第 1 章

Windows CE 概述

本章介绍嵌入式系统的概念、嵌入式处理器基本知识、嵌入式操作系统、Windows CE 操作系统和 Windows CE 程序开发工具。本章是全书的引入章节，是学习后面内容的基本准备。

1.1 嵌入式系统的概念

1.1.1 嵌入式系统的由来

嵌入式系统是一个相对模糊的定义，一个手持的 MP3 和一个 PC104 的微型工业控制计算机都可以认为是嵌入式系统。

嵌入式系统已经有了 30 多年的发展历史，它是硬件和软件交替发展的双螺旋式发展的结果。

电子数字计算机诞生于 1946 年，在其后漫长的历史进程中，计算机一直是置于一定的温湿条件的特殊的机房中，价格比较昂贵。直到 20 世纪 70 年代微处理器的出现，计算机才出现了历史性的变化。以微处理器为核心的微型计算机以其小型、价廉、高可靠性等特点，迅速走出机房。基于高速数值计算能力的微型机，表现出的智能化水平引起了控制专业人士的兴趣，要求将微型机嵌入到一个对象体系中，实现对象体系的智能化控制。例如，将微型计算机经电气加固、机械加固，并配置各种外围接口电路，安装到大型舰船中构成自动驾驶仪或轮机状态监测系统。这样一来，计算机便失去了原来的形态与通用的计算机功能。为了区别于原有的通用计算机系统，把嵌入到对象体系中，实现对象体系智能化控制的计算机，称作嵌入式计算机系统。

由于嵌入式计算机系统要嵌入到对象体系中，实现的是对象的智能化控制，因此它有着与通用计算机系统完全不同的技术要求与技术发展方向。通用计算机系统的技术要求是高速、海量的数值计算；技术发展方向是总线速度的无限提升，存储容量的无限扩大。而嵌入式计算机系统的技术要求则是对象的智能化控制能力；技术发展方向是与对象系统密切相关的嵌入性能、控制能力与控制的可靠性。早期，人们勉为其难地将通用计算机系统进行改装，在大型设备中实现嵌入式应用。然而，对于众多的对象系统(如家用电器、仪器仪表、工控单元等)，无法嵌入通用计算机系统，况且嵌入式系统与通用计算机系统的技术发展方向完全不同，因此，必须独立地发展通用计算机系统与嵌入式计算机系统，这

就形成了现代计算机技术发展的两大分支：通用计算机系统与嵌入式计算机系统。通用微处理器迅速从286、386、486到奔腾系列，而嵌入式计算机系统则走上了一条完全不同的道路，这条独立发展的道路就是单芯片化道路。

最早的嵌入式系统是单片机。最早的单片机是Intel公司的8048，它出现于1976年。与此同时，Motorola推出了68HC05，Zilog公司推出了Z80系列，这些早期的单片机均含有256B的RAM、4KB的ROM、4个8位并口、1个全双工串行口、两个16位计时器。之后在20世纪80年代初，Intel又进一步完善了8048，在它的基础上研制成功了8051。

嵌入式系统主要由嵌入式微处理器、外围硬件设备、嵌入式操作系统以及用户的应用程序四个部分组成，它是集软硬件于一体的可独立工作的“器件”。

嵌入式系统通用的概念是：以应用为中心，以计算机技术为基础，软件硬件可裁剪，功能、可靠性、成本、体积、功耗严格要求的专用计算机系统。

从广义上讲，凡是带有微处理器的专用软硬件系统都可称为嵌入式系统。如各类单片机和DSP系统。这些系统在完成较为单一的专业功能时具有简洁高效的特点。但由于它们没有操作系统，管理系统硬件的能力有限，因而在实现复杂多任务功能时，往往困难重重，甚至无法实现。

从狭义上讲，我们更加强调那些使用嵌入式微处理器构成独立系统、具有自己的操作系统、具有特定功能、用于特定场合的嵌入式系统。本书中介绍的嵌入式系统是指狭义上的嵌入式系统。

1.1.2 嵌入式系统的特点

按照历史性、本质性、普遍性的要求，嵌入式系统应定义为：嵌入到对象体系中的专用计算机系统。“嵌入性”、“专用性”与“计算机系统”是嵌入式系统的三个基本要素。对象系统则是指嵌入式系统所嵌入的宿主系统。

与“嵌入性”相关的特点：由于是嵌入到对象系统中，必须满足对象系统的环境要求，如物理环境(小型)、电气环境(可靠)、成本(价廉)等要求。

与“专用性”相关的特点：软、硬件的裁剪性；满足对象要求的最小软、硬件配置等。

与“计算机系统”相关的特点：嵌入式系统必须是能满足对象系统控制要求的计算机系统。与以上两个特点相呼应，这样的计算机必须配置有与对象系统相适应的接口电路。

另外，在理解嵌入式系统定义时，不要与嵌入式设备相混淆。嵌入式设备是指内部有嵌入式系统的产品、设备，如内含单片机的家用电器、仪器仪表、工控单元、机器人、手机、PDA等。

1.1.3 嵌入式系统的种类

按照上述嵌入式系统的定义，只要满足定义中三要素的计算机系统，都可称为嵌入式系统。嵌入式系统按形态可分为系统级(工控机)、主板级(单板、模块)、芯片级(MCU、SoC)。

- 系统级：各种类型的工控器、PC104模块。
- 主板级：各种类型的带CPU的主板及OEM产品。

- 芯片级：各种以单片机、DSP、微处理器为核心的产品。

1.2　嵌入式处理器

嵌入式系统的核心是嵌入式处理器，是控制、辅助系统运行的硬件单元。其范围极其广泛，不仅包括最初的 4 位处理器、目前仍然广泛使用的 8 位单片机，而且包括最新的 16 位、32 位和 64 位的嵌入式处理器。嵌入式处理器一般具备以下 4 个特点。

- 对实时多任务有很强的支持能力，能完成多任务并且有较短的中断响应时间，从而使内部的代码和实时内核的执行时间减少到最低限度。
- 具有功能很强的存储区保护功能。这是由于嵌入式系统的软件结构已模块化，而为了避免在软件模块之间出现错误的交叉作用，需要设计强大的存储区保护功能，同时也有利于软件诊断。
- 具有可扩展的处理器结构，以能最迅速地开发出满足应用的最高性能的嵌入式微处理器。
- 嵌入式微处理器必须功耗很低，尤其是用于便携式的无线及移动的计算和通信设备中靠电池供电的嵌入式系统更是如此，如需要功耗只有 mW 甚至 μW 级。

嵌入式处理器可分为四类：嵌入式微控制器、嵌入式 DSP 处理器、嵌入式微处理器、嵌入式片上系统，见图 1-1。

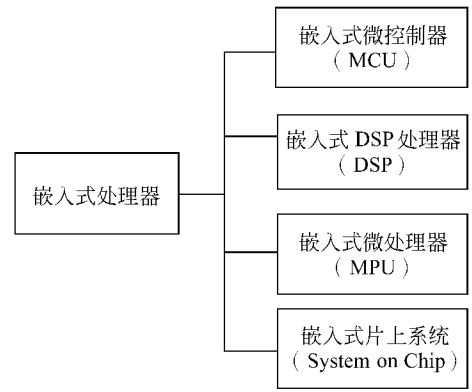

图 1-1　嵌入式处理器的分类

(1) 嵌入式微控制器

嵌入式微控制器(MCU)的典型代表是单片机，这种 8 位的电子器件目前在嵌入式设备中仍然有着极其广泛的应用。单片机芯片内部集成 ROM/EPROM、RAM、总线、总线逻辑、定时/计数器、看门狗、I/O、串行口、脉宽调制输出、A/D、D/A、Flash RAM 等各种必要功能和外设。微控制器的最大特点是单片化，体积大大减小，从而使功耗和成本下降、可靠性提高。微控制器是目前嵌入式系统工业的主流。微控制器的片上外设资源一般比较丰富，适合于控制，因此称为微控制器。由于 MCU 具有低廉的价格、优良的功能，所以其品种和数量最多，比较有代表性的有 MCS-51 系列、MCS-96/196/296、P51XA、C166/167、68K 系列以及 MCU 8XC930/931、C540、C541，并且有支持 I2C、CAN-Bus、LCD 及众多专用 MCU 和兼容系列。Microchips 公司、TI 公司的产品功耗极低，非常适用于电池供电的仪器仪表。近来 Atmel 公司推出的 AVR 单片机由于集成了 FPGA 等器件，所以具有很高的性价比，势必将推动单片机获得更高的发展。

(2) 嵌入式 DSP 处理器

DSP 处理器是专门用于信号处理方面的处理器，其在系统结构和指令算法方面进行了特殊设计，在数字滤波、FFT、谱分析等各种仪器上获得了大规模的应用。DSP 的理论

算法在20世纪70年代就已经出现，但是由于专门的DSP处理器还未出现，所以这种理论算法只能通过MPU等分立元件实现。1982年世界上诞生了首枚DSP芯片，在语音合成和编码解码器中得到了广泛应用。DSP的运算速度进一步提高，应用领域也从上述范围扩大到了通信和计算机方面。目前最为广泛应用的嵌入式DSP处理器是TI的TMS320C2000/C5000系列，另外如Intel的MCS-296和Siemens的TriCore也有各自的应用范围。

(3) 嵌入式微处理器

嵌入式微处理器(MPU)是由通用计算机中的CPU演变而来的。与计算机处理器不同的是，在实际嵌入式应用中，只保留和嵌入式应用紧密相关的功能硬件，去除其他的冗余功能部分，这样就以最低的功耗和资源实现嵌入式应用的特殊要求。和工业控制计算机相比，嵌入式微处理器具有体积小、重量轻、成本低、可靠性高的优点。目前主要的嵌入式处理器类型有Am186/88、386EX、SC-400、Power PC、68000、MIPS、ARM/StrongARM系列等。

(4) 嵌入式片上系统

嵌入式片上系统(System on Chip，SoC)微处理器是一种电路系统，它结合了许多功能区块，将功能做在一个芯片上，像是ARM RISC、MIPS RISC、DSP或是其他的微处理器核心，加上通信的接口单元，像是通用串行端口(USB)、TCP/IP通信单元、GPRS通信接口、GSM通信接口、IEEE 1394、蓝牙模块接口等，这些单元以往都是依照各单元的功能做成一个个独立的处理芯片。SoC是追求产品系统最大包容的集成器件，其最大的特点是成功实现了软硬件无缝结合，直接在处理器片内嵌入操作系统的代码模块。运用VHDL等硬件描述语言不需要再像传统的系统设计一样，绘制庞大复杂的电路板，一点点地连接焊制，只需要使用精确的语言，综合时序设计直接在器件库中调用各种通用处理器的标准(SOPC)，然后通过仿真之后就可以直接交付芯片厂商进行生产。由于SoC往往是专用的，所以不为大部分用户所知，如Philips的Smart XA、Siemens的TriCore、Motorola的M-Core、某些ARM系列器件，以及Echelon和Motorola联合研制的Neuron芯片等。

1.3 嵌入式操作系统

1.3.1 嵌入式系统发展过程中的嵌入式操作系统

综观嵌入式技术的发展，大致经历了以下4个阶段。

第一阶段是以单芯片为核心的可编程控制器形式的系统，同时具有与监测、伺服、指示设备相配合的功能。这种系统大部分应用于一些专业性极强的工业控制系统中，一般没有操作系统的支持，通过汇编语言编程对系统进行直接控制，运行结束后清除内存。这一阶段系统的主要特点是：系统结构和功能都相对单一，处理效率较低，存储容量较小，几乎没有用户接口。由于这种嵌入式系统使用简便、价格很低，以前在国内工业领域应用较为普遍，但是已经远远不能适应高效的、需要大容量存储介质的现代化工业控制和新兴的信息家电等领域的需求。

第二阶段是以嵌入式 CPU 为基础、以简单操作系统为核心的嵌入式系统。这一阶段系统的主要特点是：CPU 种类繁多，通用性比较弱；系统开销小，效率高；操作系统具有一定的兼容性和扩展性；应用软件较专业，用户界面不够友好；系统主要用来控制系统负载以及监控应用程序运行。

第三阶段是以嵌入式操作系统为标志的嵌入式系统。这一阶段系统的主要特点是：嵌入式操作系统能运行于各种不同类型的微处理器上，兼容性好；操作系统内核精小、效率高，并且具有高度的模块化和扩展性；具备文件和目录管理、设备支持、多任务、网络支持、图形窗口以及用户界面等功能；具有大量的应用程序接口(API)，开发应用程序简单；嵌入式应用软件丰富。

第四阶段是以基于 Internet 为标志的嵌入式系统，这是一个正在迅速发展的阶段。目前大多数嵌入式系统还孤立于 Internet 之外，但随着 Internet 的发展以及 Internet 技术与信息家电、工业控制技术等结合日益密切，嵌入式设备与 Internet 的结合将代表着嵌入式技术的真正未来。

1.3.2　嵌入式操作系统的特点

1. 嵌入式系统开发人员对操作系统的依赖性

早期的硬件设备很简单，软件的编程和调试工具也很原始，与硬件系统配套的软件都必须从头编写。程序大都采用汇编语言，调试是一件很麻烦的事。随着系统越来越复杂，操作系统就显得很必要。

- 操作系统能有效管理越来越复杂的系统资源。
- 操作系统能够把硬件虚拟化，使得开发人员从繁忙的驱动程序移植和维护中解脱出来。
- 操作系统能够提供库函数、驱动程序、工具集以及应用程序。

在 20 世纪 70 年代的后期，出现了嵌入式系统的操作系统。20 世纪 80 年代末，市场上出现了几个著名的商业嵌入式操作系统，包括 Vxworks、Neculeus、QNX 和 Windows CE 等，这些系统提供性能良好的开发环境，提高了应用系统的开发效率。

2. 嵌入式操作系统的特点

与其他类型的操作系统相比，嵌入式操作系统具有以下一些特点。

- 体积小。嵌入式系统有别于一般的计算机处理系统，它不具备像硬盘那样大容量的存储介质，而大多使用闪存(Flash Memory)作为存储介质。这就要求嵌入式操作系统只能运行在有限的内存中，不能使用虚拟内存，中断的使用也受到限制。因此，嵌入式操作系统必须结构紧凑，体积微小。
- 实时性。大多数嵌入式系统都是实时系统，而且多是强实时多任务系统，要求相应的嵌入式操作系统也必须是实时操作系统(RTOS)。实时操作系统作为操作系统的一个重要分支已成为研究的一个热点，主要探讨实时多任务调度算法和可调度性、死锁解除等问题。
- 特殊的开发调试环境。提供完整的集成开发环境是每一个嵌入式系统开发人员所期待的。一个完整的嵌入式系统的集成开发环境一般需要提供的工具是编译/

连接器、内核调试/跟踪器和集成图形界面开发平台。其中的集成图形界面开发平台包括编辑器、调试器、软件仿真器和监视器等。

1.3.3 嵌入式操作系统的发展状况

嵌入式操作系统已经从简单走向成熟，主要有 VxWorks、Palm OS、QNX、Linux、μC/OS、Windows CE 等。

1. VxWorks

VxWorks 操作系统是美国 WindRiver 公司于 1983 年设计开发的一种嵌入式实时操作系统(RTOS)，具有良好的持续发展能力、高性能的内核以及友好的用户开发环境，在嵌入式实时操作系统领域牢牢占据着一席之地。

VxWorks 所具有的显著特点是：

- 可靠性、实时性和可裁剪性；
- 支持多种处理器，如 x86、i960、Sun Sparc、Motorola MC68xxx、MIPS、POWER PC 等；
- 大多数的 VxWorks API 是专有的，如用于火星机器人。

2. Palm OS

Palm OS 是著名的网络设备制造商 3COM 旗下的 Palm Computing 掌上电脑公司的产品。Palm OS 在 PDA 市场上占有很大的市场份额，将近 90%，最近有所下降，目前主要与 Windows CE 进行激烈竞争。

3. QNX

QNX 加拿大 QNX 公司的产品。QNX 是在 x86 体系上开发出来的，这和别的实时系统(RTOS)不一样，别的很多 RTOS 都是从 68K 的 CPU 上开发成熟，然后再移植到 x86 体系上来的。

QNX 是一个实时的、可扩充的操作系统，它部分遵循 POSIX 相关标准，由于 QNX 具有强大的图形界面功能，因此很适合作为机顶盒、手持设备(手掌电脑、手机)、GPS 设备的实时操作系统使用。

4. Linux

嵌入式系统越来越追求数字化、网络化和智能化，因此原来在某些设备或领域中占主导地位的软件系统越来越难以为继，整个系统必须是开放的、提供标准的 API，并且能够方便地与众多第三方的软硬件沟通。

Linux 是开放源码的，不存在黑箱技术，遍布全球的众多 Linux 爱好者又是 Linux 开发有的强大技术后盾。

Linux 的内核小、功能强大、运行稳定、系统健壮、效率高，易于定制裁剪，在价格上极具竞争力。

Linux 不仅支持 x86 CPU，可以支持其他数十种 CPU 芯片。嵌入式 Linux(Embedded Linux)是指对 Linux 经过小型化裁剪后，能够固化在容量只有几百千字节或几兆字节的存储器芯片或单片机中，应用于特定嵌入式场合的专用 Linux 操作系统。嵌入式 Linux

的开发和研究是目前操作系统领域的一个热点。主要有 RTLinux 和 μCLinux。

5. μC/OS

μC/OS 即微控制器操作系统，由美国人 Jean Labrosse 于 1992 年完成并发布。它在世界各地都获得了广泛的应用，如照相机、医疗器械、音响设备、发动机控制、高速公路电话系统、自动提款机等。它是一种专门为嵌入式设备设计的内核，目前已经被移植到 40 多种不同结构的 CPU 上，运行在从 8 位到 64 位的各种系统之上。

1998 年发布了 μC/OS-Ⅱ。尤其值得一提的是 2000 年，它得到美国航空管理局(FAA)的认证，可以用于飞行器中对安全要求极为苛刻的系统之上。鉴于 μC/OS 可以免费获得代码，对于嵌入式 RTOS 而言，选择 μC/OS 无疑是经济的选择。

1.4　Windows CE 操作系统

1. Windows CE 历史简介

Windows CE 发展历史如图 1-2 所示。Windows CE 是专为掌上电脑等消费类电子产品设计的操作系统，Windows CE 1.0 在 1996 年首次面世。首批为 Windows CE 设计的是手持式 PC，配有 480×240 或 640×240 的屏幕和小键盘。在 1997 年的 Fall Comdex 97 大会上，发布了 Windows CE 2.0。Windows CE 2.0 较 Windows CE 1.0 有显著更新，主要是网络支持，和它配套的是类似的更新一些的硬件出现，具有 640×240 的横向屏幕，一些是彩色的，还具有略大一些的键盘。

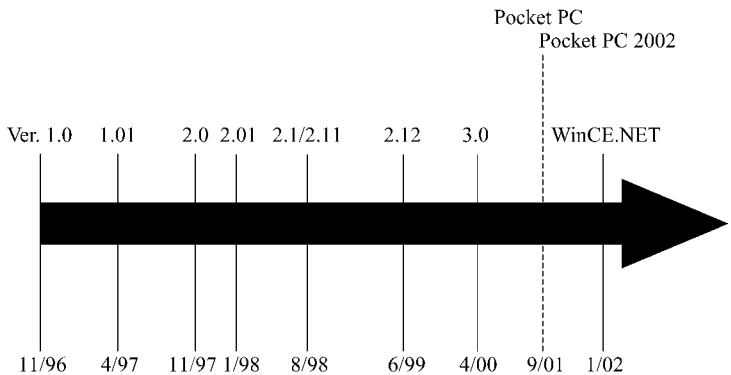

图 1-2　Windows CE 发展历史

1998 年 1 月，消费电子展览会上，微软宣布了两个平台：掌上电脑(Palm-size PC 可以看作 Pocket PC 的前身，其中使用 Windows CE 2.x 的称为 Palm-size PC，使用 Windows CE 3.x 的称为 Pocket PC)和车载 PC(Auto PC 是 Windows CE 的一种应用模式，多用在一些车载电脑、工业自动控制等场合，可按客户需要修改输入输出方式而不限定使用原有的程序)。

2000 年 4 月，微软发布了 Windows CE 3.0，宣布了 Pocket PC，它对老式掌上电脑(Palm-size PC)做了极大的增强。最初的 Pocket PC 使用预发布的具有更多功能的

Windows CE 3.0。Pocket PC的用户界面也有所不同,具有更加整洁的外观和修改过的主页。然而,Pocket PC最重要的特性是极大地改进了Windows CE的性能。微软做了很多工作来调优Windows CE的性能,这些改进加上更快的CPU,可以让系统运行得更快。

2001年晚些时候,Pocket PC更新为Pocket PC 2002。这次发布使用了Windows CE 3.0的最终发布版本,并做了一些用户接口方面的改进。同时也增加了Pocket PC Phone版本,在Pocket PC设备中集成了蜂窝电话支持功能。这些设备具有Pocket PC的功能,也具有蜂窝电话的联通功能,形成了新一代的几乎可以始终连接的移动软件。

微软的另一个开发小组发布了Smart Display(一种具有触摸屏的无线监视器),可以通过802.11b无线网络连接到个人计算机,并使用Windows XP Professional操作系统的Remote Desktop(远程桌面)来访问主机。它使用Windows CE .NET 4.1系统,具有平板式设备形式,有无线网络访问能力,有一个底座可以连接到PC上。当使用底座的时候,它可以作为第二个显示器;当不用底座的时候,它可作为PC的移动显示器。

2003年春季,Pocket PC团队发布了升级版的Pocket PC,称为Pocket PC 2003。系统在用户接口方面没有很多变化,由于是基于Windows CE .NET 4.2,所以在稳定性和性能方面都有了巨大改进。Pocket PC 2003还集成了蓝牙功能,OEM厂商可以选择是否包含该功能。

微软还和OEM厂商合作生产基于Windows CE的蜂窝电话,这些电话中的少部分被称为Smartphone(智能电话)。Smartphone在2002年末发布,最初是基于Windows CE 3.0,2003年升级到了Windows CE 4.2,并增加了一系列特征,包括.NET runtime功能。

2. 嵌入式操作系统Windows CE .NET介绍

Windows CE .NET是Windows CE 3.0的后继产品。Windows CE .NET为嵌入式市场重新设计,为快速建立下一代智能移动和小内存占用的设备提供了一个健壮的实时操作系统。Windows CE .NET具备完整的操作系统特性集和端对端开发环境,它包括了创建一个基于Windows CE的定制设备所需的一切,如强大的联网能力、强劲的实时性和小内存体积占用以及丰富的多媒体和Web浏览功能。

3. Windows CE .NET的新增特性

嵌入系统的开发人员会在Windows CE .NET中发现大量的新增特性和改进特性,其中包括:无线技术,如蓝牙(Bluetooth);设备仿真,该特性使用户可以对完整的设备环境进行仿真而无需任何额外的硬件投资;平台向导,使用户可以从众多的预置设备设计中进行选择,以便跳跃式地开始开发流程;此外,还有丰富的多媒体和Web浏览功能,如Microsoft Internet Explorer 6.0和Windows Media编解码器(Codec)和控件。这个端对端的工具组能在最新的硬件设备上快速建立应用程序。

(1) 无线设备

蓝牙是一种新兴的无线通信技术,它允许设备在大约10m的范围之内互相进行通信。这种技术的主要目标是使设备无需物理电缆即可通信。使用这种技术的一些主要设

备包括无线耳机、调制解调器和打印机。Windows CE .NET 对蓝牙技术具有与生俱来的支持，它允许设备使用具有蓝牙功能的移动电话（例如数据调制解调器）交换文件和对象，以及使用具有蓝牙功能的局域网（LAN）访问点提供网络连接。

（2）设备仿真

Windows CE .NET 具有仿真技术，该技术允许开发人员在 Windows 2000 或 Windows XP Professional 工作站上开发和测试设计，而无需额外的硬件投资。

（3）新的平台向导

新的平台向导使用户可以迅速而容易地基于所构建的设备类型创建一个新的平台，并且为设备的设计提供一个基础：移动电话/智能电话、数字成像设备、工业自动化设备、Internet 媒体设备、PDA/移动手持设备、住宅门禁、POS 设备、机顶盒、微内核、Web 板设备、Windows 瘦客户机、主板支持包（BSP）等。

Windows CE .NET 包括了多种主板支持包，从而缩短了让操作系统正常工作在硬件上所需的时间。Windows CE .NET 当前支持多种标准开发主板（SDB）。

（4）广泛的 CPU 选择

Windows CE .NET 支持四种微处理器家族以及仿真技术。

- ARM：支持的处理器包括 ARM720T、ARM920T、ARM1020T、StrongARM、XScale。
- MIPS：支持的处理器包括 MIPS Ⅱ/32 with FP、MIPS Ⅱ/32 without FP、MIPS16、MIPS Ⅳ/64 with FP、MIPS Ⅳ/64 without FP。
- SH-x：支持的处理器包括 SH-3、SH-3 DSP、SH-4。
- X86：支持的处理器包括 486、586、Geode、Pentium 系列。

消费电子产品领域核心处理器种类很多，因为不同的消费电子产品针对不同的市场，对于核心处理器的要求也很多样，所以 Windows CE 必须能够支持不同的核心处理器。

Windows CE 被设计成跨核心处理器的系统，但是由于每种处理器的指令集不同，所以肯定会需要做针对不同核心处理器的移植。在 Windows CE 中这部分与核心处理器相关的代码被分成三个部分。第一部分是内核（Kernel）部分，这部分代码由微软提供。第二部分是 OEM Adaptation Layer（简称 OAL）（OEM 是 Original Equipment Manufacture 的简称），这部分代码由设备制造商提供。第三部分是 Boot Loader，这部分也是由设备制造商提供。Boot Loader 用来初始化核心处理器和关键芯片完成硬件初始化。OAL 是描述具体硬件平台的那部分代码，负责抽象和管理硬件资源。

还有一种第三方的商家，也被叫做 Windows CE 系统集成商，他们会提供所有硬件平台相关的 OAL 并绑定相应处理器的 Windows CE 的内核。一般这些会被提供给某个特定的硬件平台，叫做 Board Support Package（BSP），这类软件包还包括了相应的驱动程序。BSP 可以从 Windows CE 系统集成商那里购买，也可以由 OEM 自己设计。

（5）.NET Compact Framework

.NET Compact Framework 是.NET Framework 的一个子集，专门面向内存占用较少的设备而设计。.NET Compact Framework 是一个面向安全、可下载应用程序的独立于硬件的程序执行环境，定位于资源有限的计算设备，并且专门为这些设备进行了优化。

.NET Compact Framework 提供了多种语言可供选择(最初是 Microsoft Visual Basic .NET 和 Microsoft Visual C# .NET),并且消除了语言互操作性所面临的一些常见问题。例如,Visual C#和 Visual Basic 组件可以轻松混合到一个解决方案之中,从而使更多的开发人员可以更轻易地参与到一个解决方案之中。.NET Compact Framework 所支持的每种语言都可以平等地访问底层的框架和操作系统特性。.NET Compact Framework 还包括了对 Web 服务的支持,这些服务使开发人员可以在小型、短时连接设备上对网络传输的粒度进行更细致的控制,允许后台的数据预取,并且使应用程序可以汇集来自不同的服务器的数据。.NET Compact Framework 支持所有被 Windows CE .NET 支持的处理器。

(6) 丰富的多媒体和最新的 Web 浏览技术

支持最新的多媒体体验,包括 Microsoft DirectX API 和 Windows Media 8 编解码器和控件。Microsoft DirectX API 通过允许对 3D 视频显示硬件以设备无关的方式进行设备有关的访问,对 3D 互动图形程序提供支持。Windows Media 8.0 编解码器可以对最新的高保真、低带宽编码多媒体流提供支持。

使用 Internet Explorer 浏览器支持 Internet Explorer 自定义的和个性化的用户界面方面的增强,支持 Passport、HTML 4.0、DHTML、SSL、MSXML 3.0、JScript 5.5、ActiveX 控件、Java 小程序和 Java 虚拟机。

(7) Windows CE .NET 行业应用

工业自动化制造商过去通常部署来自不同供应商的孤立信息技术系统,现在他们开始考虑那些能在整个企业范围内为通用信息访问提供高效、经济基础结构的技术。产品设计和生产、销售和生产计划、产品管理和工艺控制,以及生产和后勤的无缝连接,是在当今这个注重时间、基于 Web 的崭新生产制造环境下取得竞争优势的关键所在。Windows CE 操作系统是一个适合下一代互连工业自动化设备的理想小体积嵌入平台。由于使用了 MSMQ(Microsoft Message Queuing)这样的先进应用服务,Windows CE 使实现与工厂生产现场现有 IT 设施的全面集成成为可能。它还具有极大增强了的实时支持以提供时间关键的嵌入应用程序所需要的边界限制、确定性的响应时间和控制。因为 Windows CE 能从闪存磁盘中启动,也就避免了暴露在灰尘、高温和震动环境下,从而使它可以适应甚至是最恶劣的生产环境。

1.5 Windows CE 程序开发工具

1. 操作系统开发(定制)工具 Platform Builder

Platform Builder 用来开发、调试、配置操作系统映像,为构建一个完整的可运行的 Windows CE 提供大量的要素,包括应用程序、DLL、设备驱动、字体和图标等。此外,它还包含了.NET 精简框架运行时库(可选组件),这样定制 Windows CE 智能移动设备可以包含.NET 精简框架运行时库。

2. 应用程序开发工具

应用程序开发涉及托管代码的概念。托管代码是为面向 .NET 的公共语言运行库

的服务编写的代码。为了面向这些服务，该代码必须向运行库提供最低级别的信息(元数据)。默认情况下，所有 Visual C# .NET、Visual Basic .NET 和 JScript .NET 代码都受托管。默认情况下，Visual C++ .NET 代码不受托管，但编译器可以通过指定命令行开关(/CLR)来产生托管代码。

应用程序开发可以选择 Embedded Visual C++ 4.0 SP3 开发“本地”应用程序(非托管代码)，可以使用 C、C++、MFC(微软基础类)、ATL(活动模板库)。

应用程序开发也可以选择 Visual Studio .NET 2003 中的 Visual Basic .NET，Visual C# .NET 开发托管代码程序要使用.NET Compact Framework(.NET 精简框架)。

1.6　习题

1. 嵌入式系统的概念是什么？
2. 嵌入式系统的特点是什么？
3. 嵌入式处理器分为哪四类？
4. 常见的嵌入式操作系统有哪些？
5. Windows CE.NET 有哪些新特点？
6. Windows CE 的开发工具有哪些？

第 2 章

C# 程序设计基础

本书所有例程都是用 C# 语言来实现的。本章主要介绍 C# 数据类型、操作符、表达式、程序流程控制语句、方法、数组，以及面向对象编程。其中面向对象的编程技术是本章的重点，主要介绍：

- 类和对象
- 继承
- 多态

本章是学习本书后续内容的必要准备。

2.1 认识 C#

2.1.1 简单的 Windows 应用程序

在开始设计简单的 Windows 应用程序之前，先要安装 Visual Studio .NET 2003，安装步骤请参考相关资料。

(1) 启动 Visual Studio .NET 2003。

(2) 选择【文件】|【新建】|【项目】命令，打开【新建项目】对话框。

(3) 在【项目类型】列表框中选择【Visual C# 项目】，然后在【模板】列表框中选择【空项目】。

(4) 在【名称】文本框中，输入 Hello 作为该项目的名称；在【位置】下拉列表框中，输入要将项目保存到的目录，或单击【浏览】按钮以选择目录。

(5) 单击【确定】按钮，Visual Studio 将创建一个新项目 Hello，并显示解决方案资源管理器。

(6) 在解决方案资源管理器中，右击解决方案下的 Hello 项目，在弹出的快捷菜单中选择【添加】|【添加新项】命令。

(7) 在【类别】列表框中选择【本地项目】，然后在【模板】列表框中选择【代码文件】，在【名称】文本框中输入 Test。

(8) 单击【打开】按钮，然后在空的 Test 代码文件中输入如下代码。

```
using System;
Class Test
```

```
{
    static void Main()
    {
        System.Console.WriteLine("hello,world!");
    }
}
```

(9) 按 F5 键运行该应用程序，可得到 hello,world! 的输出。

2.1.2　代码分析

1. 命名空间的引用

在 C# 程序中要使用类库中的类，必须添加包含要使用的类库的程序集的引用。类库中的每个类都从属于特定的命名空间。在 C# 中使用 using 语句定位所使用类库中的类的命名空间，以便程序直接找到该类，例如：

```
using System;
```

2. 定义类

每一个 C# 程序包括至少一个自定义类。这些类称为程序员自定义类或用户自定义类。在 C# 中用关键字 class 引导一个类的定义，其后接着类的名称(本例中是 Test)。关键字 class 是 C# 的保留字。class Test 后的“{”表示开始一个类的定义，对应的“}”用来结束类的定义(如果花括号不成对出现，会出现编译错误)。例如：

```
class Test
{
⋮
}
```

3. Main 方法

Main 方法是程序的入口，程序控制在该方法中开始和结束。该方法必须声明为 static。void 表示该方法没有返回值。在 Main 方法中，语句 System.Console.WriteLine(“hello,world!”) 在命令行输出窗口显示字符串“hello,world!”。WriteLine 是一个方法，一般通过在类名称后加上点操作符“.”和方法名称来调用。

2.2　变量与数据类型

2.2.1　变量与常量

1. 变量的含义

变量在程序的运行过程中值可以发生变化。变量名是一个标识符。标识符是用来标识变量名称、常量名称、方法名称、数组名称、类名称、属性名称等的有效字符序列。简单地说，标识符就是一个名称。但标识符必须符合以下规则：

- 必须以字母或下划线“_”开头；

- 必须只包含字母字符、十进制数字字符、连接字符、组合字符、格式设置字符或下划线。

另外，有一些关键字对于C#编译器而言有特定的含义，例如前面出现的using、namespace关键字。如果错误地使用其中一个关键字，编译器会产生一个错误，但马上就会知道错了，所以不必担心。

下面显示了一些有效和无效的变量名。

```
aB123_45        //有效
_567            //有效
_               //无效
12ABC           //无效
Xyz$wv          //无效
```

标识符是大小写敏感的(即区分大小写)，因此两个标识符即使是只有大小写不同，也会被视为不同的标识符。例如，在同一程序中，rose、Rose和ROSE是3个不同的变量。

变量关系到数据的存储。实际上，可以把计算机内存中的变量看作架子上的盒子。在这些盒子中，可以放入一些东西，再把它们取出来，或者只是看看盒子里是否有东西。变量也是这样，数据可放在变量中，也可以从变量中取出或查看。

尽管计算机中的所有数据都是相同的东西(一组0和1)，但变量有不同的内涵，称为类型。下面再使用盒子来类比。盒子有不同的形状和尺寸，某些东西只能放在特定的盒子中。建立这个类型系统的原因是不同类型的数据需要用不同的方法来处理。变量限定为不同的类型，可以避免混淆它们。例如，组成图片文件的0和1序列与组成声音文件的0和1序列，其处理方式是不同的。

2. 变量的声明

在C#中，使用变量之前必须首先声明它，即给变量指定一个名称和一种类型。声明了变量后，就可以把它们用作存储单元，存储声明的数据类型的数据。

声明变量的一般格式为：

```
数据类型    变量名;
```

例如：

```
int v;
int t;
string title1="输出结果1";
string title2="输出结果2";
```

如果使用未声明的变量，代码就不会被编译，但此时编译器会告诉发生了什么问题，所以这不是一个致命错误。另外，使用未赋值的变量也会产生一个错误，编译器会检测出这个错误。

3. 常量的含义

代码中经常包含反复出现的常数值。有时，代码中也可能必须使用某些难于记忆或没有明显意义的数值。

在这些情况下，可通过使用常量来提高代码的可读性，并使代码更易于维护。常量是有意义的名称，用于代替不变的数字或字符串。顾名思义，常量存储那些在应用程序的整个执行过程中都保持不变的值。

常量必须具有一个有效的符号名称，其命名规则与变量命名规则相同。

4. 常量的声明

应为常量编写包括一个访问说明符、一个 const 关键字、类型和一个表达式的声明，如下面的这些例子所示。

```
Public const int DaysInYear=365;
Private const int WorkDays=250;
Const double conPi=3.14159265358979;
Const double conPi2=conPi * 2;
```

可以在一行中声明多个常量，不过，如果每一行只声明一个常量，代码会更具可读性。如果在一行中声明多个常量，它们必须具有相同的访问级别。要在一行中声明多个常量，可用一个逗号和一个空格分隔声明，如下所示。

```
Const int Four=4,Five=5,Six=6;
```

2.2.2　简单数据类型

C#的数据类型分为以下几种。

- 简单类型
- 结构类型
- 枚举类型

表 2-1 显示了内置 C#类型的关键字，这些类型是 System 命名空间中预定义类型的别名。

表 2-1　C#类型关键字

C#类型	.NET Framework 类型	C#类型	.NET Framework 类型
bool	System.Boolean	int	System.Int32
byte	System.Byte	uint	System.UInt32
sbyte	System.SByte	long	System.Int64
char	System.Char	ulong	System.UInt64
decimal	System.Decimal	object	System.Object
double	System.Double	short	System.Int16
float	System.Single	ushort	System.UInt16

表 2-1 中，除了 object 外，所有类型均称为简单类型。

C#类型的关键字及其别名可以互换。例如，可使用下列两种声明中的一种来声明一个整数变量。

```
int x = 123;
System.Int32 x = 123;
```

若要显示任何 C# 类型的实际类型，使用系统方法 GetType()。例如，下列语句显示了表示 myVariable 类型的系统别名。

```
Console.WriteLine(myVariable.GetType());
```

还可使用 typeof 运算符。

简单数据类型主要包括整数类型、实数类型、布尔类型、字符类型等。

1. 整数类型

C# 支持 8 种整数：sbyte、byte、short、ushort、int、uint、long 和 ulong。

sbyte：8 位有符号整数。

byte：8 位无符号整数。

short：16 位有符号整数。

ushort：16 位无符号整数。

int：32 位有符号整数。

uint：32 位无符号整数。

long：64 位有符号整数。

ulong：64 位无符号整数。

表 2-2 列出了整型数的表示范围。如果整数表示的值超出了 ulong 的范围，将产生编译错误。

表 2-2 整型类型表示数的范围

类 型	范 围	大 小
sbyte	−128～127	有符号 8 位整数
byte	0～255	无符号 8 位整数
char	U+0000～U+ffff	16 位 Unicode 字符
short	−32 768～32 767	有符号 16 位整数
ushort	0～65 535	无符号 16 位整数
int	−2 147 483 648～2 147 483 647	有符号 32 位整数
uint	0～4 294 967 295	无符号 32 位整数
long	−9 223 372 036 854 775 808～9 223 372 036 854 775 807	有符号 64 位整数
ulong	0～18 446 744 073 709 551 615	无符号 64 位整数

2. 实数类型

实数类型是同时使用整数部分和小数部分来表示数字的类型。实数类型包括 float、double、decimal。

float：7 位有效数字。

double：15～16 位有效数字。

decimal：28～29 位有效数字。

实型数的表示范围见表 2-3。

表 2-3　实型类型表示数的范围

类　型	大 致 范 围	精　度
float	$\pm 1.5\times10^{-45}\sim\pm 3.4\times10^{38}$	7 位
double	$\pm 5.0\times10^{-324}\sim\pm 1$	15～16 位
decimal	$\pm 1.0\times10^{-28}\sim\pm 7.9\times10^{28}$	28～29 位

如果希望实数为非整型数据，使用文本类型字符。文本类型字符用 M 或 m 表示 decimal，用 F 或 f 表示 float，用 D 或 d 表示 double。decimal 关键字表示 128 位数据类型。同浮点型相比，decimal 类型具有更高的精度和更小的范围，这使它适合于财务和货币计算。使用后缀 m 或 M，例如：

```
decimal myMoney = 300.5m;
```

如果没有后缀 m，该数字将被视为 double，从而导致编译错误。整型被隐式转换为 decimal，其计算结果为 decimal。因此，可以用整数初始化十进制变量而不使用后缀，例如：

```
decimal myMoney = 300;
```

在浮点型和 decimal 类型之间不存在隐式转换，因此，必须使用显式转换在这两种类型之间进行转换。例如：

```
decimal myMoney = 99.9m;
double x = (double) myMoney;
myMoney = (decimal) x;
```

例如：

```
//decimal 类型转换
using System;
public class TestDecimal {
    public static void Main ()
    {
        decimal d = 9.1m;
        int y = 3;
        Console.WriteLine(d + y);  // 结果转换为 decimal 类型
    }
}
```

在此例中，同一个表达式中兼用 decimal 和 int 类型，计算结果为 decimal 类型。

3. 字符类型

char 关键字用于声明表 2-4 所示范围内的 Unicode 字符。Unicode 字符是 16 位字

符,用于表示世界上多数已知的书面语言。

表 2-4 字符类型的范围

类型	范 围	大 小	.NET Framework 类型
char	U+0000～U+ffff	16 位 Unicode 字符	System.Char

char 类型的常数可以写成字符、十六进制换码序列或 Unicode 表示形式,也可以显式转换整数字符代码。以下所有语句均声明了一个 char 变量并用字符 X 将其初始化。

```
char MyChar = 'X';              // 字符
char MyChar = '\x0058';         // 十六进制
char MyChar = (char)88;         // 整型数转换成字符
char MyChar = '\u0058';         // Unicode 码
```

char 类型可隐式转换为 ushort、int、uint、long、ulong、float、double 或 decimal 类型。但是,不存在从其他类型到 char 类型的隐式转换。

下面的示例阐释了 char 中的某些方法。

```
using System;

public class CharStructureSample {
    public static void Main() {
        char chA = 'A';
        char ch1 = '1';
        string str = "test string";

        Console.WriteLine(chA.CompareTo('B'));              //输出:'-1'
        Console.WriteLine(chA.Equals('A'));                 //输出:'True'
        Console.WriteLine(Char.GetNumericValue(ch1));       //输出:'1'
        Console.WriteLine(Char.IsControl('\t'));            //输出:'True'
        Console.WriteLine(Char.IsDigit(ch1));               //输出:'True'
        Console.WriteLine(Char.IsLetter(','));              //输出:'False'
        Console.WriteLine(Char.IsLower('u'));               //输出:'True'
        Console.WriteLine(Char.IsNumber(ch1));              //输出:'True'
        Console.WriteLine(Char.IsPunctuation('.'));         //输出:'True'
        Console.WriteLine(Char.IsSeparator(str, 4));        //输出:'True'
        Console.WriteLine(Char.IsSymbol('+'));              //输出:'True'
        Console.WriteLine(Char.IsWhiteSpace(str, 4));       //输出:'True'
        Console.WriteLine(Char.Parse("S"));                 //输出:'S'
        Console.WriteLine(Char.ToLower('M'));               //输出:'m'
        Console.WriteLine('x'.ToString());                  //输出:'x'
    }
}
```

和 C 和 C++ 一样,C# 存在转义字符,用来在程序中指定特殊的字符(见表 2-5)。

表 2-5　转义字符

转义字符	说　明
一般字符	除.、$、^、{、[、(、|、)、*、+、?、\外，其他字符与自身匹配
\a	与响铃(警报)\u0007 匹配
\b	如果在[]字符类中，则与退格符\u0008 匹配
\t	与 Tab 符\u0009 匹配
\r	与回车符\u000D 匹配
\v	与垂直 Tab 符\u000B 匹配
\f	与换页符\u000C 匹配
\n	与换行符\u000A 匹配
\e	与 Esc 符\u001B 匹配
\040	将 ASCII 字符匹配为八进制数(最多三位)；如果没有前导零的数字只有一位数或者与捕获组号相对应，则该数字为后向引用(有关详细信息，参见反向引用)。例如，字符\040 表示空格
\x20	使用十六进制表示形式(恰好两位)与 ASCII 字符匹配
\cC	与 ASCII 控制字符匹配。例如，\cC 为 Ctrl-C
\u0020	使用十六进制表示形式(恰好四位)与 Unicode 字符匹配
\	在后面带有不识别为转义符的字符时，与该字符匹配。例如，* 与\x2A 相同

4. 布尔类型

布尔类型数据是用来表示“真”和“假”这两个概念的。这虽然看起来非常简单，实际上应用非常广泛。计算机是用二进制来表示数据的，即不管何种数据，在计算机中都以二进制来处理和存取。布尔型数据表示的逻辑变量只有两种取值：“真”和“假”，在 C# 中分别取值 true 和 false。

注意，在 C 和 C++ 中，用 0 来表示假，其他任何非 0 表示真。在 C# 中 true 值不能被任何其他非 0 值取代。在整数类型和布尔类型之间不存在转换，将整数类型转换成布尔类型是不合法的。

```
bool x=1     //错误，不存在这种写法，只能写成 x=true 或 x=false
```

2.2.3　结构类型

利用上面的简单类型可以进行一些常用的数字运算、文字处理，但是实际中经常会碰到一些更为复杂的数据类型。比如，通讯录中的记录可以包含他人的姓名、电话和地址。如果按照简单类型来管理，每一条记录都要放到 3 个不同的变量中，这样工作量很大，也不直观。有没有更好的办法呢？

正如上面的例子，在实际生活中经常把一组相关的信息放在一起，把一些相关的变量组成一个单一的实体，这个单一实体的类型就是结构类型。结构体中的每一个变量称为结构

体成员，结构类型的变量采用 struct 来进行声明。例如，可以如下定义通讯录记录结构：

```
struct PhoneBook{
   public string name;
   public string phone;
   public string address;
}
PhoneBook Book1;
```

Book1 就是一个 PhoneBook 结构类型的变量。上面的 public 表示对结构类型成员的访问权限。对结构成员的访问可以通过结构变量名加上访问符“.”，再跟成员名称。例如：

```
Book1.name="Mike";
```

结构类型所包含的成员的数据类型可以相同也可以不同。例如可以在通讯录的记录中加上年龄这个成员。

```
struct PhoneBook{
   public string name;
   public int age;
   public string phone;
   public string address;
}
```

还可以把结构类型当作另一个结构成员的类型。例如：

```
struct PhoneBook{
   public string name;
   public int age;
   public string phone;
   public string address{
        public string street;
        public string city;
        public string code;
   }
}
```

C#中，在定义结构成员的同时，可以定义结构的方法。下面的示例显示了一个正确定义的结构。

```
public struct Int32: IComparable, IFormattable
{
    public const int MinValue = -2147483648;
    public const int MaxValue = 2147483647;

    public static string ToString(int i)
    {
        // 方法的代码
    }

    public string ToString(string format, IFormatProvider formatProvider)
    {
```

```
        //方法的代码
    }

    public override string ToString()
    {
        //方法的代码
    }

    public static int Parse(string s)
    {
        //方法的代码
        return 0;
    }

    public override int GetHashCode()
    {
        //方法的代码
        return 0;
    }

    public override bool Equals(object obj)
    {
        //方法的代码
        return false;
    }

    public int CompareTo(object obj)
    {
        //方法的代码
        return 0;
    }

}
```

不要为 struct 提供默认的构造函数。注意，C# 不允许 struct 具有默认的构造函数。运行库会插入一个构造函数，将所有的值初始化为零状态。这样，不用在每个实例上运行构造函数就可以创建结构数组。不要使 struct 依赖于为每个实例调用的构造函数。结构实例可以在不运行构造函数的情况下使用零值创建。还应该为所有实例数据设置为零、假或空（如果适用）等有效的状态设计 struct。

2.2.4　枚举类型

枚举（enum）实际上是用一组逻辑上密不可分的整数提供一个便于记忆的符号。例如，声明代表一个星期的枚举类型的变量：

```
enum WeekDay{
    Sunday,Monday,Tuesday,Wednesday,Thursday,Friday,Saturday
};
WeekDay Day;
```

注意，结构类型是由不同的数据类型组成的新的数据类型，其变量的值是由各个成员