e-PG Pathshala
Subject : Computer Science
Paper: Embedded System
Module: Emulators and Debuggers
Module No: CS/ES/37
Quadrant 1 – e-text

In this module we will discuss about the debuggers and its tools. Simulator, Emulator and In-circuit emulators used in embedded system development will also be discussed. At the end emulator hardware will be discussed.

1. Debugger

A Debugger or debugging tool is a computer program that is used to test and debug other programs. The code to be examined should be running on an instruction set simulator to identify the fault in the code because the software problem cannot be identified when we are running it on the original hardware. The debugger can be used to identify if the program is running correctly, and identify the cause of failure when it fails. The debugger may be a source-level debugger, or a low-level debugger. If it is a source-level debugger, the debugger can show the actual position in the original code, when the program crashes. If it is a low-level debugger or a machine-language debugger, it shows that line in the program. Catching run-time errors is not as obvious. Most embedded systems do not have a “screen”. Hence we cannot find the run time errors as in general software development.

1.1 Debugging an Embedded System

Debugging an embedded system is similar to debugging a host based application. Many embedded systems are not possible to debug unless they are operating at full speed. Hence debugging of an embedded system uses host computer.

The debugger can exist as two pieces, a debug kernel in the target and a host application that communicates with it and manages the source database and symbol tables.

1.1.1 Requirements for Debugging

There are three requirements for debugging an embedded or real-time system. They are Run control, Memory substitution and real time analysis.

I. Run control is the ability to start, stop, peek, and poke the processor and memory.
II. Memory substitution is replacing ROM-based memory with RAM for rapid and easy code download, debug, and repair cycles.
III. Real-time analysis is following code flow in real time with real-time trace analysis.

1.2 Model Debugging System

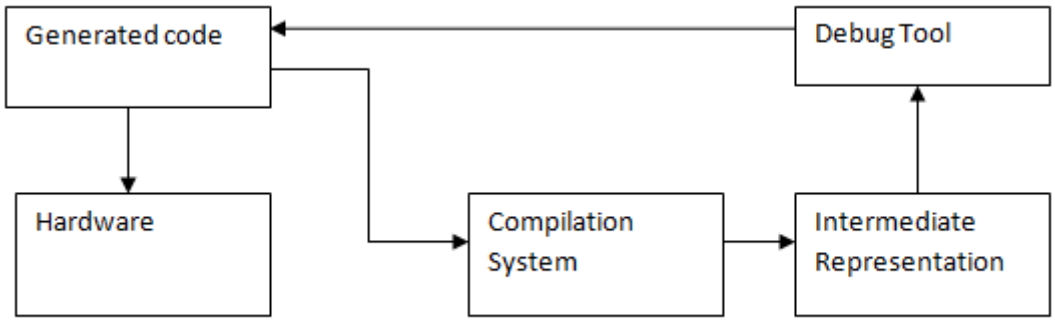

Figure 1. Model Debugging system

Figure 1 shows a model of a debugging system. The data path from the debugging tool represents symbol table information that allows the mapping of machine level information to source level constructs. Any debugging system has at least two processes executing:

- Test program and
- The debugger.

One Part of the debugger runs on the host, and the other on the target machine. To make the debugging system non-intrusive, we need to execute code only at breakpoint (breakpoint is an intentional stopping or pausing in the program) and run debugger as a separate process and provide separate execution unit to execute debugger.

1.2.1 Debugging Tools

Debugging is an essential step in the embedded system development process. Figure 2 shows the different debugging options. It can be done with simulators, In-circuit emulators and using remote target processors.

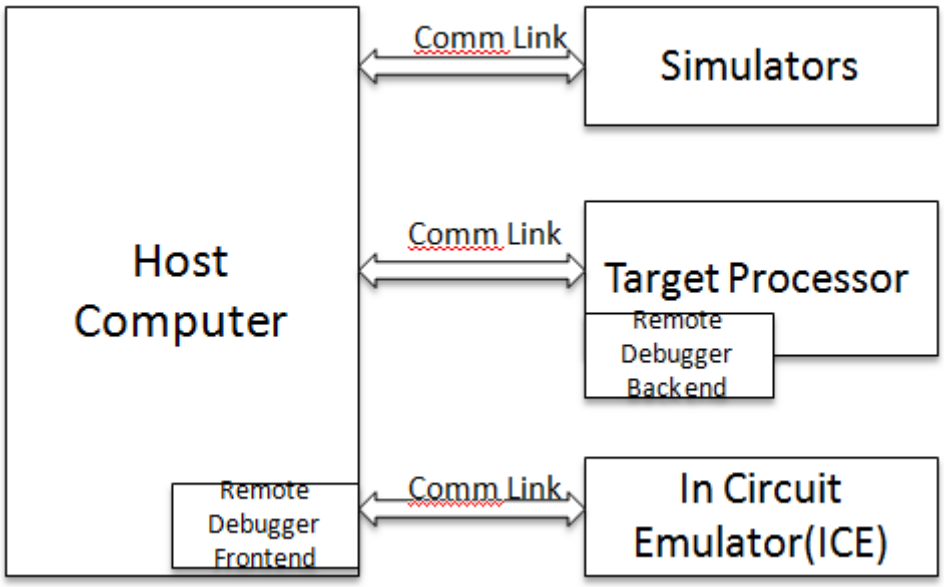

Figure 2 : Debugging tools

1.2.2 Debugging with simulators

Simulator is a host-based program that simulates functionality and instruction set of target processor. The front-end has text or GUI-based windows for source code, register contents, etc. Simulators are valuable during early stages of development. Disadvantage of this method is that, it only simulates the processor, and not the peripherals.

1.2.3 Debugging with Remote Debuggers

Remote Debugger is used to monitor/control embedded SW. It is used to download, execute and debug embedded software over a communications link. The program running on the host of a remote debugger has a user interface that looks just like any other debugger that you might have used. The main debugger screen is usually either a command-line interface or graphical user interface (GUI). GUI debuggers typically contain several smaller windows to simultaneously show the active part of the source code, current register contents, and other relevant information about the executing program. Note that in the case of embedded systems, the debugger and the software being debugged are executing on two different computer systems. The front-end has text or GUI-based windows for source code, register contents, etc. Backend provides low-level control of target processor, runs on target processor and communicates to the front-end over a communication link. Debugger and software being debugged are executing on two different computer systems. It supports higher level of interaction between host and target. It allows

-Start/restart/kill, and stepping through program.

- Software breakpoints.

- Reading/writing registers or data at specified address.

Remote debuggers are one of the most commonly used downloading and testing tools during development of embedded software. This is mainly because of their low

cost. Embedded software developers already have the requisite host computer. In addition, the price of a remote debugger does not add significantly to the cost of a suite of cross-development tools (compiler, linker, locator, etc.). However, there are some disadvantages to using a debug monitor, including the inability to debug startup code. Another disadvantage is that code must execute from RAM. Disadvantage of this system is that it requires a target processor to run the final software package.

1.2.4 Debugging with(In Circuit Emulator) ICE

In-Circuit Emulator (ICE) takes the place of the target processor. It contains a copy of target processor, plus RAM, ROM, and its own embedded software. It allows you to examine the state of the processor while the program is running. It uses the remote debugger for human interface. It supports software and hardware breakpoints. It has real-time tracing. It stores the information about each processor cycle which is executed. It allows you to see in what order things happen. ICE provides greater flexibility, ease for developing various applications on a single system in place of testing that multiple targeted systems. Disadvantage of this method is that, it is expensive.

Emulation refers to the ability of a computer program or electronic device to imitate another program or device. An emulator is a piece of hardware/software that enables one computer system to run programs that are written for another computer system. An in-circuit emulator (ICE) provides a lot more functionality than a remote debugger. In addition to providing the features available with a remote debugger, an ICE allows you to debug startup code and programs running from ROM, set breakpoints for code running from ROM, and even run tests that require more RAM than the system contains. The ICE is itself an embedded system, with its own copy of the target processor, RAM, ROM, and embedded software. In-circuit emulators are usually expensive. But they are powerful tools, and help significantly for debugging. Like a debug monitor, an emulator uses a remote debugger for its host interface. In some cases, it is even possible to use the same debugger frontend for both. But because the emulator has its own copy of the target processor, it is possible to monitor and control the state of the processor in real time. This allows the emulator to support such powerful debug features as hardware breakpoints and real-time tracing.

With a debug monitor, you can set breakpoints in your program. Emulators, by contrast, also support hardware breakpoints. Hardware breakpoints allow you to stop execution in response to a wider variety of events, not only instruction fetches, but also interrupts and reads and writes of memory. Typically, an emulator incorporates a large block of special-purpose RAM that is dedicated to storing information about each processor cycle executed. Another type of debug tool similar to an ICE is a background debug mode (BDM), or JTAG debugger. JTAG debuggers are typically less expensive than in-circuit emulators but offer much of the same functionality.

A circuit for emulating target system remains independent of a particular targeted system and processor.

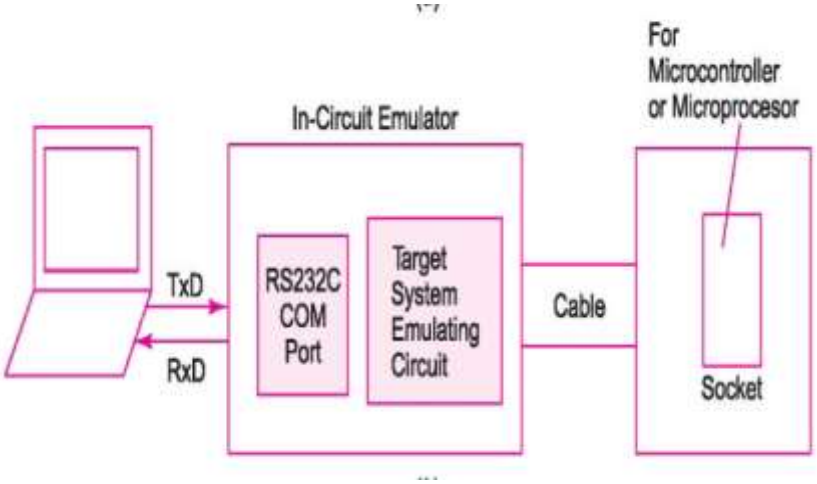

Figure.3 In Circuit Emulator(ICE) Diagram

Figure.3 shows an In Circuit Emulator(ICE) Diagram. ICE interfaces the COM port of a computer. It emulates target Microcontroller(MCU) IOs. The ICE socket connects MCU externally. It uses computer developed object files and hex files for the MCU. It uses debugger at the computer developed files for the MCU application.

2.1 Difference in Emulator and ICE

Emulator uses the circuit consisting of the microcontroller or processor itself. The Emulator emulates the target system with extended memory and with code downloading ability during the edit-test-debug cycles. ROM Emulator emulates only a ROM. ICE uses another circuit with a card that connects to target processor through a socket.

2.1.1 Hardware

Many hardware emulators are available in the field of development of embedded systems. In this module we will see 'Net ROM' as an example of hardware emulator(figure 4).

Figure 4. Net ROM

NetROM is an example of a general class of tools called Emulators. From the point of view of the target system, the ROM emulator is designed to look like a standard ROM device. A ROM emulator is a hardware-assist device. The term hardware-assist refers to additional specialized devices that supplement a software-only debugging solution. ROM emulator has a connector that has the exact mechanical dimensions and electrical characteristics of the ROM it is emulating. RAM can be written quickly via a separate channel from a host computer. However, the connector is used to bring the signals from the ROM socket on the target system to the main circuitry. This circuitry provides high-speed. Thus, the target system sees a ROM device, but the software developer sees a RAM device. This RAM can have its code easily modified and allows debugger breakpoints to be set.

3.Benefits and drawbacks of the Debugging with Emulator

Emulators are the better debugging systems, which have better graphics quality and additional features than original hardware. It saves states also. Emulators maintain the original look, feel, and behavior of the embedded system. Even though the cost of developing an emulator is high, it proves to be the more cost efficient solution over time. Emulators allow software exclusive to one system to be used on another. It is more difficult to design emulators and it also requires better hardware than the original system.

4. Summary

In this module we discussed about Debugging and its tools. Emulator and In-Circuit-Emulator(ICE) have been explained with their merits and demerits. An Emulator Hardware example, NetROM, is discussed at the end.

5. References

1. Arnold S. Berger," Embedded Systems Design: An Introduction to Processes, Tools, and Techniques", CMP books, 2002.
2. Michael Barr, Ambony massa, "Programming Embedded Systems, Second Edition with C and GNU Development Tools", second edition, O'REILLY publications, 2006.
3. Raj Kamal, Publication: McGraw-Hill, "Embedded Systems - Architecture,Programming and Design",Second edition, 2008.