Help

A digital multimeter, as shown in Figure 2.7, is a typical embedded system. This embedded system has two inputs: the mode selection dial on the front and the red/black test probes. The output is a liquid crystal display (LCD) showing measured parameters. The large black chip inside the box is a microcontroller. The software that defines its very specific purpose is programmed into the ROM of the microcontroller. As you can see, there is not much else inside this box other than the microcontroller, a fuse, a rotary dial to select the mode, a few interfacing resistors, and a battery.

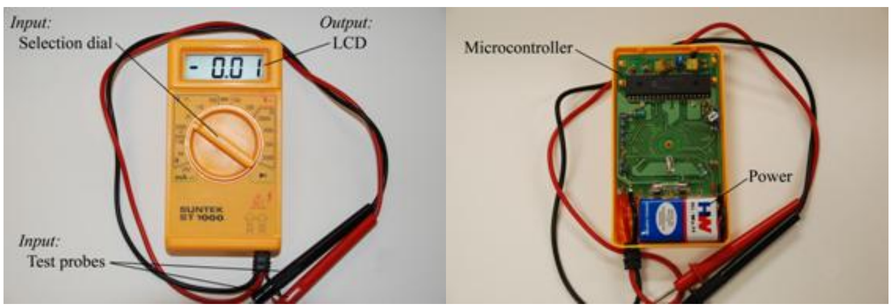

Figure 2.7. A digital multimeter contains a microcontroller programmed to measure voltage, current and resistance.

CHECKPOINT 2.28

What is a microcomputer?

Hide Answer

A microcomputer is a small computer that includes a processor, memory and I/O devices.

As defined previously, a microcomputer is a small computer. One typically restricts the term **embedded** to refer to systems that do not look and behave like a typical computer. Most embedded systems do not have a keyboard, a graphics display, or secondary storage (disk). There are two ways to develop embedded systems. The first technique uses a microcontroller, like the ARM Cortex M-series. In general, there is no operating system, so the entire software system is developed. These devices are suitable for low-cost, low-performance systems. The book Embedded Systems: Real-Time Operating Systems for ARM Cortex-M Microcontrollers (http://users.ece.utexas.edu/%7Evalvano/arm/outline3.htm) describes how to design a real-time operating system for the Cortex M family of microcontrollers. On the other hand, one can develop a high-performance embedded system around a more powerful microcontroller such as the ARM Cortex A-series. These systems typically employ an operating system and are first designed on a development platform, and then the software and hardware are migrated to a stand-alone embedded platform.

The external devices attached to the microcontroller allow the system to interact with its environment. An **interface** is defined as the hardware and software that combine to allow the computer to communicate with the external hardware. We must also learn how to interface a wide range of inputs and outputs that can exist in either digital or analog form. In this class we provide an introduction to microcomputer programming, hardware interfacing, and the design of embedded systems. The book Embedded Systems: Real-Time Interfacing to ARM Cortex-M Microcontrollers (http://users.ece.utexas.edu/%7Evalvano/arm/outline.htm) on the details of hardware interfacing and system design. The book Embedded Systems: Real-Time Operating Systems for ARM Cortex-M Microcontrollers (http://users.ece.utexas.edu /%7Evalvano/arm/outline3.htm) describes real-time operating systems and applies embedded system design to real-time data acquisition, digital signal processing, high-speed networks, and digital control systems. In general, we can classify I/O interfaces into parallel, serial, analog or time. Because of low cost, low power, and high performance, there has been and will continue to be an advantage of using time-encoded inputs and outputs.

A **device driver** is a set of software functions that facilitate the use of an I/O port. One of the simplest I/O ports on the Stellaris® microcontrollers is a parallel port or General Purpose Input/Output (GPIO). One such parallel port is Port A. The software will refer to this port using the name **`GPIO_PORTA_DATA_R`**. Ports are a collection of pins, usually 8, which can be used for either input or output. If Port A is an input port, then when the software reads from **`GPIO_PORTA_DATA_R`**, it gets eight bits (each bit is 1 or 0), representing the digital levels (high or low) that exist at the time of the read. If Port A is an output port, then when the software writes to **`GPIO_PORTA_DATA_R`**, it sets the outputs on the eight pins high (1) or low (0), depending on the data value the software has written.

Help

The other general concept involved in most embedded systems is they run in **real time**. In a real-time computer system, we can put an upper bound on the time required to perform the input-calculation-output sequence. A real-time system can guarantee a worst case upper bound on the response time between when the new input information becomes available and when that information is processed. This response time is called interface **latency**. Another real-time requirement that exists in many embedded systems is the execution of periodic tasks. A periodic task is one that must be performed at equal-time intervals. A real-time system can put a small and bounded limit on the time error between when a task should be run and when it is actually run. Because of the real-time nature of these systems, microcontrollers have a rich set of features to handle many aspects of time.

CHECKPOINT 2.29

An input device allows information to be entered into the computer. List some of the input devices available on a general purpose computer.

Hide Answer

Typical input devices include the keys on the keyboard, mouse and its buttons, touch pad, DVD reader, and microphone. USB drives, Ethernet, and wireless can be used for input and output.

CHECKPOINT 2.30

An output device allows information to exit the computer. List some of the output devices available on a general purpose computer.

Hide Answer

Typical output devices include the LEDs on the keyboard, monitor, printer, DVD burner, and speaker. USB drives, Ethernet, and wireless can be used for input and output.

The embedded computer systems will contain a Texas Instruments TM4C123 microcontroller, which will be programmed to perform a specific dedicated application. Software for embedded systems typically solves only a limited range of problems. The microcomputer is embedded or hidden inside the device. In an embedded system, the software is usually programmed into ROM and therefore fixed. Even so, **software maintenance** (e.g., verification of proper operation, updates, fixing bugs, adding features, extending to new applications, end user configurations) is still extremely important. In fact, because microcomputers are employed in many safety-critical devices, injury or death may result if there are hardware and/or software faults. Consequently, testing must be considered in the original design, during development of intermediate components, and in the final product. The role of simulation is becoming increasingly important in today's market place as we race to build better and better machines with shorter and shorter design cycles. An effective approach to building embedded systems is to first design the system using a hardware/software simulator, then download and test the system on an actual microcontroller.

In summary, embedded systems have these properties:

- A microcontroller hidden inside
- A dedicated purpose
- Run in real time
- Input/output is important
- High volume, low cost
- Extremely reliable
- Low power
- Small size and weight

Help