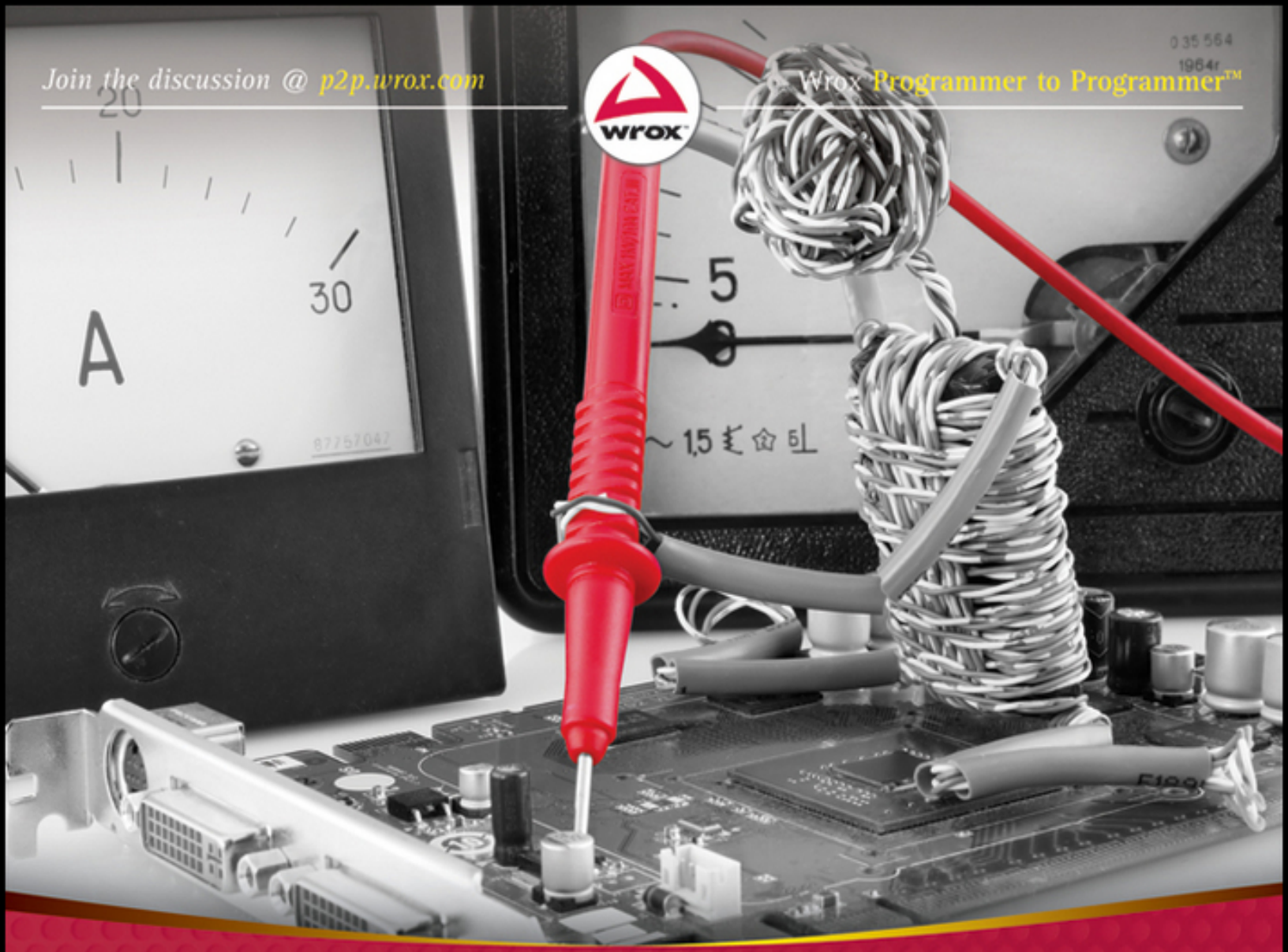
Join the discussion @ p2p.wrox.com



Wrox Programmer to Programmer™

Professional Windows® Embedded Compact 7

Foreword by Mike Hall, *Principal Software Architect at Microsoft*

Samuel Phung, David Jones, Thierry Joubert

CONTENTS

PART I

Introducing Embedded Development

CHAPTER 1: Embedded Development
CHAPTER 2: Windows Embedded Compact 7
CHAPTER 3: Development Station Preparation
CHAPTER 4: Development Process
CHAPTER 5: Development Environment and Tools

Chapter 1

Embedded Development

WHAT'S IN THIS CHAPTER?

- Defining an embedded device
- Using software for an embedded device
- Establishing key elements for embedded development

Embedded development has been around for decades. The terms such as *embedded system* and *embedded computer* are widely used by marketing professionals across multiple industries. However, the actual meaning and representation for the term *embedded* is still vague.

Although it's not within this book's objective to delve into the definition for the term embedded, you need to understand a general boundary for the embedded device, embedded software, and development environment relevant to Windows Embedded contents in this book.

WHAT IS AN EMBEDDED DEVICE?

When referring to an embedded device, some of you may still think of the small devices, typically built with a microcontroller with limited processing capability and memory. Contrary to this thinking, many embedded devices in today's market are built with a powerful processor, abundant memory, and storage.

Some of the current embedded devices are built with computing technology that can rival an enterprise class

server from just a few years ago. Not too long ago, enterprise servers were built with processors that operate in the sub gigahertz (GHz) range, with system memory in the 100 megabyte (MB) range and storage in the gigabyte (GB) range.

Today, many of you use smartphones built with a GHz processor, 512MB or more system memory, and 8GB to 32GB of storage.

As technology rapidly advances and enables more powerful processor modules with more integrated features to be built in a smaller footprint at a lower cost, it enables a new generation of consumer, industrial, medical, robotics, education, and other devices to be built with better and innovative features that can deliver these devices to the market with a higher perceived value at a lower cost.

Today, using available technology, embedded devices can be built with a broad range of capabilities, with processors ranging from low-power, 8-bit microcontrollers with limited memory to powerful processors with CPU clocks operating in the GHz, memory in the GB, and storage in the 100-GB range. With some imagination and creativity, the possibilities for embedded devices are endless.

Similarity to Personal Computer

From a general architecture point of view, an embedded device has many similarities to a typical personal computer (PC).

- It has a processor.
- It has system memory.
- It has storage to store software.
- It requires software applications to be useful.

Difference from Personal Computer

Although a general purpose PC is designed to enable the user to install different operating systems and software applications to perform different tasks, an embedded device is built with preconfigured software, designed to perform a specific set of tasks and functions.

Table 1-1 shows some of the differences between the PC and embedded device.

TABLE 1-1: Personal Computer and Embedded Device Comparison

PERSONAL COMPUTER	EMBEDDED DEVICE
Support 1024×768 or better display resolution	Headless (does not support displays) Smaller displays that support limited resolution LCD modules that display ASCII text.
Keyboard and mouse to capture user input	Handful of hardware buttons and touch screens to capture user input
Abundant system memory, common for PC to equip with 2GB or more RAM	Limited system memory
Abundant storage, common for PC to equip with 300GB or larger hard disk for storage	Flash storage with limited storage capacity

Specialized Purpose Device

An embedded device is a specialized purpose device designed to serve a specific purpose and built to meet designated specifications and cost objectives.

For some markets, the same hardware used to build a general purpose PC can be used to build a specialized purpose embedded device; therefore, making the distinction between an embedded device and personal computer vague.

Following are two separate application scenarios that use similar hardware. One of them is categorized as a general purpose computer and the other as a specialized purpose device:

- A small retail store owner uses a computer as a point-of-sale terminal. In addition, this store owner uses the same computer to send and receive e-mail using Outlook, to write business letters using Word, and to

install additional software onto the PC to perform accounting and record keeping-related tasks.

- A major department chain store uses computers as point-of-sale terminals for each of its branch locations. To minimize service and support issues and to simplify management tasks, the computers are configured to perform only point-of-sale-related tasks. The computers are also configured to not enable additional software to be installed and limit access to its system to prevent existing software accidentally being removed.

In the first preceding scenario, the store owner uses the computer as a general-purpose PC. In the second scenario, the department chain store uses the computer as a specialized-purpose device.

Example of Embedded Devices

Embedded devices are all around you. Think about your daily living, when you travel, visit a theme park, interact with a financial institution, use entertainment devices in your home, drive your automobile, and so on.

Following are examples of some of the embedded devices in today's market:

- Mobile phone
- Set-top-box
- Television
- Media entertainment system
- Printer
- Portable media player
- GPS navigation device
- Credit card processing terminal
- Automated ticketing machine
- Digital camera
- Medical instrument
- Engineering instrument

- Network router
- Information kiosk
- Automated teller machine
- Video projector
- Self-serve checkout station at your local super market

WHAT IS EMBEDDED SOFTWARE?

While software applications for general purpose PCs are designed to function on a broad range of computers, built by different manufacturers that meet general requirements (such as processor speed, available memory, and storage), software developed for the embedded device is intended for one specific model or category of devices.

Comparing to software for general purpose PCs, following are the main differences for embedded software:

- Designed to operate on hardware with limited resources.
- Application codes are tightly coupled with hardware.
- Errors and exceptions can’t be thrown to the user.

Programming Languages and Principles

Other than the different development considerations, which will be explored later in this chapter, similar programming languages and principles apply to developing Windows applications for a desktop PC and embedded applications for a Windows Embedded Compact device.

If you are developing desktop PC applications using Visual Studio 2005 and 2008, you are already familiar with the Visual Studio application development environment for Windows Embedded Compact. With little more effort, you

can easily adapt existing Visual Studio experience to develop Windows Embedded Compact applications.

Programming Discipline

When developing applications for a general-purpose desktop PC, you can make the following assumptions about the PC, without considering the end user's environment in detail:

- It's equipped with a 1.0 GHz or faster processor.
- It has at least 1GB or more of system memory.
- It has a hard disk with abundant storage space, in the 100GB range.
- The video output is capable of supporting a 1024×768 or better display resolution.
- A keyboard and mouse are used to capture user input.
- A network connection is available.

When developing applications for an embedded device, you cannot make any of the preceding general assumptions and must have details and accurate information about the device's features and capability. You also need to have a clear understanding about the device's operating environment and how the end users interact with the device. The embedded device may be built with the following features:

- Headless without a user interface
- Limited system memory and storage
- Battery powered
- A few hardware buttons to capture user input

You also may be required to develop applications for an embedded device to meet one or more of the following design objectives:

- Minimize power consumption to extend operating time for battery-powered devices.

- Minimize memory and resources leakage for devices that operate 24/7.
- Meet the hard real-time characteristic for the timing critical device that requires the application to perform required tasks within a specified time slot.
- Prevent file corruption resulting from unexpected loss of power to the device.
- Develop applications to directly access and control the device's hardware.
- Develop applications to access a headless device (built without a user interface) remotely for service and maintenance purposes.
- Develop applications to perform self-diagnostic functions and implement automated routines to correct errors found.

Seasoned embedded developers develop the discipline to account for additional design considerations from their experience, which may include technical issues, user interactions, and an operating environment related to the final product. Often, these additional design considerations are not part of the specifications and requirements; however, in many cases, these additional design considerations are critical to the project's success.

Specialized Purpose Application

Although software for the PC is designed to operate on all desktop and portable notebook computers that meet a general technical specification, embedded software is a specialized-purpose application designed to run on one particular class or category of specialized-purpose device.

Software for the PC is designed to enable the end user to install and remove the application at will. However, embedded software is usually shipped as a preinstalled component on the device and is designed to limit the end

user’s ability to remove the software from the device or to make changes.

DEVELOPMENT CONSIDERATIONS

Embedded development skill is a discipline that cumulates and improves over time, with active engagement and hands-on involvement in the actual embedded development projects.

Operating systems, programming languages, and the hardware platforms are tools used by the developer to design, compose, and build embedded devices.

Whether working on a PC or an embedded device development project, you can have similar development concerns and needs, such as the following:

- Firmware
- Operating system
- Hardware adaptation codes
- Device drivers for peripherals
- File system
- Network protocol stack
- Codecs
- Support libraries
- Application

An embedded device development project can involve any or all these concerns. To be effective, an embedded developer needs to have a good understanding about the device’s operating environment, how the user uses the device, the hardware platform, and design objectives.

Different categories of embedded device design objectives are quite different, such as the following:

- A consumer-oriented device needs to meet the targeted performance at the least cost, sufficient for the device to meet the 1-year to 3-year warranty period.
- In addition to meeting the targeted function requirements, many embedded devices designed for industrial applications need to meet strict quality requirements, operate 24/7, and survive a harsh operating environment that involves a wide operating temperature range from −40°C to +85°C and exposure to high humidity and chemical conditions.

In general, when working on an embedded development project, in addition to the design specification, the development team needs to consider the following:

- Hardware
- Operating environment
- User environment

Hardware

In addition to the direct impact to the cost of manufacturing, the selected hardware can affect the development schedule and engineering cost. When selecting the hardware, you need to consider the following:

- Is the required hardware readily available in the market?
- Are there sufficient components available in the market and engineering resources to develop customized hardware for the project?
- Is the selected hardware’s processor architecture supported by the selected operating system?
- Can the hardware vendor provide support for the selected operating system?
- Does the selected hardware provide the best value from an overall project perspective? (Lower-cost hardware may require additional development, have limitations that create other cost centers, raise the overall project

cost, and significantly impact the project's time-to-market schedule.)

Operating Environment

To develop a good product, you need to understand how the product is used and the environment the product needs to operate in. You need to take into account the following considerations, which may impact the hardware requirements:

- How can the embedded device be used?
- What temperature range is the device expected to operate in?
- Can the device operate 24/7?
- Can the device be deployed on an automobile, a vessel, or an airplane?
- Can the device be subjected to vibration and shock during operation?
- Can the device be subjected to strong electrostatic shock during operation?
- Can the device be placed in an outdoor environment?

User Environment

User expectation is one of the most important factors. If the embedded device does not meet user expectation, the user is not likely to purchase or use the device.

Feasibility

From an engineering perspective, with sufficient resources and time, the development team can engineer a perfect device. In real life, all development projects are bounded by the following:

- Limited development resources and budget.

- Development must be completed within a predetermined schedule.

After all the technical, environment, and user requirements are met, the product development team also needs to consider the required resources and time needed to successfully complete all required development tasks. These considerations can have a strong impact on the business' cost, profit, and time-to-market and can influence whether to move the project forward.

SUMMARY

Embedded development is an engineering discipline that involves multiple technical skills, covering both hardware and software. As technology rapidly changes and evolves, the embedded development environment will continue to change, adopt new technology, and create new ways to do things.

A career in the embedded development field can be challenging and rewarding at the same time. It's a challenge to learn and adapt rapidly to changing technologies. It's also rewarding to work with a broad range of technology to create cool devices that can help solve challenging problems.

Chapter 2

Windows Embedded Compact 7

WHAT'S IN THIS CHAPTER?

- Introducing Windows Embedded Compact
- Exploring new features in Windows Embedded Compact 7
- Understanding a little bit of history
- Seeing what you do with Windows Embedded Compact
- Choosing Windows Embedded Compact

With the first version released in 1996, the Windows Embedded Compact family of technology has been through more than 15 years in the making. Evolving through seven major versions, with countless hours of development, bug fixes, improvements, and enhancements, this latest version is solid, packed with features, and optimized to enhance performance and security.

To help you better understand Windows Embedded Compact, this chapter provides a brief overview of Windows Embedded Compact 7, the market it serves, and some of the key features.

WHAT IS WINDOWS EMBEDDED COMPACT?

Windows Embedded Compact is not binary-compatible with any version of the desktop Windows operating system (OS) and is not a scaled-down version of a desktop Windows OS.