In-circuit emulator

JULIO ORTEGA and WILLIAM L. PALYA
Jacksonville State University, Jacksonville, Alabama 36265

A microprocessor-based in-circuit emulator is described. The emulator is for 6802, 6808, and 6809 microprocessors. It does not operate at full bus speed or execute instructions from the target machine. It is designed to be an inexpensive debugging tool for 6800 family microprocessors. It will identify design or construction errors relatively quickly by executing a set of diagnostics from firmware.

Microprocessor-based controllers can solve a wide variety of the instrumentation problems that arise in a behavioral laboratory. These controllers approximate the functionality of a rack of relay or solid state programming equipment and yet require only a few easily interconnected chips. Small special-purpose controllers can be used as dedicated process control/data acquisition systems, as peripheral processors to unload some function from a main processor, or as interfaces to peripherals for which no commercial support is available.

Microprocessor-based controllers perform only a very specific function and, as a result, are inherently simple. They could, for example, simply convert an input to an output, as is illustrated in Figure 1. This type of controller was used by Ortega and Palya (1982) to control a Selectric typewriter with ASCII RS232 input. Alternatively, Figure 2 illustrates a controller that could collect data while controlling some process and pass stored data to an up-line computer in response to interrogation such as the system suggested by Reed, Lewart, and Schneider (1981).

Most required interfaces are already provided as single chips that directly connect to the microprocessor. Single chips can provide: general I/O such as digital input and output (6821), analog input (14444), and analog output (DAC 0830); communication protocols such as RS232 (6850) or IEEE 488 (68488); and CPU support facilities such as programmable timing (6840) and math functions (9511).

A prerequisite to the effective use of these controllers is the technology required to design, build, and debug them. Microprocessor design is actually no more demanding than programming solid state modules. The required information is conveniently packaged in Heathkit's individualized learning programs on microprocessors (EE-3401 and EE-3404) and microprocessor interfacing (EE-3402). Heathkit also offers a microprocessor trainer

The authors gratefully acknowledge the contributions of Elizabeth Palya in all phases of this work. Requests for reprints, software, or schematics should be sent to William L. Palya, Department of Psychology, Jacksonville State University, Jacksonville, Alabama 36265.

Figure 1. An example of a simple controller that converts an input to an output.

Figure 2. An example of a controller that could collect data while controlling some process and pass stored data to an up-line computer.

(ET-3400A) that contains the hardware necessary to prototype various controllers, as well as the circuits discussed in their learning program.

The software for a controller is most conveniently developed on a system that provides powerful editing, fast assembly, mass storage, hard copy, and the ability to burn PROMS (Fox, Wolf, & Lesgold, 1979). Cross-assemblers or cross-compilers eliminate the need for a well supported development system based on the target microprocessor by producing code for one machine while running on another. A wide variety of microprocessor cross-assemblers or cross-compilers are commercially available that run on PDP-11s or other large systems and allow the use of any convenient well supported system to be used for microprocessor development.

Unfortunately, debugging a controller either while it is under development or when it fails while in service can be very frustrating. Microprocessor-based systems that are thrashing, are wired wrong, or are completely dead cannot be used to diagnose themselves. As a result, they can be nearly impossible to debug without relatively

expensive test equipment or a great deal of skill. This is because the presence of a high or low signal level on any one line provides little information unless the signal levels on other lines are simultaneously known. The enormity of some tasks is an additional problem. Testing 4,096 bytes of memory, for example, requires providing 4,096 16-bit addresses and inspecting 4,096 8-bit data words.

This paper describes a manual and a microprocessor-based in-circuit emulator (ICE) for 6802, 6808, and 6809 microprocessors. These limited ICEs operate the bus of the system under test as if they were the microprocessor and enable the user to identify faults. However, neither of these emulators is intended to execute programs from the target systems or to operate at normal bus speeds. As a result, they cannot be used in situations that are time critical, with dynamic RAM, DMA subsystems, or in designs that synchronize with an off-chip clock, such as many 6800 designs. Emulators that run at full bus speed and execute target system software are commercially available but are generally too expensive to be cost-effective for psychology laboratories.

The use of an ICE is straightforward. The microprocessor chip is removed from the system under test, and a flat cable with a 40-pin plug is used to attach the ICE in place of the microprocessor. Each line of the 40-pin plug can be set to a one or a zero, or its current state can be read by the ICE. In this way, the user can inspect and manipulate any element in the system under test in the same way as the microprocessor.

Inexpensive but very slow in-circuit emulation can be done manually with a bank of switches and a logic probe. Figure 3 provides the circuit necessary to control the bus of 6802, 6808, or 6809 with simple toggle switches.

A manual emulator is straightforward to use. The desired address is set in the address switches, the appropriate control lines are toggled as specified by the cycle-by-cycle description of that microprocessor, the data lines are then set (or read with a logic probe), and finally, the control lines are again toggled appropriately. Care must be taken in some emulation states. For example, data must not remain asserted on the data bus while read/write is high.

Alternatively, a small microprocessor-based emulator

Figure 3. Circuit for a manual ICE using toggle switches.

Figure 4. Simplified block diagram of the microprocessor-based emulator.

can be used to simplify the debugging process. The microprocessor writes to or reads from the bus of the system under test in response to commands entered on a CRT terminal. Figure 4 provides a simplified block diagram of the microprocessor-based emulator. The ICE contains a 6808 microprocessor, 4-KB PROM, 2-KB RAM, an ACIA serial port, and 40 parallel I/O lines. It has been implemented as a 12 x 13 cm printed circuit board and requires an external power supply that provides 1.0-1.5 A of +5 V dc, 100 mA of +12 V dc, and 100 mA of -12 V dc.

The firmware in the ICE configures the I/O lines to emulate those of the microprocessor being replaced. The user can request that information be placed on the address, data, or control lines, or that information read from those lines be displayed and continuously updated on the CRT terminal. With these commands, the user can examine the state of various lines in the system under test (e.g., decode logic) or inject signals onto those lines and observe whether or not the signal was received at the proper pin of the "microprocessor."

The ICE also provides the firmware necessary to examine and/or modify locations in response to simple commands entered from the CRT terminal. Typing an address will display the contents of that address. The contents can then be modified by subsequently entering the desired data. Consecutive locations can be examined by typing "line feed" to increment through memory or "up arrow" to decrement through memory.

Several diagnostic programs have also been constructed. Three memory diagnostics (simple, address shorts, and pattern sensitivity) are available to test a specified block of memory. The ACIA diagnostic writes characters to a specified ACIA output and attempts to read the same character from that ACIA input. This test is successful with a functioning ACIA if the input and output are shorted together either at the ACIA itself or

at the RS232 plug. This diagnostic also reports the status of the error bits in the ACIA, such as DCD and CTS. The PIA diagnostic either outputs user-supplied data with the specified PIA or displays and continuously updates the input bit pattern from that PIA. The functionality of the PIA can be tested by observing the outputs with a logic probe or shorting inputs to constant-current voltage or ground. A memory dump utility reads a block of memory and displays it on the terminal. This utility can be used to confirm the contents of a PROM.

The architecture of the firmware is illustrated in Figure 5. The ICE initially requests which microprocessor to emulate. Subsequently, it can run in either the direct line manipulation mode (entered with #) or the diagnostic mode (entered with @). Each mode supports a variety of commands. The direct line manipulation mode uses a table of appropriate PIA bytes and bits to identify the requested line for each microprocessor (it can be expanded to support other microprocessors). The diagnostic mode uses an "init," a "read," and a "write" routine for each microprocessor (also expandable). The diagnostic routines use these three interface routines to do all I/O.

Figure 5. Architecture of the ICE firmware. Both the direct manipulation mode and diagnostic mode are illustrated.

The schematic for the microprocessor-based emulator is given in Figure 6.

Figure 6. Schematic diagram of the microprocessor-based ICE.

REFERENCES

- FOX, J. L., WOLF, R. V., & LESGOLD, A. M. Microprocessor uses in a time-shared facility. *Behavior Research Methods & Instrumentation*, 1979, 11, 271-280.
- ORTEGA, J., & PALYA, W. L. An IBM I/O Selectric typewriter controller. *Behavior Research Methods & Instrumentation*, 1982, 14, 99-102.
- REED, A. V., LEWART, D., & SCHNEIDER, L. H. Low-cost microcomputer networking: How to get high throughput on a low budget. *Behavior Research Methods & Instrumentation*, 1981, 13, 221-226.

APPENDIX

When powered up or reset, the ICE requests the processor type. Processor types currently supported are 6802/6808, 6809, and 6809E (DLM only). After the processor type has been entered, the diagnostic monitor prompts with "@." The following commands are supported.

MT0 aaaa,bbbb

Test memory from address aaaa to address bbbb. This is a nondestructive test. Memory errors are reported in the form:

AAAA BB CC

where AAAA = memory address where error occurred

BB = expected data

CC = received data

MT1 aaaa,bbbb

Test memory from address aaaa to address bbbb. Looks for blocks of memory mapped over other blocks (address line shorts). If there are no errors, memory is cleared.

MT2 aaaa,bbbb

Test memory from address aaaa to address bbbb. Inserts a sequence of random numbers into memory and then reads it back. May find some tricky pattern sensitivities in static RAM.

MD aaaa,bbbb

List the contents of addresses from aaaa to bbbb on the CRT.

,P

Prompt the user for the addresses of the PIA that will be tested via the P command. (See PRF, PRS, PW below.)

,U

Prompt the user for the addresses of the ACIA that will be tested via the U command. (See U1 and UR below.)

PRF

Read the two data registers from the specified PIA and print their contents to the CRT. This is repeated continuously until this test is aborted with an "@." Individual lines on the PIA can be tested by tying them high or low.

PRS

Read PIA status as above, except that the IRQ1 and IRQ2 contents are also printed. If an IRQ is detected high, this test pauses so that the transition can be seen. This is repeated continuously until this test is aborted with an "@."

PW bbbbbbbbbbbbbbbb or PW%HH (b = binary, H = hex)

Write the provided data to the specified PIA. The individual outputs of the PIA can then be tested with a logic probe.

UT

Test the specified ACIA. A sequence of the 256 possible bytes is sent out the ACIA and incoming data are printed on the CRT. All error conditions are reported. A loop-back test is assumed, but not necessary. Another character is transmitted after a time-out regardless of errors. Errors reported are: no DCD, no CTS, parity error, framing error, overrun error, and character sent not character received.

UR

Read serial data from the ACIA and print them on the CRT. This test reports the same error conditions as does UT.

aaaa

Open the address aaaa and display the contents on the CRT. The user can enter new data if the contents are to be changed.

(lf) (LINE FEED)

Read the next location and display it as above.

(↑) (UP ARROW)

Read the previous location.

(/) (SLASH)

Reread the current location.

(#) (NUMBER SIGN)

Abort any diagnostic, enter the direct line manipulation mode.

(@) (AT SIGN)

Abort any diagnostic.

The direct line manipulation mode is a low-level debug tool that assumes the user has an understanding of the hardware in the microprocessor system being tested. The user can request combinations of signals that are destructive. The commands are as follows.

AR

Continually read and display the address bus status. The user can then short address lines to ground or +5. Faults such as shorted lines or opens can then be detected.

AW aaaa

Write address aaaa to the address bus. A logic probe can be used with this function to find shorted lines or to check address decoding.

DR

Continually read and display the data bus.

DW dd

Write the data dd to the data bus.

CR

Read and display the control lines.

CW

Prompt for individual control line output data.

(#) (NUMBER SIGN)

Abort any mode.

(@) (AT SIGN)

Abort any mode, return to diagnostic emulation mode.