector output on interrupt acknowledge

Communication Ports A and B

Each communication port comprises a full-duplex universal asynchronous receiver/transmitter (UART). The baud rate for each receiver and each transmitter can be independently selected using the built in baud rate generator.

The transmitter accepts parallel data from the CPU and sends it to the Unix process connected to the port. The receiver accepts data from the Unix process connected to the port and transfers it to the CPU during read operations. If no process is connected to the port there will never be any data to read and writes will send data to that giant-bit-bucket-in-the-sky.

Programming and Register Description

The operation of the DUART is programmed by writing control words into the appropriate registers. Operational feedback is provided by the status registers, which can be read by the CPU. The DUART's register addresses are described in Table 1.

TABLE 1 Register Addressing

RS4	RS3	RS2	RS1	Read	Write
0	0	0	0	Mode Register A (MR1A, MR2A)	Mode Register A (MR1A, MR2A)
0	0	0	1	Status Register A (SRA)	Clock Select Register A (CSRA)
0	0	1	0	Do not access	Command Register A (CRA)
0	0	1	1	Receiver Buffer A (RBA)	Transmitter Buffer A (TBA)
0	1	0	0	Not implemented	Auxiliary Control Register (ACR)
0	1	0	1	Interrupt Status Register (ISR)	Interrupt Mask Register (IMR)
0	1	1	0	Not implemented	Not implemented
0	1	1	1	Not implemented	Not implemented
1	0	0	0	Mode Register B (MR1B, MR2B)	Mode Register B (MR1B, MR2B)
1	0	0	1	Status Register B (SRB)	Clock Select Register B (CSRB)
1	0	1	0	Do not access	Command Register B (CRB)
1	0	1	1	Receiver Buffer B (RBB)	Transmitter Buffer B (TBB)
1	1	0	0	Interrupt Vector Register (IVR)	Interrupt Vector Register (IVR)
1	1	0	1	Not implemented	Not implemented
1	1	1	0	Not implemented	Not implemented
1	1	1	1	Not implemented	Not Implemented

Mode registers 1 and 2 of each port are accessed through independent pointers. The pointer is set to registers MR1A and MR1B by a system reset or issuing a "reset pointer" command via the corresponding command register. Any read or write of the mode register while the pointer is at MR1A or MR1B switches the pointer to MR2A or MR2B. Subsequent accesses will address MR2A or MR2B unless the pointer is reset to MR1A or MR1B.

Mode, command, clock select, and status registers are duplicated for each port to provide independent operation and control. Refer to the register bit formats in Table 2.

For information on programming the M68681 you should consult a programmer's guide to the Motorola MC68681 DUART.

TABLE 2 Register Bit Formats

Port A Mode Register 1 (MR1A) and Port B Mode Register 1 (MR1B)

N/A	Rx IRQ Select	N/A	N/A	N/A	Bits-per-character
Bit 7	Bit 6 0 = RxRDY 1 = FFULL	Bit 5	Bit 4 Bit 3	Bit 2	Bit 1 Bit 0 0 0 = 5 0 1 = 6 1 0 = 7 1 1 = 8

Port A Mode Register 2 (MR2A) and Port B Mode Register 2 (MR2B)

Port Mode	N/A	N/A	N/A
Bit 7 Bit 6	Bit 5	Bit 4	Bit 3 Bit 2 Bit 1 Bit 0
0 0 = Normal 0 1 = Auto Echo 1 0 = N/A 1 1 = N/A			

Clock-Select Register A (CSRA) and Clock-Select Register B (CSRB)

Receiver-Clock Select			Transmitter-Clock Select		
Bit 7 Bit 6 Bit 5 Bit 4	Set 1 ACR Bit 7 = 0	Set 2 ACR Bit 7 = 1	Bit 3 Bit 2 Bit 1 Bit 0	Set 1 ACR Bit 7 = 0	Set 2 ACR Bit 7 = 1
0 0 0 0	50	75	0 0 0 0	50	75
0 0 0 1	110	110	0 0 0 1	110	110
0 0 1 0	134.5	134.5	0 0 1 0	134.5	134.5
0 0 1 1	200	150	0 0 1 1	200	150
0 1 0 0	300	300	0 1 0 0	300	300
0 1 0 1	600	600	0 1 0 1	600	600
0 1 1 0	1200	1200	0 1 1 0	1200	1200
0 1 1 1	1050	2000	0 1 1 1	1050	2000
1 0 0 0	2400	2400	1 0 0 0	2400	2400
1 0 0 1	4800	4800	1 0 0 1	4800	4800
1 0 1 0	7200	1800	1 0 1 0	7200	1800
1 0 1 1	9600	9600	1 0 1 1	9600	9600
1 1 0 0	38400	19200	1 1 0 0	38400	19200
1 1 0 1	N/A	N/A	1 1 0 1	N/A	N/A
1 1 1 0	N/A	N/A	1 1 1 0	N/A	N/A
1 1 1 1	N/A	N/A	1 1 1 1	N/A	N/A

Port A Command Register (CRA) and Port B Command Register (CRB)

N/A	Misc Commands	Transmitter	Receiver
Bit 7	Bit 6 Bit 5 Bit 4	Bit 3 Bit 2	Bit 1 Bit 0
	0 0 0 = No Command 0 0 1 = Reset MR Pointer 0 1 0 = Reset Receiver 0 1 1 = Reset Transmitter 1 0 0 = Reset Error Status 1 0 1 = N/A 1 1 0 = N/A 1 1 1 = N/A	0 0 = No Action 0 1 = Transmitter Enabled 1 0 = Transmitter Disabled 1 1 = N/A	0 0 = No Action 0 1 = Receiver Enabled 1 0 = Receiver Disabled 1 1 = N/A

Port A Status Register (SRA) and Port B Status Register (SRB)

N/A	N/A	N/A	N/A	TxEMT	TxRDY	FFULL	RxRDY
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3 0 = No 1 = Yes	Bit 2 0 = No 1 = Yes	Bit 1 0 = No 1 = Yes	Bit 0 0 = No 1 = Yes

Auxiliary Control Register (ACR)

Baud Rate Set	N/A			N/A	N/A	N/A	N/A
Bit 7 0 = Set One 1 = Set Two	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

Interrupt Status Register (ISR)

N/A	N/A	RxRDYB FFULLB	TxRDYB	N/A	N/A	RxRDYA FFULLA	TxRDYA
Bit 7	Bit 6	Bit 5 0 = No 1 = Yes	Bit 4 0 = No 1 = Yes	Bit 3	Bit 2	Bit 1 0 = No 1 = Yes	Bit 0 0 = No 1 = Yes

Interrupt Mask Register (IMR)

N/A	N/A	RxRDYB FFULLB	TxRDYB	N/A	N/A	RxRDYA FFULLA	TxRDYA
Bit 7	Bit 6	Bit 5 0 = Masked 1 = Pass	Bit 4 0 = Masked 1 = Pass	Bit 3	Bit 2	Bit 1 0 = Masked 1 = Pass	Bit 0 0 = Masked 1 = Pass

RAM Device

The RAM device allows you to attach a specified amount of memory to the 68000. When the RAM device is attached to the simulator, using the Edit Setup Dialog, the dialog shown in Figure 16 is displayed. This dialog allows you to enter the base address and size of the RAM module. Both of these values are in hexadecimal.

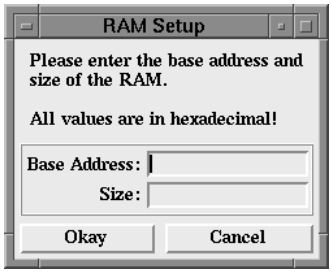

FIGURE 16 RAM Device Setup Dialog

Assembler

The 68000 assembler is similar to many UNIX system compilers and assemblers. The assembler accepts files with the following line layout:

```
LABEL     OPCODE     OPERANDS     COMMENTS
```

The `LABEL` field is optional, but if used it must start in the first column of the line. Fields are separated by spaces or tabs. An example assembly language file is shown in Figure 17.

```
        ORG     $2000       Start at location 2000 Hex
START   CLR.W   SUM         Clear variable SUM
        MOVE.W  COUNT,D0    Load COUNT value
LOOP    ADD.W   D0,SUM      Add D0 to SUM
        SUB.W   #1,D0       Decrement counter
        BNE     LOOP        Loop if counter not zero
        BREAK               Stop Simulator if it's running
SUM     DS.W    1           Reserve one word for SUM
COUNT   DC.W    25          Initial value for COUNT
```

FIGURE 17 Sample Assembly Language File

Once an assembly language file has been created it can be assembled. The assembler will indicate if any syntax errors were found. If no errors were encountered the assembler creates an object file in Motorola S-Record format with a .H68 suffix. If errors were found, to get a better idea of the errors assemble the file using the "`-l`" option. This option tells the assembler to produce a listing file with a .LIS suffix in addition to the .H68 file. The listing file can be used to find syntax errors as-well-as logic errors while executing the program.

Once the file has been assembled with no errors, the object file can be loaded into the simulator for execution. The listing file can also be loaded into BSVC's Program Listing window to facilitate debugging.

Acknowledgments

The Motorola 68000 simulator was developed by Bradford W. Mott at North Carolina State University. The simulator is loosely based on the 68000 simulator written by Jay Lloyd at NCSU.

The Motorola 68000 assembler was developed by Paul McKee at NCSU.

Contacts

If you have questions about the Motorola 68000 Simulator contact:

Bradford W. Mott
bwmott@eos.ncsu.edu