1

Windows Embedded CE

While the traditional Windows desktop operating system (OS) developed by Microsoft was designed to run on well-defined and standardized computing hardware built with the x86 processor, Windows Embedded CE was designed to support multiple families of processors.

This chapter provides an overview of CE and improvements for the latest Windows Embedded CE 6.0 R2 release. Multiple Windows Embedded products, including Windows Embedded CE, are being promoted and supported by the same business unit within Microsoft, the Windows Embedded Product group.

What Is Embedded?

Embedded is an industry buzz word that's been in use for many years. Although it's common for us to hear terms like embedded system, embedded software, embedded computer, embedded controller, and the like, most developers as well as business and teaching professionals have mixed views about the embedded market.

To fully understand the potential offered by the Windows Embedded product family, we need to have good understanding about what's considered an embedded device and embedded software.

Before getting into talking about Windows Embedded products, let's take a brief look at what embedded hardware and software are.

Embedded Devices

A computer is an electronic, digital device that can store and process information. In a similar fashion, an embedded device has a processor and memory and runs software.

While a computer is designed for general computing purposes, allowing the user to install different operating systems and applications to perform different tasks, an embedded device is generally developed with a single purpose and provides certain designated functions. Often the embedded device is designed as a closed system and does not allow applications from other developers to run on the system.

Throughout our daily lives, each of us interacts with multiple embedded devices. Here's a listing of some of the more common ones:

- Telephone, car phone, and mobile phone
- VCR, video CD player, DVD player/recorder, digital video recorder/player
- Remote control for TV, audio system, DVD player, garage door opener, security systems for automobiles and other devices
- Credit card reader, cash register, and self-service kiosk
- Digital camera, camcorder, digital photo frame, and gaming console
- Fax machine, copy machine, and printer

The above list contains only a few of the more common devices. It would take a much longer list to cover all embedded devices.

Embedded Software

Many software development projects use similar programming languages, such as the C language, with different operating systems and running on different types of hardware.

Fundamentally, the overall software development process should be the same regardless of the type of project. In an ideal situation, a competent project manager should evaluate the project thoroughly and establish the best possible development process to accomplish the tasks. Developers should use due diligence and apply their best efforts to reduce development time and to develop efficient code that can run on computing hardware in a way that minimizes resources used and cost of the product.

In the real world, it's the business manager's responsibility to deliver more profit while lowering costs. To the development team, this translates into having to solve more difficult problems with fewer resources.

The key difference between developing an embedded application and developing an application for the desktop computer has to do with the specifications for the application.

Application for Desktop Computer

When developing applications for the desktop computer, the developer expects the application will run on a computer with at least a monitor, keyboard, and mouse. The monitor is expected to be able to display VGA or higher display resolution. In the current market, even the cheapest monitor can support a 1024 × 768 display resolution.

In general, when developing application for the desktop computer, the developer doesn't need to give much consideration to the user interface or the display monitor, keyboard and mouse. In addition, the developer can safely assume that the typical computer will have a 1.0 Ghz or faster processor with

512 MB or more in system memory; this assumption is used in measuring the application performance needed to meet certain user requirements.

In a nutshell, when developing applications for the desktop computer, the developer can make a general assumption about the target hardware needed for running the application.

Application for Embedded Device

When developing applications for embedded devices, the developer must study and understand all the hardware features available, and how the end user will interact with and use the device.

Many embedded devices do not have a display monitor. For those built with display, the screen size is very small, with low resolution. This is true of such devices as the Windows Mobile smartphone, GPS navigation devices, and portable media players. Most embedded devices are not designed with keyboard and mouse to capture user input, and they use a limited number of special function keys to process user input.

To minimize cost, most embedded devices are built with slower processors and less system memory than desktop computers.

Thus when developing applications for embedded devices, the developer must consider the limited user interface and limited system memory as well as the slower processor. In addition to developing efficient code to maximize the user's experience, the developer also has to take great care to avoid memory leaks. Many of these embedded devices are on constantly once they go into service; examples are the Windows Mobile smartphone, security monitoring systems, and network appliances. Even a very small amount of memory leakage will accumulate over time and become a serious flaw in the device.

Embedded Devices and Software

A desktop application that consumes 30 to 50 MB of system memory is not considered to be using a lot of memory, when the whole system has 1 GB or more. But an embedded device may have only 64 MB of system memory to be shared between RAM and the file system. An application that consumes more than 1 MB of memory in this scenario is considered to be using a very big chunk of memory. But there also are embedded devices equipped with high performance processors and huge amounts of system memory in the industrial automation and aerospace industries.

In summary, we cannot classify embedded devices based on processor speed, memory resources, or product footprint. An embedded device is designed with embedded software to provide a set of designated primary functions, and will perform these primary function throughout its useful life.

The Windows Embedded Family

The Windows Embedded CE, Windows XP Embedded, and Windows Embedded Point of Service operating systems are products of Microsoft's Windows Embedded product group. Developers new to CE may be thinking it's a scaled-down version of the Windows desktop OS. But while CE shares some common development tools with the desktop Windows OS, the CE kernel is distinctively different from the desktop Windows OS.

I provide a brief overview of the other Windows Embedded products below to help you better understand the differences between them.

Windows XP Embedded

Windows XP Embedded (XPe) is a componentized version of the Professional edition of Windows XP, with some added features designed specifically to support the embedded device market. Microsoft announced a new name for this product, Windows Embedded Standard, on April 15, 2008.

Using a development tool called Target Designer and a properly configured design template, OS, device drivers, and application components, the developer can build a componentized XPe image with a significantly smaller footprint than the Professional edition of Windows XP.

The XPe image can be configured to boot from the network, removable USB storage, or CD-ROM. This OS includes the Write Filters component. When implemented in XPe, the Write Filters feature can configure the storage partition as Read Only to support devices subject to frequent unexpected loss of power. Without the Write Filters, Windows XP system shutoff due to unexpected loss of power might corrupt the file system and prevent it from functioning normally.

Since XPe is binary-compatible with the Professional edition of Windows XP, applications written for Windows XP can work with XPe with the proper components configured for the OS.

Windows Embedded Point of Service

The Windows Embedded Point of Service (WEPOS) is designed for point-of-service devices in the retail and hospitality markets that need to connect to a variety of peripherals, such as the following:

- ATMs for banking transactions
- Full-featured point-of-sale and point-of-service terminals
- Kiosks

WEPOS can be installed from a CD and does not require a development tool to generate the OS runtime image as XPe does. WEPOS is binary-compatible with the Professional edition of Windows XP.

Windows Embedded CE 6.0

CE is not binary-compatible with any version of the desktop Windows OS. Other than the common *Windows* term in the product name, Windows Embedded CE is not a scaled-down version of the desktop Windows OS and does not share the common desktop OS kernel.

CE is not a port from the desktop Windows. It is an embedded OS designed from the ground up to support a new generation of small-footprint, smart, connected, and service-oriented embedded devices. It was designed to support embedded devices with less system memory, less storage space, and a slower processor compared to the desktop PC. Since its inception in 1996, each new version of this OS has been improved with added features while maintaining the following design criteria:

- **Small Footprint** — Scalable footprint depends on selected components.
- **Modular Architecture** — Componentized operating system enables the OEM to make decisions about components to be included in the final runtime image.
- **Real-Time Support** — Provides bounded, deterministic response times.
- **Supports Broad Range of Hardware** — x86, ARM, MIPS, and SH-4
- **Efficient Power Management** — Provides uniform power management libraries.
- **Efficient Development Tools** — Applications can be developed with C/C++, C#, and Visual Basic using the Visual Studio Integrated Development Environment.
- **Efficient Debugging and Testing Tools** — These include the Windows Embedded CE Test Kit (CETK); CoreCon, which is a connectivity framework providing connectivity between the VS2005 development workstation and CE device; Kernel Independent Transport Layer (KITL); and Remote Tools.

Before the current CE 6.0 version, the earlier version of CE was limited to supporting 32 concurrent running processes and could address only 32 MB of virtual memory in each process. The latest version, CE 6.0, has been redesigned to remove these limits. The latest OS kernel can support up to 32,000 simultaneous processes and can access up to 2 GB of virtual memory in each process. The new kernel also includes an improved file system that supports larger storage media, larger individual file size, and encryption for removable media.

On April 15, 2008, Microsoft announced that it would change the Windows Embedded CE product name to *Windows Embedded Compact* for the next release of this OS.

Modular and Compact OS

Windows Embedded CE is a highly modular operating system. Each CE runtime image is made up of a collection of OS components, selectable from the platform builder development tool's component library. The collection of OS components consists of device drivers, hardware interfacing libraries, programming libraries, networking libraries, applications, and other software technology frameworks.

The device manufacturers, using the platform builder, can develop and configure a customized CE designed to include only the needed components to support the hardware and the application on the device. All unnecessary components are excluded from the final OS runtime image to yield the smallest possible image size.

The platform builder tool and the associated OS design wizard provide an intuitive integrated development environment to help create the initial OS design with help from the OS design wizard and a library of preconfigured design templates. The platform builder provides a component library listing all available components to further configure the OS design.

With a smaller OS runtime footprint, it takes less system memory, less storage, and fewer processor resources to run. Devices with smaller OS run time take less time to boot up, which helps provide an efficient device. By minimizing the OS footprint, the device can be built with less expensive hardware.

Real-Time Operating System

Windows Embedded CE is a hard real-time operating system, providing reliable core services to support embedded system designs that demand low-latency, deterministic, real-time system performance.

CE has the following features required by a real-time system:

- **Preemptive Multithreading** — Determines when a context switch should occur.
- **Prioritized Thread Scheduling** — Uses a priority-based time-slice algorithm to schedule threads.
- **Priority Inversion Prevention** — When a lower-priority thread is sharing the same resource with a higher-priority thread, priority inversion can occur when the lower-priority thread and the higher-priority thread are competing for the same resource.
- **Predictable Thread Synchronization** — When multiple threads compete for resources, it's necessary to manage and synchronize thread priority. Otherwise, priority inversion can occur.

There are hard real-time and soft real-time systems. A soft real-time system can miss its bounded time response, missing the timing deadline once in a while, and still maintain a reasonable level of acceptable performance. For example, a Voice over IP (VoIP) device may delay the delivery of voice packets once in a while and end up dropping some of the packets because of network traffic loading, but still provide acceptable performance. A hard real-time system cannot miss any of its bounded time responses. When a hard real-time system misses a bounded time response, it causes major system failure. Imagine what happens when an automobile's electronic brake system fails to engage in a timely manner while the automobile is traveling at high speed and needs to make an urgent stop to avoid a collision. In a real-time system, the bounded time response means that the system must respond to service an event, such as the interrupt, within a maximum allowable time defined by the system. Otherwise, a major failure will occur.

Supported Hardware

Windows Embedded CE is designed to run on hardware built with x86, ARM, MIPS, and SH4 processors.

Business statistics indicate that only about 2 percent of the total number of microprocessors are used to build desktop PCs. The other 98 percent are used to build non-PC devices for the embedded market.

The ability to support four processor families enables Windows Embedded CE to reach a much broader market than the PC market and creates broader employment opportunities for the "embedded" developer and development of new types of devices.

Most hardware vendors have reference platforms with the necessary device drivers, Board Support Package (BSP), and Software Development Kit (SDK) to support CE. The BSP is a collection of device drivers and OEM Adaptation Layer (OAL) code for the hardware.

Microsoft maintains a long list of embedded processor boards and systems with BSP available to support Windows Embedded CE.

> **The following URL provides a list of hardware with BSPs for the OS.**
>
> `http://msdn.microsoft.com/en-us/embedded/aa714506.aspx`
>
> **If this URL is broken, search** `www.microsoft.com` **using the "Windows CE" and "supported BSP" keywords.**

New Features in CE 6.0 R2

About a year after the initial CE 6.0 release, CE 6.0 R2 was released with additional features and improvements. CE 6.0 R2 is delivered as an overlay to the originally installed CE 6.0. For the development workstation that already has CE 6.0 installed, the CE 6.0 R2 installation will keep the existing BSPs and OS designs previously created. The CE 6.0 R2 installation incorporates all the Quick Fix Engineering (QFE) and updates that have been released for CE 6.0. Following are the key features added for CE 6.0 R2:

- The flash driver and Secure Digital controller drivers have been improved. The Secure Digital controller driver has been updated to support the Secure Digital 2.0 specification. The Secure Digital 2.0 provides better performance and enables faster speeds for SDIO modules. It also supports higher-capacity memory cards, up to 32 GB.
- A new USB smart-card reader driver has been added to support the USB Chip/Smart Card Interface Devices Specification.
- The ATAPI storage driver has been upgraded to include support for the Serial ATA disk controller.
- The BIOSLoader has been upgraded to overcome the 2-GB limit of the old FAT16 file partition.
- A Pluggable Font technology has been added to enable a third-party font engine to be used.
- Three Board Support Packages (BSPs) were added — an x86-based HP/Compaq t5530 thin-client BSP, an SH4-based ST7109 BSP, and an ARMV4I-based Marvel PXA270 BSP.
- The Terminal Services Client has been improved. RDP 5.2 has been upgraded to RDP 6.0, which brings the RDP stack for Windows Embedded CE in line with the stack used for Windows Vista. RDP 6.0 also provides support for spanning a remote desktop session across multiple local displays. The RDP 6.0 update also provides significant improvement in security through Secure Socket Layer (SSL), Transport Layer Security (TLS), Network Level Authentication (NLA), and Server Authentication.
- Internet Explorer (IE) 6.0 has improved significantly. It has been updated to provide better security and performance. The IE team back-ported some of the algorithms to increase performance for IE 7.0 to IE 6.0 for Windows Embedded CE.
- Added support for Web Services on Device (WSD). *WSD* is a Microsoft implementation of the Devices Profile for Web Services standard. WSD provides a method for a discovery protocol to take place between new devices being attached to the network and devices already on the network.

Customizable UI

Many devices built on top of the Windows Embedded CE technologies don't present themselves as CE devices. Often, the users of these devices don't know that they're using a CE device, even after years of usage.

An OEM device built with Windows Embedded CE typically powers up to a custom splash screen, launches a custom application at start-up, and provides the user interface unique to the device, without showing any of the standard Windows desktop or UI.

The Windows Embedded CE's User Interface (UI) is customizable by the OEM to create a unique look and feel for the CE device. Sample UI skins with source code are provided along with the platform builder installation.

> **The Zune from Microsoft is built on top of Windows Embedded CE. The Zune is being sold as a portable MP3 Video player. A majority of Zune users don't know that the Zune application is running on CE.**

Wired and Wireless Connectivity

Windows Embedded CE enables you to build scalable wired and wireless devices that connect mobile and embedded devices into existing infrastructures.

CE provides broad wireless support for Personal Area Networks (PANs), Local Area Networks (LANs), and Wide Area Networks (WANs), including Bluetooth, IrDA, and 802.11 WiFi.

CellCore, a set of wireless mobile communication service components, added to the CE 6.0 release, provides the following added features to Windows Embedded CE:

- **Radio Interface Layer (RIL)** — Handles the communication between CellCore and the radio hardware.
- **Telephony** — The Telephony programming elements that are applicable to CellCore, which include Extended TAPI (ExTAPI), Assisted TAPI, and Telephony Service Provider (TSP) API.
- **Wireless Application Protocol (WAP) API** — The WAP API is an open standard for wireless communication to access the Internet from the mobile device.
- **Short Message Service (SMS) Providers** — The SMS is a communication protocol for sending short text messages between mobile devices.
- **Subscriber Identity Module (SIM) Management** — The SIM is a small smart card, containing identification and other data, used in the mobile phone.

Using CE, you can remotely authenticate, authorize, administer, and update new applications and operating system services on the device.

Graphics and Multimedia

Windows Embedded CE includes graphics and multimedia technology similar to the desktop, provides multimedia streaming capabilities to the device, and supports the various protocol and streaming formats required for audio and video playback of either local files or streamed data over a network connection.

A large set of graphics and multimedia components are included with CE:

- AlphaBlend API
- Direct3D, DirectDraw, DirectShow
- DVD-Video, DVR Engine
- Still Image Encoders and Decoders
- Multiple Monitor Support
- G.711, GSM 6.10 and IMA ADPCM Audio Codec
- MP3, MPEG-1 Layer 1 and 2 Audio Codec, MP4, WMA, WMA, WMV Codec
- Windows Media Player OCX 7
- Digital Rights Management (DRM)
- Streaming Media Playback

Windows Embedded CE's graphic and multimedia components enable device manufacturers to build a networked media device, a television set-top box, media player, multimedia electronic picture frame, digital video recorder, and more.

Multilanguage, International Localization

Windows Embedded CE provides Multilingual User Interface (MUI), locale-specific support, Input Method Manager (IMM), and Input Method Editor (IME) to support devices localized for different international markets.

Windows Embedded CE provides an efficient development environment for the device manufacturer to develop the initial device in one language and be able to port the same design with minimum code change for different international markets. This flexibility enables the device manufacture to minimize development cost and shorten development time.

The Windows Embedded CE includes international languages support and can be built and localized for more than 130 different regions. Here is a list of languages it can support:

Arabic	Farsi	Kazah	Slovak
Armenian	Finnish	Konkani	Slovenian
Baltic	French	Korean	Spanish
Basque	FYRO Macedonian	Latvian	Swahili
Belarusian	Galician	Latin	Swedish
Bulgarian	Georgian	Lithuanian	Syriac
Catalan	German	Malay	Tamil
Chinese	Greek	Marathi	Tatar
Croatian	Gujarati	Mongolian	Telugu
Cyrillic	Hebrew	Norwegian	Thai
Czech	Hindi	Polish	Turkish
Divehi	Hungarian	Portuguese	Ukrainian
Dutch	Icelandic	Punjabi	Urdu
English (US)	Indonesian	Romanian	Uzbek
English (Worldwide)	Italian	Russian	Vietnam
Estonian	Japanese	Sanskrit	
Faeroese	Kannada	Serbian	

Real-Time Communication and VoIP

Windows Embedded CE includes Real-Time Communication (RTC) and Voice over IP (VoIP) components, which can be employed for communication using audio and text messaging in real time.

Windows Embedded CE provides an RTC Client API built on the Session Initiation Protocol (SIP), enabling you to build CE devices to make and receive calls from any SIP client. The RTC Client API uses the following industry standard protocols:

- Session Initiation Protocol (SIP)
- Session Description Protocol (SDP)
- Real-Time Transport Protocol (RTP)
- Public Switched Telephone Network/Internet Interworking (PINT)

For the CE 6.0 release, a new set of VoIP components was added to help the OEM take advantage of the CE platform to design and build VoIP devices. In addition to the core VoIP communication component, the VoIP application suite, with source code provided, was added to CE 6.0 to improve functionality. The VoIP application suite includes the following components:

- **Homescreen Application** — Provides the shell functionality as the core of a phone application with features to launch other applications and provide network status, voice-mail and e-mail message status, and system monitoring.
- **Phone Application** — Provides the user interface to dial a phone number to initiate a phone call. The phone application also provides the user interface to the VoIP stack.
- **Settings Application** — Provides the user interface to configure system settings for the VoIP application suite.
- **Bootstrap Application** — This application connects to a server and downloads the provisioning information for the phone.

OS Design Templates

Windows Embedded CE provides a set of OS design templates. An OS design template contains a predefined selection of CE components to use as the starting point for a new OS design to develop a customized CE runtime image. The following OS design templates are included with CE Platform Builder and provide starting points for developing the various devices noted:

- **Digital Media Receiver** — Music and video playback and storage devices
- **Enterprise Terminal** — Thin client, point of sale, point of information, kiosk, and other types of information appliance devices
- **Enterprise Web Pad** — Portable Internet appliance devices with large display screens using a touch screen to capture user input
- **Gateway** — Wireless access points, network routers, and other type of devices requiring Internet connection sharing features
- **Industrial Controller** — Human Machine Interface (HMI) devices for industrial automation and control applications
- **Internet Appliance** — Browser-based Internet appliances using standard keyboards and monitors
- **IP Phone Advanced** — Video phones with QVGA touch screens, using VoIP to save cost, and with advanced provisioning and conference calling
- **IP Phone Basic** — Basic Internet-based telephone devices using VoIP
- **Set-Top Box** — Television set-top boxes able to display Internet and media contents
- **Small-Footprint Device** — Small-footprint devices with a minimal set of system components
- **Windows Network Projector** — Windows network projects. Using the Connect to Windows Network Projector feature, a Windows Vista computer can redirect its display contents to the Windows Network Projector.
- **Windows Thin Client** — Windows-based thin-client terminals

To develop a customized OS runtime image, select the OS design template with the closest features to the type of device you are working with, and add or remove components to meet the design goal.

Developing CE Applications

The Windows Embedded CE operating system is based on the Win32 application programming interface (API). CE is designed to be a compact small-footprint operating system. The CE version of the Win32 API is a subset of the Win32 API for Windows XP and does not include all the function calls available with the Windows XP Win32 API.

CE uses the UTF16 Unicode character encoding standard. This is a common practice for embedded systems design and helps make it easier to port applications to support other languages.

In addition to the Win32 API, CE provides support for the following programming frameworks:

- ❑ Active Template Library (ATL)
- ❑ ActiveX
- ❑ Microsoft Component Object Module (COM)
- ❑ Microsoft Foundation Classes (MFC)
- ❑ .NET Compact Framework

The Windows Embedded CE Platform Builder is a plug-in for the Visual Studio (IDE) to develop OS design, device driver, BSP, and OAL hardware interfacing codes.

While it's possible to develop native code applications using the development environment provided by the platform builder tool, application development using Visual C++, Visual C#, or Visual Basic within Visual Studio provides a better environment to develop both managed and native code application.

Testing and Debugging

Windows Embedded CE provides an effective and easy-to-use testing and debugging environment, enabling the developer to test and debug the OS runtime image and applications running on the actual hardware and to insert breakpoints in the source files to halt program execution while the code is actually running on the hardware. This level of combined hardware and software debugging enables the developer to trace the codes to pinpoint problem areas quickly.

CE provides the following debugging connection and testing tools:

- ❑ Kernel Independent Transport Layer (KITL)
- ❑ CoreCon
- ❑ CE Test Kit (CETK)

The platform builder tool provides the following remote tools that use the KITL connection to perform the debugging function. KITL is used to provide the connection to debug the OS runtime image on the hardware:

- Remote File Viewer
- Remote Heap Walker
- Remote Zoom In
- Remote Process Viewer
- Remote Registry Editor
- Remote System Information
- Remote Performance Monitor
- Remote Spy
- Remote Kernel Tracker
- Remote Call Profiler

When developing CE applications using the Visual Studio 2005 IDE, the CoreCon connection is the mechanism used to establish a link between the development workstation and the CE device. After the CoreCon connection is established, a Visual C++, Visual C#, or Visual Basic application development session can download the compiled application binary to the CE device for debugging. It's possible to insert a breakpoint at the proper line of source code to halt application execution while running on the CE device and enable stepping through the code line-by-line to trace the source code and analyze application behavior.

The CETK tool can run stand-alone or within the platform builder tool. The CETK tool provides the facilities to test the BSP, device driver, and application. Microsoft also uses the same CETK tool to certify the board support package and device driver for its certification program.

I'll work through sample exercises in subsequent chapters covering each of these debugging and testing resources.

What Can Windows Embedded CE Do?

With the correct combination of hardware, Windows Embedded CE is suitable for a broad range of applications.

The CE operating system's real-time capability enables it to be a good platform for time-critical systems where desktop Windows cannot accomplish the goal without acquiring additional expensive third-party software.

The CE operating system can run on low-cost, power-efficient embedded hardware with limited system memory and flash storage, housed in a small enclosure. This can be wall-mounted to function as the main controller for an always-on, low-power, environment-friendly home automation system. While a Windows XP machine may be able to perform these tasks, it is not able to meet the "low-cost," "power-efficient," and "environment-friendly" aspects of the equation.

As computer technology advances, new possibilities are created. Imagine what you could do with a $90, 32-bit embedded controller with the following features:

- 300-MHz Vortex86SX CPU with 128 MB system memory
- 64-MB bootable flash storage with Windows Embedded CE 6.0 installed
- 10/100-Mbps Ethernet interface
- 2 Serial ports
- Parallel port
- 2 USB 2.0 host interface
- 16 general purpose input output (GPIO) control pins
- Powered by a single 5-V DC @ less than 500 mA
- Supporting Visual Basic, C#, and Visual C++ within the Visual Studio 2005 IDE

Using the Vortex86SX controller with some glue logic components and linking them to a few relays, sensors, and motors, you can create a range of interesting, intelligent, and functional applications that can integrate and communicate with your desktop computer, portable computer, or even the mobile phone you carry at all times. Among these applications are:

- Home automation system controller
- Intelligent building or HVAC controller
- Robotic controller
- Security system
- Greenhouse system controller
- Industrial automation controller

Summary

This chapter provides an overview of CE and a brief look at the other products within the Windows Embedded product family. It also covers CE's key features, development environment, and debugging and testing tools.