

LA-UR -81-2991

TITLE: AN LSI-11/MOTOROLA MICROCOMPUTER-DEVELOPMENT SYSTEM **MASTER**

AUTHOR(S): Steven C. Bourret

SUBMITTED TO: 1981 Nuclear Science Symposium  
San Francisco, California  
October 21-23, 1981  
For publication in 1982 IEEE Trans. Nucl. Sci.

University of California

By acceptance of this article, the publisher recognizes that the U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or to allow others to do so, for U.S. Government purposes.

The Los Alamos Scientific Laboratory requests that the publisher identify this article as work performed under the auspices of the U.S. Department of Energy.



**LOS ALAMOS SCIENTIFIC LABORATORY**

Post Office Box 1663 Los Alamos, New Mexico 87545

An Affirmative Action / Equal Opportunity Employer



# AN LSI-11/MOTOROLA MICROCOMPUTER DEVELOPMENT SYSTEM\*

by

Steven C. Bourret  
Los Alamos National Laboratory  
Group Q-1, MS 540  
Los Alamos, NM 87545

A microcomputer development system was built using an RT-11 operating system and a cross assembler in developing a Motorola MC6801-L1 microcomputer-based stepping-motor controller to work with an LSI-11 host computer. The advantage of this technique is that we can use the familiar LSI-11 hardware and the powerful RT-11 operating system software for other microcomputer development. This paper describes how the technique works and its use in developing the stepping-motor controller.

## Introduction

The Digital Equipment Corporation (DEC) LSI-11 is used for 90% of the computer needs in the Safeguards Assay Group at the Los Alamos National Laboratory. However, for certain instrumentation development, we also need to use the capabilities of a microcomputer on a chip, such as the Motorola MC6801-L1. In developing the hardware and software for a special stepping-motor controller, we used an LSI-11 with its RT-11 operating system, an MC6801-L1 microcomputer, and a cross assembler. Appropriate software and procedures were written to make this configuration into a usable microcomputer development system.

## Hardware

The hardware (Fig. 1) consists of the LSI-11 system and the MC6801-L1 microcomputer, which is mounted on a standard DEC foundation module. The LSI-11 system is made up of a teletype terminal, a DSD dual-drive floppy disk for the system device, and a chassis that houses the interfaces, memory, and CPU.

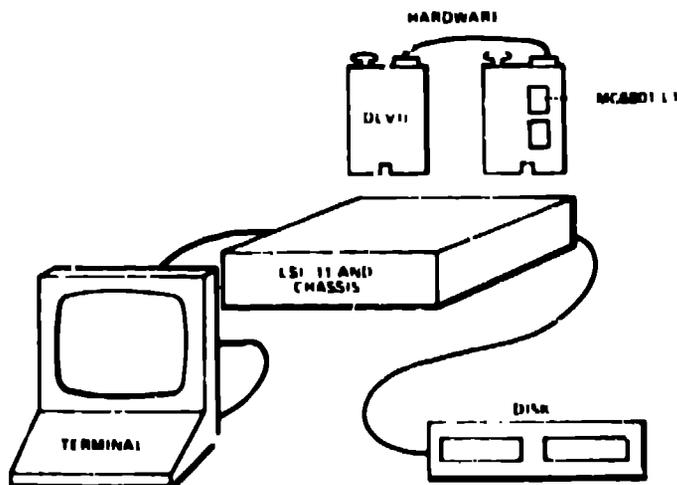


Fig. 1.  
LSI-11 system hardware.

The MC6801-L1 microcomputer has an enhanced M6800 instruction set that allows all of the M6800 software to be upward compatible. It has a serial communications interface, 4 programmable I/O ports, 2048 bytes of ROM, and 128 bytes of RAM (expandable to 24-K

\*This work was performed under the auspices of the US Department of Energy.

bytes). The MC6801-L1 also has a 16-bit programmable timer and a LILBUG monitor program resident in the 2-K bytes of PROM. All these features are contained in one 40-pin chip that requires only a single +5 V power supply.

Special features were added to the MC6801-L1 to expand its functions:

- (1) Buffers to drive the serial line for the RS-232 standard voltage. (The serial line out of the MC6801-L1 uses TTL voltage levels.)
- (2) One crystal to run the MC6801-L1 at 300 and 9600 baud.
- (3) Circuitry to cause a restart from the serial line (for sensing format errors).
- (4) Chips to expand the amount of RAM (for controlling the read and write logic).

Figure 2 shows a diagram of the foundation module with the added chips; Fig. 3 shows the logic circuitry. Additional space is available on the board to add special interfacing circuitry.

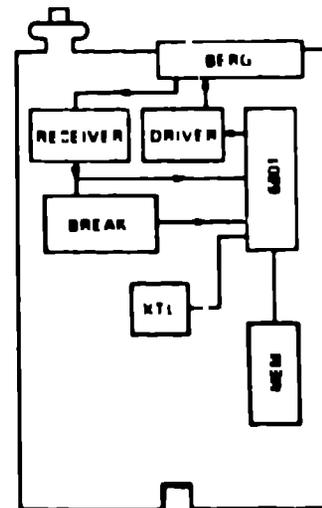


Fig. 2.  
Diagram of foundation module and chips.

## Software

The special software used with the RT-11 operating system consists of the cross assembler; the MOUT program, which establishes a link between the LSI-11 and the MC6801-L1; the KB handler; and the LILBUG monitor.

The cross assembler, a relocating assembler written for the MC6801, was purchased from Boston Systems for the LSI-11/Motorola microcomputer development system.

MOUT is a utility program that establishes a software link between the LSI-11 and its console terminal and the MC6801-L1 LILBUG. This link is made over a serial line and runs at 300 baud. When a key is struck

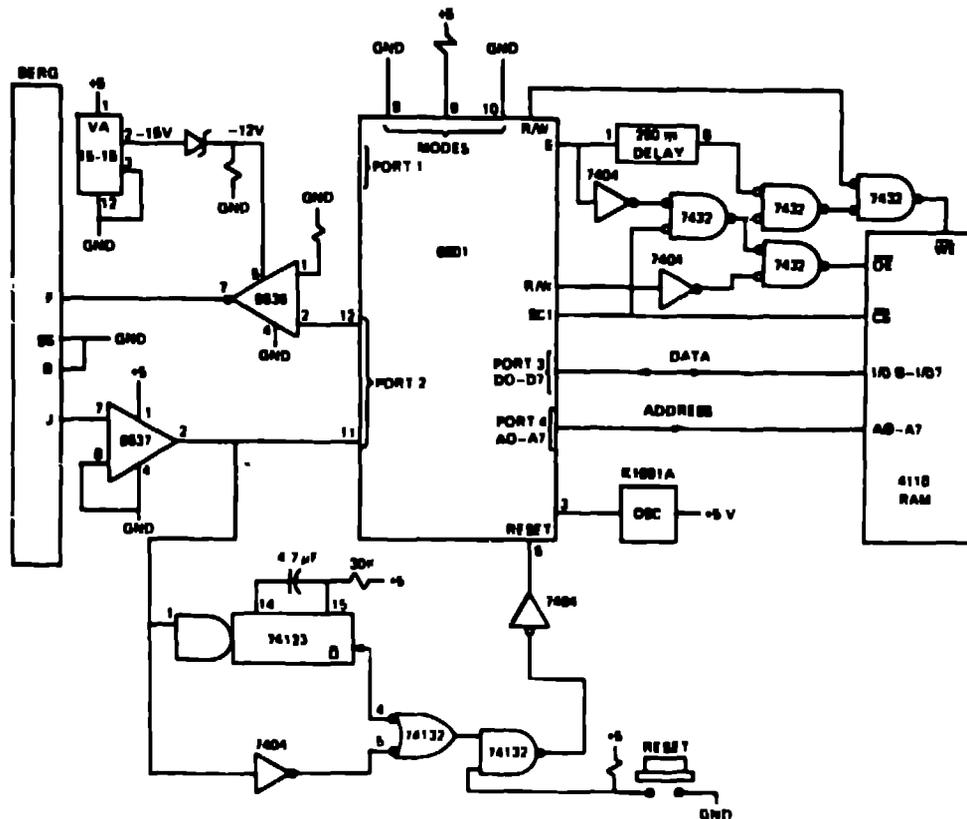


Fig. 3.  
Logic circuitry.

on the LSI-11 console terminal, the character is sent to the MC6801-L1 and then echoed back and printed on the console display. Also, when the MC6801-L1 sends a message to the LSI-11, the message is relayed to the console display. The LSI-11 looks transparent to the user--as if one were talking directly to the MC6801-L1. Additional features of MODT are a Control C, which will send the user back to RT-11 monitor, and a Control B, which will send the LILBUG back to restart.

The KB handler, a program written for the RT-11 operating system to handle a second terminal keyboard and printer, is used to down-line load programs that are outputs from the cross assembler. The handler is invoked by the RT-11 utility program PIP; it is used in the same manner as any other handler or program used by PIP. For example:

```
.R PIP
*KB:=TEST.OBJ
*
```

The LILBUG program is contained in the 2-k bytes of PROM. This program is entered upon a restart pulse from PIN 6 on the chip. The LILBUG monitor sets itself up for 300 baud, sends out a message over its serial line, and then waits for one of its many commands. LILBUG can load and verify a program, dump a program, and examine and change data in memory and registers. It has break points, it can punch a program, and it can free run or trace through a program.

#### Procedure

- (1) Run RT-11, then run TECO or another editor.
- (2) Enter the program.
- (3) Run the cross assembler.

```
*RUN CR6801
TEST.OBJ, TEST.LST = TEST.SRC
```

- (4) When the cross assembler has finished and there are no errors in the program, go back to RT-11 monitor and run MODT; it will restart LILBUG and print the start-up message.

- (5) Set up the MC6801-L1 to use its external memory by setting Port 1 data direction register to be output. (The register is located at Address 5.)

- (6) Invoke the l command for loading.

```
.RUN MODT
LILBUG
: M 5/FFFF
:L
```

- (7) Return to RT-11 monitor by typing Control C.

Now you are ready to use PIP and the KB handler to down-line load.

```
.R PIP
*KB:=TEST.OBJ
*
```

When it is all done, return to RT-11 monitor and run MODT. This time the program can be examined with LILBUG to debug. If the program does not work, return to RT-11 and re-edit the program. Figure 4 shows a flow chart of this procedure.

#### Controller

In developing the stepping motor controller, we used the LSI-11 to provide the pulse train to the

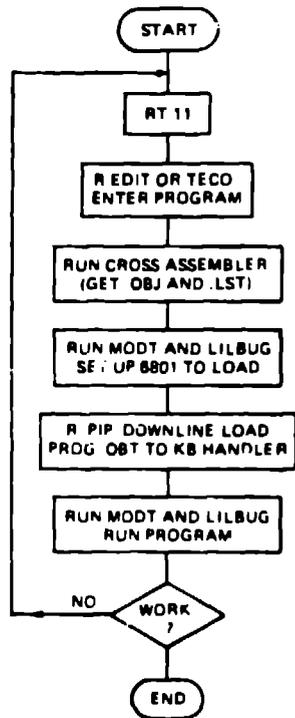


Fig. 4.  
Software development procedure.

driver. The controller used the pulses returning from the incremental encoder on the motor to provide self-ramping. To make the controller with the MC6801-L1 requires a translated program and a method to communicate with the host LSI-11 computer.

Because space was available on the foundation module for additional circuitry, we added an interface between the motor driver, the motor encoder, and the MC6801-L2. This required two IC chips; the logic is shown in Fig. 5.

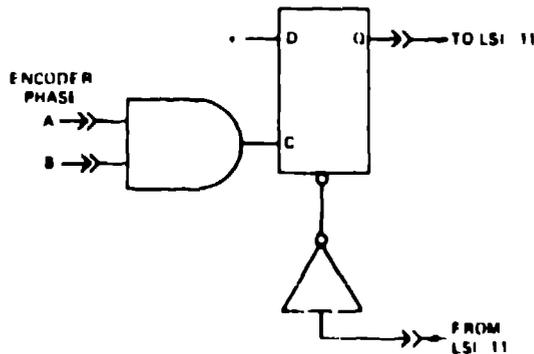


Fig. 5.  
Stepping motor controller schematic.

The sequence of events for this controller follows.

- (1) A step given by the MC6801-L1 program is sent to the stepping-motor driver, causing the motor to step.
- (2) When the motor has completed its step, the motor encoder sends a pulse back to the interface on the foundation module, setting the flip-flop.
- (3) The program has been waiting for this flip-flop to set, and when it does, the program clears the flip-flop.
- (4) A delay is done corresponding to the programmed speed.
- (5) The next step is given.

The sequence allows maximum possible ramping speed without any loss of synchronization. For the same programmed speed but a larger load, the motor either will take longer to arrive at the speed or will never quite reach the top speed. The flow chart for the program is shown in Fig. 6.

#### Conclusions

Compared to a \$20 000 commercial development system, this system has a faster assembly speed, a more powerful editor, higher speed peripherals, the familiar RT-11 operating system, and a link between the PDP-11 and the MC6801-L1. Because this system serves two computers, it is more cost effective for our application.

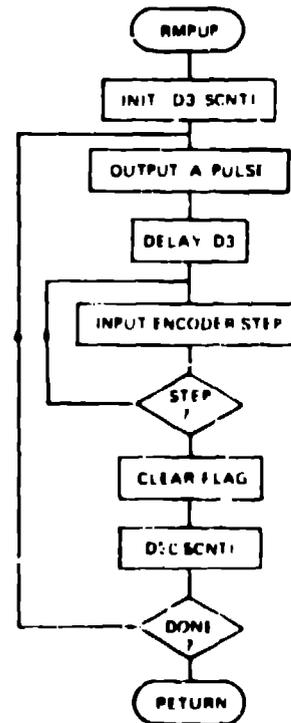


Fig. 6.  
Stepping motor program flow chart.