Section 30.7 Embedded Computers in Electronic Instruments

Tim Mikkelsen, Agilent Technologies

30.7.1 Introduction

All but the simplest electronic instruments have some form of embedded computer system. More and more computing is being embedded in the instrument because of reductions of computing cost and size. This is increasing computing power as well as increasing the number of roles for computing in the instrument domain. Consequently, systems that were previously a computer or PC and an instrument are now just an instrument.

This transition is happening because of the demand for functionality, performance and flexibility in instruments and also because of the low cost of microprocessors. Embedded computers are almost always built from microprocessors or microcontrollers. In fact, the cost of microprocessors is sufficiently low and their value is sufficiently high that most instruments have more than one embedded computer.

30.7.2 Embedded Computer Model

The instrument and its embedded computer normally interact with four areas of the world: the measurement, the user, peripherals and external computers. The instrument needs to receive measurement input and/or send out source output. A *source* is defined as an instrument that generates or synthesizes signal output. An *analyzer* is an instrument that analyzes or measures input signals. These signals can consist of analog and/or digital signals. The *front end* of the instrument is the portion of the instrument that conditions, shapes and modifies the signal to make it suitable for acquisition by the analog-to-digital converter.

The instrument normally interacts with the user of the measurement. The instrument also generally interacts with an external computer that is usually connected for control or data-connectivity purposes. Finally, the instrument is often connected to local peripherals, primarily for printing and storage. Shown in figure 30.7.1 is a generalized block diagram for the embedded computers in an instrumentation environment.

(Source: Fig. 10.1)

Figure 30.7.1 An embedded computer generalized block diagram.

The embedded computer is typically involved in the control and transfer of data via external interfaces. This enables the connection of the instrument to external PCs, networks, and peripherals. Examples include *IEEE 488* (also known as *GPIB* or *HP-IB*), *RS-232* (serial), *Centronics* (parallel), *Universal Serial Bus* (USB) and *IEEE 1394* (FireWire).

The embedded computer is also typically involved in the display to and input from the user. Examples include *keyboards*, *switches*, *rotary pulse generators* (RPGs, i.e, knobs), *LEDs* (single or alpha-numeric displays*), LCDs*, *CRT*s, and *touch screens*.

Many instruments often have a large amount of configuration information because of their advanced capabilities. The embedded computer enables saving and recalling of the instrument state. Also, the embedded computer is used sometimes for user customization of the instrument. This can range from simple configuration modifications to complete instrument programmability.

With very powerful embedded computers available in instruments, it is often unnecessary to connect the instrument to an external computer for more-advanced tasks. Examples include *go/no-go,* also known as *pass/fail* testing and data logging.

The embedded computer almost always performs calculations, ranging from very simple to very complex, that convert the raw measurement data to the target instrument information for measurement — or vice versa for source instruments. The embedded computer generally controls the actual measurement process. This can include control of a range of functions, such as analog-to-digital conversion, switching, filtering, detection and shaping. The embedded computer is almost always used to perform at least a small amount of self-testing. Most instruments use embedded computers for more extensive calibration tests.

30.7.3 Benefits of embedded computers in instruments

In addition to the direct uses of embedded computers, it is instructive to think about the value of an embedded computer inside an instrument. The benefits occur throughout the full life cycle of an instrument, from development through maintenance.

One of the biggest advantages of embedding computers within an instrument is that they allow several aspects of the hardware design to be simplified. In many instruments, the embedded computer participates in acquisition of the measurement data by servicing the measurement hardware. Embedded computers also simplify the digital design by providing mathematical and

logical manipulations, which would otherwise be done in hardware. They also provide calibration both through numerical manipulation of data and by controlling calibration hardware. This is the classic transition of function from hardware to software.

The embedded computer allows for lower manufacturing costs through effective automated testing of the instrument. They also are a benefit because they allow for easier and lower-cost defect fixes and upgrades (with a ROM or program change).

When used as a stand-alone instrument, embedded computers can make the set-up much easier by providing on-line help or set-up menus. This also includes automatic or user-assisted calibration. Although many are stand-alone instruments, a large number are part of a larger system. The embedded computers often make it easier to connect an instrument to a computer system by providing multiple interfaces and simplified or automatic set up of interface characteristics.

30.7.3.1 Support circuitry. Although requirements vary, most microprocessors require a certain amount of support circuitry. This includes the generation of a system clock, initialization hardware and bus management. In a conventional design, this often requires 2 or 3 external integrated circuits (ICs) and 5 to 10 discrete components. The detail of the design at this level depends heavily on the microprocessor used. In complex or high-volume designs, an *Application-Specific Integrated Circuit* (ASIC) can be used to provide much of this circuitry.

30.7.3.2 Memory. The microprocessor requires memory both for program and data store. Embedded computer systems usually employ both ROM and RAM.

Read-Only Memory (ROM) retain their contents if power is removed.

Random Access Memory (RAM) is a historical, but inadequate, term that really refers to read/write memory, memory whose contents can be changed. RAM memory is volatile; it will lose its contents when power is no longer applied. RAM is normally implemented as either static or dynamic devices.

Static memory is a type of electrical circuit that will retain its data, with or without access, as long as power is supplied. It is normally built with latches or flip-flops.

Dynamic memory is built out of a special type of circuit that requires periodic memory access (every few milliseconds) to refresh and maintain the memory state. It uses a switched capacitor for the storage element. This is handled by memory controllers and requires no special attention by the developer.

The advantage of the dynamic memory RAM is that it consumes much less power and space. ROM is used for program storage because the program does not usually change after power is supplied to the instrument. A variety of technologies are used for ROM in embedded applications:

Nonvolatile memory . Some instruments are designed with special nonvolatile RAM that retains its contents after power has been removed. This is necessary for storing such information as calibration and configuration data. This can be implemented with regular RAM memory that has a battery backup. It can also be provided by special nonvolatile memory components, most commonly, flash memory devices.

Flash memory is a special type of EEPROM that uses block transfers — instead of individual bytes — and has a fairly slow, in computer terms, write time. So, it is not useful as a general read/write memory device, but is perfect for nonvolatile memory purposes. Also, a limited number of writes are allowed (on the order of 10,000). All embedded systems have either a ROM/RAM or a flash/RAM memory set so that the system will be able to operate the next time the power is turned on.

30.7.4 lnstrument hardware

Given that the role of these microprocessors is instrumentation (measurement, analysis, synthesis, switches, etc.), the microprocessor needs to have access to the actual hardware of the instrument. This instrument hardware is normally accessed by the microprocessor like other peripheral components — such as registers or memory locations. Microprocessors frequently interface with the instruments' analog circuits using *analog-to-digital converters* (ADCs) and *digital-to-analog converters* (DACs).

In an analog instrument, the ADC bridges the gap between the analog domain and the digital domain. In many cases, substantial processing is performed after the input has been digitized. Increases in the capabilities of ADCs enable the analog input to be digitized closer to the front end of the instrument, allowing a greater portion of the measurement functions to occur in the embedded computer system. This has the advantages of providing greater flexibility and eliminating errors introduced by analog components.

Just as ADCs are crucial to analog measuring instruments, DACs play an important role in the design of source instruments — such as signal generators. They are also very powerful when used together. For example, instruments can have automatic calibration procedures where the

embedded computer adjusts an analog circuit with a DAC and measures the analog response with an ADC.

30.7.5 Physical Form of the Embedded Computer

Embedded computers in instruments take one of three different forms: a separate circuit board, a portion of a circuit board, or a single chip. In the case of a separate circuit board, the embedded computer is a board-level computer that is a circuit board separate from the rest of the measurement function. An embedded computer that is a portion of a circuit board contains a microprocessor, its associated support circuitry and some portion of the measurement functions on the same circuit board. A single-chip embedded computer can be a microcontroller, digital signal processor or microprocessor core with almost all of the support circuitry built into the chip.

30.7.5.1 Digital signal processor (DSP). A DSP is a special type of microcontroller that includes special instructions for digital signal processing, allowing it to perform certain types of mathematical operations very efficiently. These math operations are primarily *multiply and accumulate* (MAC) functions, which are used in filter algorithms.

30.7.5.2 Microprocessor cores. These are custom microprocessor, IC segments or elements that are used within custom-designed ICs. Assume an instrument designer has a portion of an ASIC that is the CPU core. The designer can then integrate much of the rest of the system, including some analog electronics, on the ASIC, creating a custom microcontroller. This approach minimizes size and power. In very high volumes, the cost can be very low. However, these chips are intended for very specific applications and are generally difficult to develop.

30.7.5.3 Architecture of the embedded computer instrument. Just as an embedded computer can take a variety of physical forms, there are a number of ways to configure an embedded computer instrument. The architecture of the embedded computer has significant impact on many aspects of the instrument including cost, performance, ease of development and expandability. The range of choices include:

- Peripheral-style instruments (externally attached to a PC)
- PC plug-in instruments (circuit boards inside a PC)
- Single-processor instruments
- Multiple-processor instruments
- Embedded PC-based instruments (where the embedded computer is a PC)

- Embedded workstation-based instruments

30.7.6 Embedded Computer System Software

As stated earlier, the embedded computer in an instrument requires both hardware and software components. Embedded computer system software includes:

- Operating system - The software environment that the instrument applications run within.
- lnstrument application - The software program that performs the instrument functions on the hardware
- Support and utility software - Additional software the user of the instrument requires to configure, operate, or maintain the instrument —such as reloading or updating system software, saving and restoring configurations

30.7.7 User Interfaces

Originally, instruments used only panel-mounted, *direct controls* that were connected directly to the analog and digital circuits. As embedded computers became common, instruments began employing *menu* or *keypad-driven* systems, in which the user input was read by the computer, which then modified the circuit operation. Today, designs have progressed to the use of *Graphical User Interfaces* (GUls). Although some instruments are intended for automated use or are *faceless* (have no user interface), most need some way for the user to interact with the measurement or instrument. All of these user interface devices can be mixed with direct control devices — such as meters, switches and potentiometers/knobs. There are a variety of design challenges in developing effective user interfaces in instruments.

30.7.8 External Interfaces

Most instruments include external interfaces to a *peripheral*, another instrument device or to an external computer. An interface to a peripheral allows the instrument to use the external peripheral, normally printing or plotting the measurement data. An interface to another device allows the instrument to communicate with or control another measurement device. The computer interface provides a communication channel between the embedded computer and the external computer. This allows the user to:

- Log (capture and store) measurement results
- Create complex automatic tests
- Combine instruments into systems

- Coordinate stimulus and response between instruments

The external computer accomplishes these tasks by transferring data, control, set-up, and/or timing information to the embedded computer. At the core of each of these interfaces is a mechanism to send and receive a stream of data bytes.

30.7.8.1 Hardware Interface Characteristics

Each interface that has been used and developed has a variety of interesting characteristics, quirks, and tradeoffs. However, external interfaces have some common characteristics to understand and consider:

- Parallel or serial - how is information sent (a bit at a time or a byte at a time)?
- Point to point or bus/network - how many devices are connected via the external interface?
- Synchronous or asynchronous - how is the data clocked between the devices?
- Speed -what is the data rate?

Probably the most fundamental characteristic of hardware interfaces is whether they send the data stream one bit at a time, *serial,* or all together, *parallel*. Most interfaces are serial. The advantage of serial is it limits the number of wires down to a minimum of two lines (data and ground). However, even with a serial interface, additional lines are often used (transmitted data, received data, ground, power, request to send, clear to send, etc.). Parallel interfaces are normally 8 bit or 16 bit. Some older instruments had custom *Binary Coded Decimal* (BCD) interfaces, which usually had six sets of 4-bit BCD data lines. Parallel interfaces use additional lines for handshaking-explicit indications of *data ready* from the sender and *ready for data* from the receiver.

30.7.9 Software Protocol Standards

The software protocol standards listed in Table 30.7.1 provide the hardware interface between devices and computers. The *physical layer* is necessary, but not sufficient. To actually exchange information, the devices (and/or computers) require defined ways to communicate, called *protocols*. If a designer is defining and building two or more devices that communicate, it is possible to define special protocols (simple or complex). However, most devices need to communicate with standard peripherals and computers that have predefined protocols that need to be supported. The protocols can be very complex and layered (one protocol built on top of another). This is especially true of networked or bus devices.

Table 30.7.1 Common Instrument Software Protocols

Software Protocol	Description
The IEEE 488.2	The *IEEE 488.2, Codes, Formats, Protocols and Common Commands for Use with IEEE 488.1* is a specification that defines: 39 common commands and queries for instruments, the syntax for new commands and queries, and a set of protocols for how a computer and the instrument interact in various situations. Although this is a companion to the IEEE 488.1 interface, it is independent of the actual interface, but it does depend on certain interface characteristics.
SCPI	The *Standard Commands for Programmable lnstruments* (SCPI) is a specification of common syntax and commands so that similar instruments from different vendors can be sent the same commands for common operations. It also specifies how to add new commands that are not currently covered in the standard,
TCP/IP	The *Transmission Control Protocol/lnternet Protocol* (TCP/~P) specification is the underlying protocol used to connect devices over network hardware interfaces. The network hardware interface can be a LAN or a WAN.
FTP	The *File Transfer Protocol* (FTP) specification is a protocol used to request and transfer files between devices over network hardware interfaces.
HTTP	The *HyperText Transfer Protocol* (HTTP) specification is a protocol used to request and transfer Web (HyperText Markup Language, HTML) pages over network hard\~are interfaces.
VXI-11	The *VXI-11 plug-and-play specification* is a protocol for communicating with instruments that use the VXI-bus (an instrument adaptation of the VME bus), GPIB/HP-IB, or a network hardware interface.

(Source: TABLE 10.15)

30.7.8 Using Embedded Computers

Instruments normally operate in an analog world. It has characteristics, such as noise and nonlinearity of components, that introduce inaccuracies in the instrument. Instruments generally deal with these incorrect values and try to correct them by using software in the embedded computer to provide calibration (adjusting for errors inside the instrument) and correction (adjusting for errors outside of the instrument).

30.7.8.1 Using instruments that contain embedded computers. In the process of selecting or using an instrument with an embedded computer, a variety of common characteristics and challenges arise. This section covers some of the common aspects to consider:

- Instrument customization - What level of instrument modification or customization is needed?
- User access to the embedded computer - how much user access to the embedded computer as a general-purpose computer is needed?
- Environmental considerations - what is the instrument's physical environment?
- Longevity of instruments - how long will the instrument be in service?

30.7.8.2 User access to an embedded computer. As described earlier, many instruments now use an embedded PC or workstation as an integral part of an instrumentation system. However, the question arises: "Is the PC or workstation visible to the user?" This is also related to the ability to customize or extend the system. Manufacturers realize benefits from an embedded PC because there is less software to write and it is easy to extend the system (both hardware and software). The manufacturers realizes these benefits — even if the PC is not visible to the end user.

If the PC is visible, users often prefer an embedded PC because it is easy to extend the system, the extensions (hardware and software) are less expensive, and they don't require a separate PC. However, there are problems in having a visible, embedded PC. For the manufacturer, making it visible exposes the internal architecture. This can be a problem because competitors can more easily examine their technologies. Also users can modify and customize the system. This can translate into the user overwriting all or part of the system and application software. This is a serious problem for the user, but is also a support problem for the manufacturer. Many instrument manufacturers that have faced this choice have chosen to keep the system closed, and not visible to the user, because of the severity of the support implications.

The user or purchaser of an instrument has a choice between an instrument that contains a visible embedded PC and an instrument that is "just" an instrument, independent of whether it contains an embedded PC.

It is worth considering how desirable access to the embedded PC is to the actual user of the instrument. The specific tasks that the user needs to perform using the embedded PC should be considered carefully. Generally, the instrument with a visible embedded PC is somewhat more expensive.