第四章 Windows CE.NET 体系结构

4.1 Windows CE.NET 特性

4.1.1 什么是Windows CE .NET?

- 适于嵌入式产品的、小映像尺寸的、 32位实时，多任务，抢占式嵌入式操作系统操作系统。
- Win32 API子集
- 高度组件化和可配置
- 对标准硬件和特定硬件都可定制

OS + Tools = Windows CE.NET

高级、实时　　创建下一代智能、连接、小映像尺寸设备

4.1.2 Windows CE.NET设计目标

1. 适应小型系统
2. 支持多种处理器和计算机结构，并支持多种设备接口
3. 遵循Windows平台的应用开发规范
4. 操作系统各部分模块化，可选择定制
5. 提供网络通信、图形用户界面、数据库、文件系统等支持
6. 支持高要求的实时应用
7. 提供高级电源管理

4.1.3 Windows CE.NET特点

- 灵活的电源管理功能，包括睡眠/唤醒模式；
- 使用了对象存储技术，包括文件系统、注册表及数据库；
- 良好的通信能力
- 256个中断优先级别，支持嵌套
- 更好的线程响应能力
- 出色的图形界面
- 多任务处理能力
- 内置的多媒体处理功能

4.1.3 Windows CE.NET核心特性

1. 内存架构

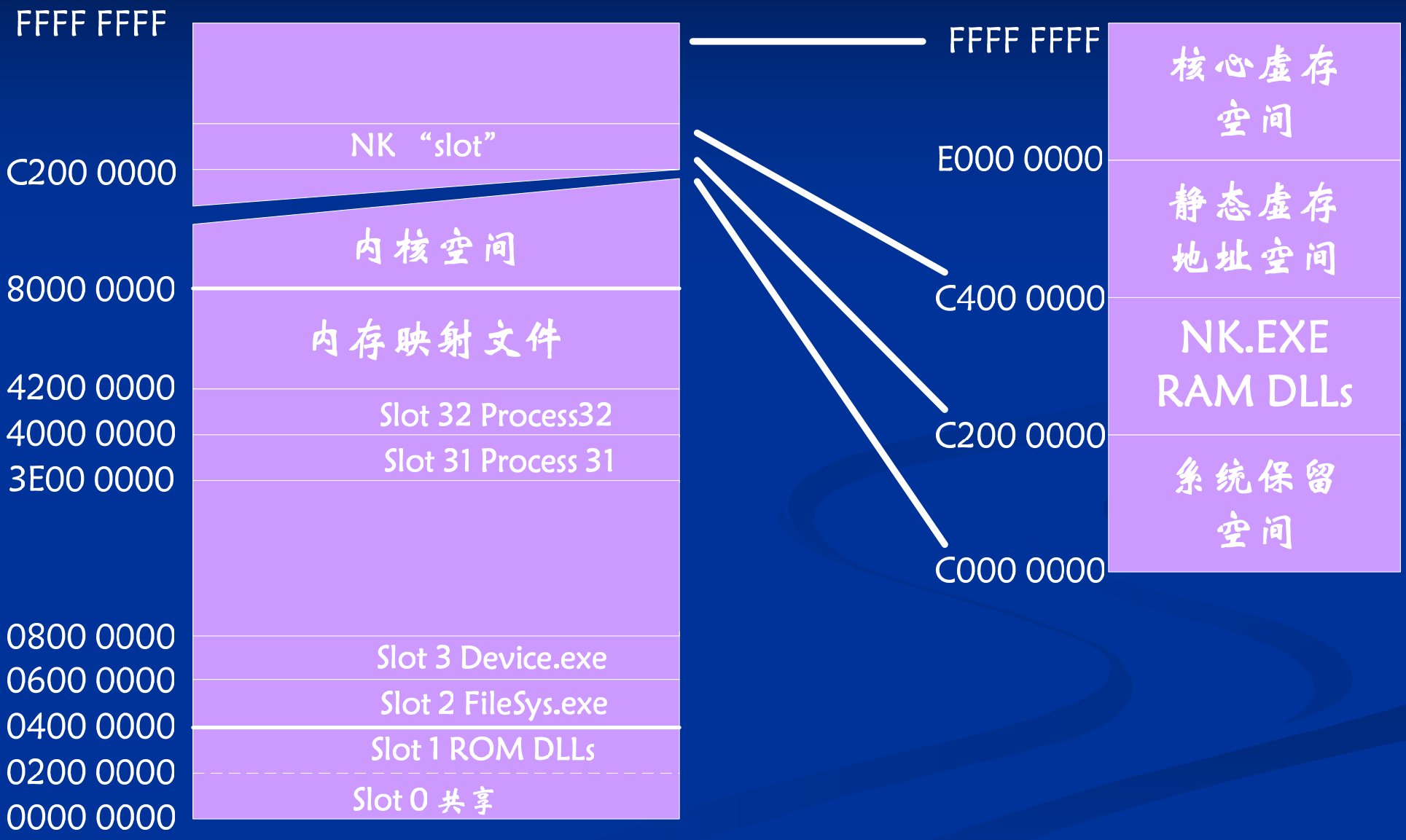
Windows CE.NET提供了灵活的内存访问机制，可以分为三个部分：

- 物理页面管理：主要负责跟踪系统的物理内存的使用情况；
- 虚存管理：主要管理系统的地址映射，页面的换进换出等；
- 堆管理：主要管理进程空间内部的动态内存释放和回收，以支持程序的动态数据结构。

Windows CE.NET只能管理512MB的物理内存和4GB的虚拟地址空间，不同的处理器内存管理方法不同。

- 0X00000000~0X41FFFFFFH：所有应用程序使用，共33个槽。
- 0X42000000~0X7FFFFFFFH：所有应用程序共享区域。
- 0XA0000000~0XBFFFFFFFH：重复定义0X80000000~0X9FFFFFFFH之间的物理空间。
- 0XC0000000~0XC1FFFFFFH：系统保留空间。
- 0XC2000000~0XC3FFFFFFH：nk.exe使用的空间
- 0XA0000000~0XDFFFFFFFH：用户自定义的静态虚拟地址空间，只能用于非缓冲使用。
- 0XE0000000~0XFFFFFFFFH：当内核需要大的虚拟地址空间时分配该范围的空间

内存空间分布


FFFF FFFF
C200 0000
8000 0000
4200 0000
4000 0000
3E00 0000
0800 0000
0600 0000
0400 0000
0200 0000
0000 0000
NK “slot”
内核空间
内存映射文件
Slot 32 Process32
Slot 31 Process 31
Slot 3 Device.exe
Slot 2 FileSys.exe
Slot 1 ROM DLLs
Slot 0 共享
FFFF FFFF
E000 0000
C400 0000
C200 0000
C000 0000
核心虚存空间
静态虚存地址空间
NK.EXE
RAM DLLs
系统保留空间

2. 调度

Windows CE.NET提供了多级别的调度能力，其调度器可视为一个具有增强实时性能的、更多体系结构支持的、结构简化的Windows XP调度器。

	Windows XP	Windows CE.NET
进程	进程结构比较复杂，系统支持进程数目众多。进程空间高达4G	进程结构比较简单，系统最多支持32个独立的进程。进程空间32M
线程	内核基本调度单位	与Windows XP基本相同
轻量级线程	由用户自己实现调度，调度代价小	与线程概念基本相同，主要是为了兼容Windows平台应用
SMP	支持	不支持

	Windows XP	Windows CE.NET
调度算法	32级优先级队列的抢占式调度，同优先级线程采用时间片轮转调度。32级可以分成实时（16-31）和普通（0-15）两个类别。	256级优先队列的抢占式调度，同优先级线程采用时间片轮转调度。256个基本优先级可以分成8个大的级别，持有THREAD_PRIORITY_TIME_CRITIVAL标志的优先级别不能被抢占
调度状态	调度状态众多，转换关系复杂	六种基本状态，转换关系较为简单
配额	时间配额计算比较复杂，处理桌面任务和服务有较大区别，不同优先级具有不同的配额策略	没有时间配额策略，大约采用10ms左右的时间片
多用户	具有多用户能力	不支持多用户
IPC	完善的通信机制和通信保护机制	不支持命名管道、不支持资源句柄的复制和继承等操作，使用共享内存的形式较为普遍。

3. 实时能力

- Windows CE.NET的设计目标可以适应大部分（95%）硬实时系统的需求：
  - 1ms定时周期的误差约为100us;
  - 在200MHz的x86系统下可以期望达到50us。
- Window CE.NET的中断延时和中断处理方式密切相关：
  - 采用在ISR中直接处理时，延时非常短；
  - 采用IST方式处理中断事务的情况，延时较长，调度系统保证在此种情况下的延时不超过100us。

4. 设备驱动程序

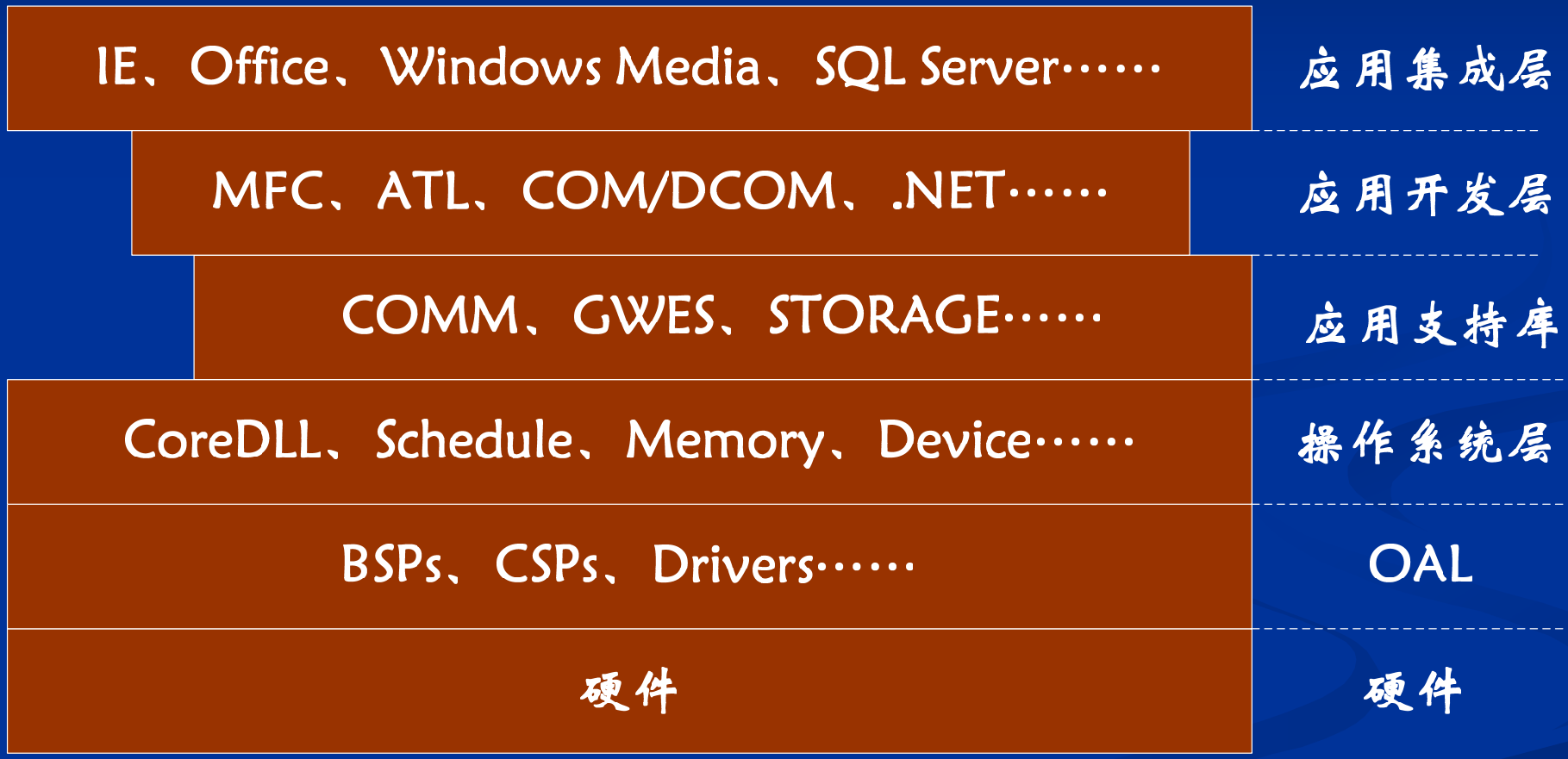
- Window CE.NET集成了大量的设备驱动程序，它们作为系统特性而存在，在Platform Builder 中可以方便地从一个特定的平台配置中加入或者删除。
- Windows CE.NET提供类驱动程序模型以及实现的类驱动程序，这些类驱动模型为应用程序的公共接口提供了可能性。

5. 高级电源管理

高级电源管理是作为设备管理模块的一个部分实现的，在Windows CE.NET设备管理部分中，电源管理器提供了一个符合ACPI标准的电源管理的接口，并使用设备管理的事件传递机制处理相关的电源事件。实际电源管理的实现者是电源管理驱动程序（实现电源管理接口）和每种具体的设备驱动程序中的电源管理例程。

4.1.5 系统模型

1. 分层模型

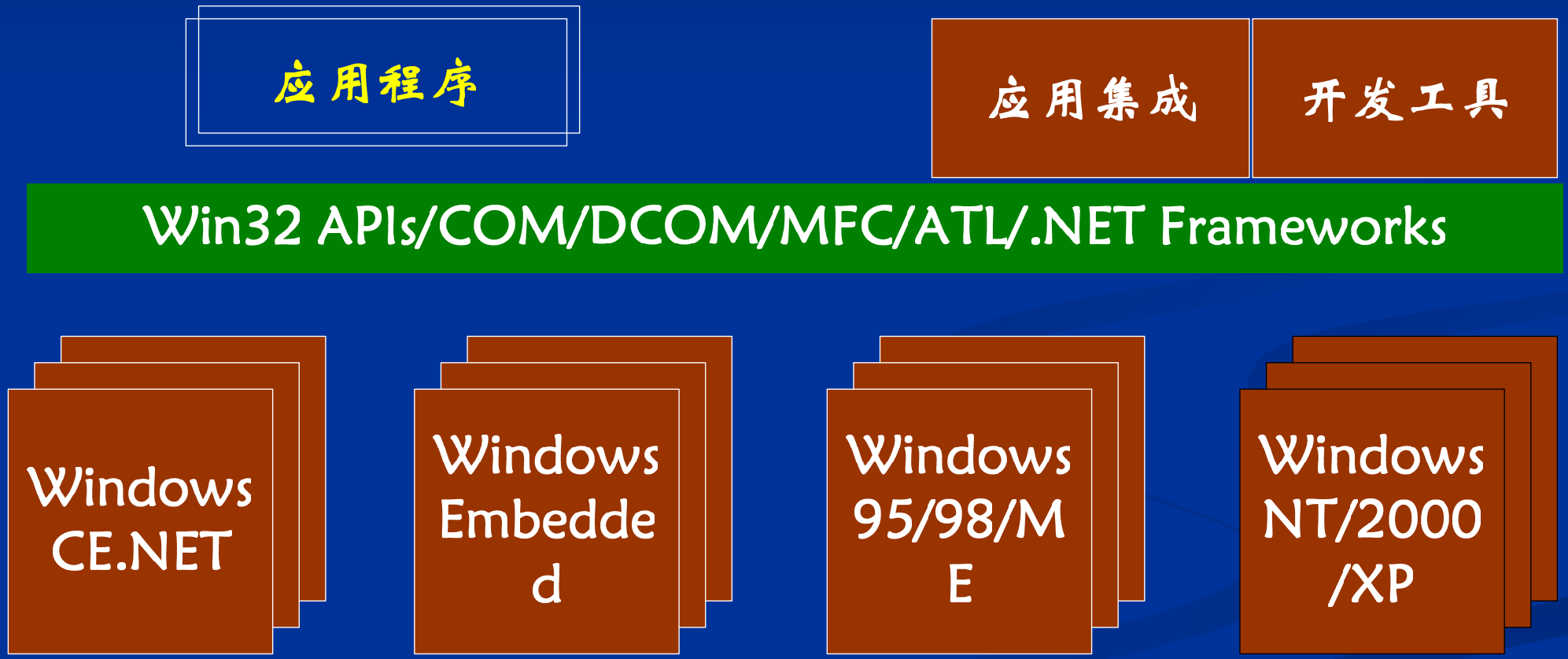


2. 组件模型

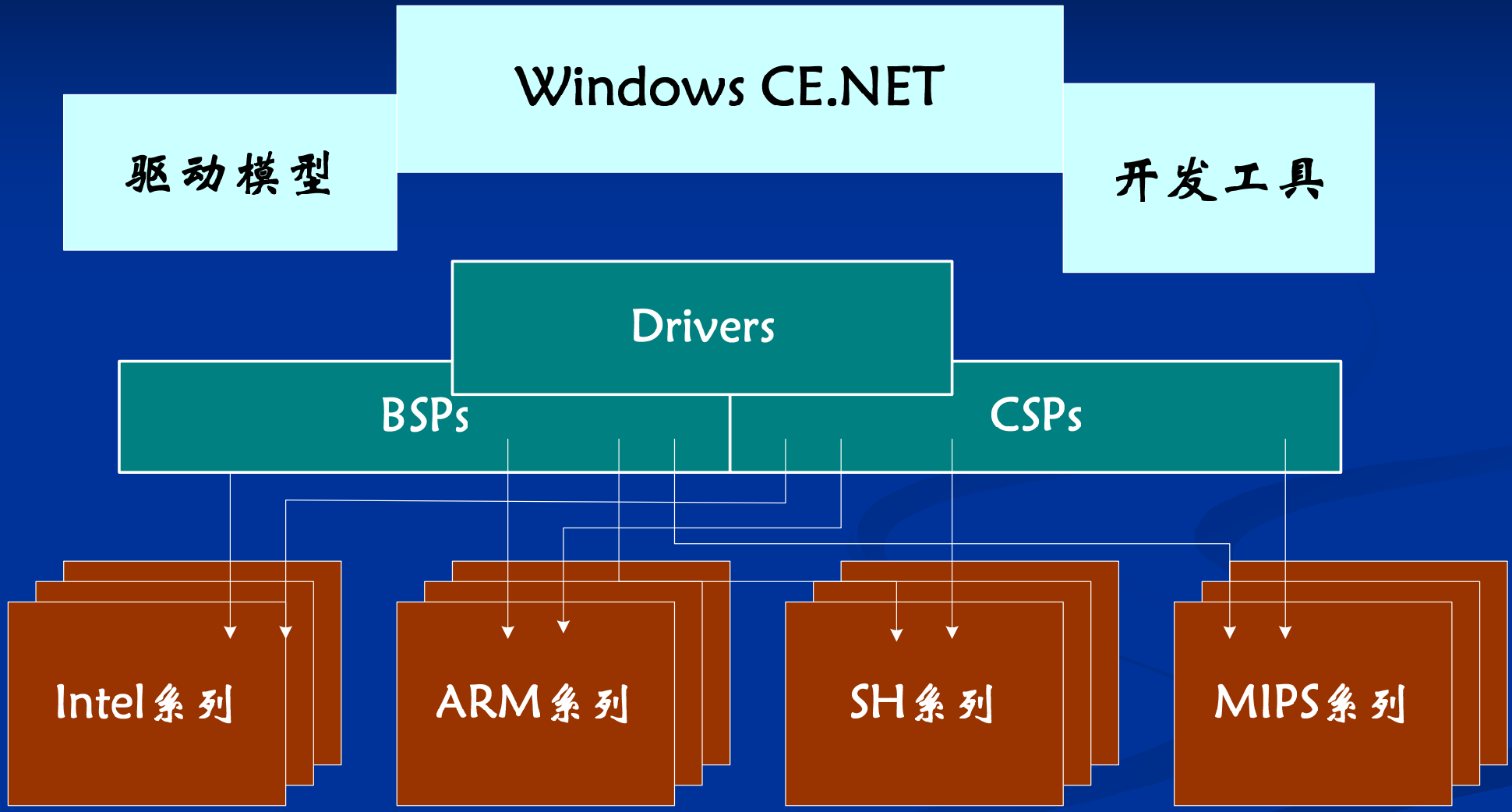

可以定制
基本
不可改动
ISV, OEM
应用程序
应用集成层
嵌入式外壳
远程连接
Windows CE 外壳
应用开发层
WIN32 APIs
COREDLL, WINSOCK, OLE, COMMCTRL, COMMDLG, WININET, TAPI
应用支持库
内核
GWES
设备管理
文件管理
红外
TCP/IP
IPv6
操作系统层
OAL
Bootloader
驱动
设备驱动
文件驱动
OEM适配层
OEM 硬件
硬件层

3. 迁移模型

- 应用迁移模型



• 系统迁移模型


Windows CE.NET
驱动模型
开发工具
Drivers
BSPs
CSPs
Intel系列
ARM系列
SH系列
MIPS系列

4.1.6 注册表类型

Windows CE.NET支持两类注册表：

- 基于对象存储的结构：这类注册表的所有数据使用系统数据对象存储方式存储，比较适合于使用电池、内存不断电的硬件系统，它更加有效，并且更小。
- 基于HIVE的结构：HIVE注册表使用文件存储注册表数据，不存在系统内存断电时备份和恢复的问题，这可以使系统冷启动更快。

4.1.7 代码组织

1.源代码树的构成

安装Platform Builder4.2（选择安装共享源代码）以后，在WINCE420目录下会有如下几个目录：

- PLATFORM：按照不同平台存放的BSP
- PRIVATE：是Windows CE.NET的共享源代码
- PUBLIC：存放Windows平台开发工具，包括大量驱动程序和应用程序开发包
- SDK：按照平台体系结构存放各种开发工具，如编译器等
- OTHERS：包括MFC、ATL的共享代码、库文件以及.NET的共享库

2. 代码共享

Windows CE.NET的共享代码分为两类：

- 一类是PRIVATE目录下的内容，它们的共享受到Microsoft Shared Source License的约束；
- 另一类是其它部分代码，它们的共享条件是Microsoft end-user license agreement。

3. 创建系统

创建一个可以启动的操作系统镜像的方法是使用Platform Builder的创建系统工具。根据需要配置好了一个平台后，可以使用Build菜单项，这样Platform Builder就会调用编译工具把各个模块编译链接起来，并进行内存布局生成启动内存镜像。

4.1.8 Windows CE.NET支持的CPU

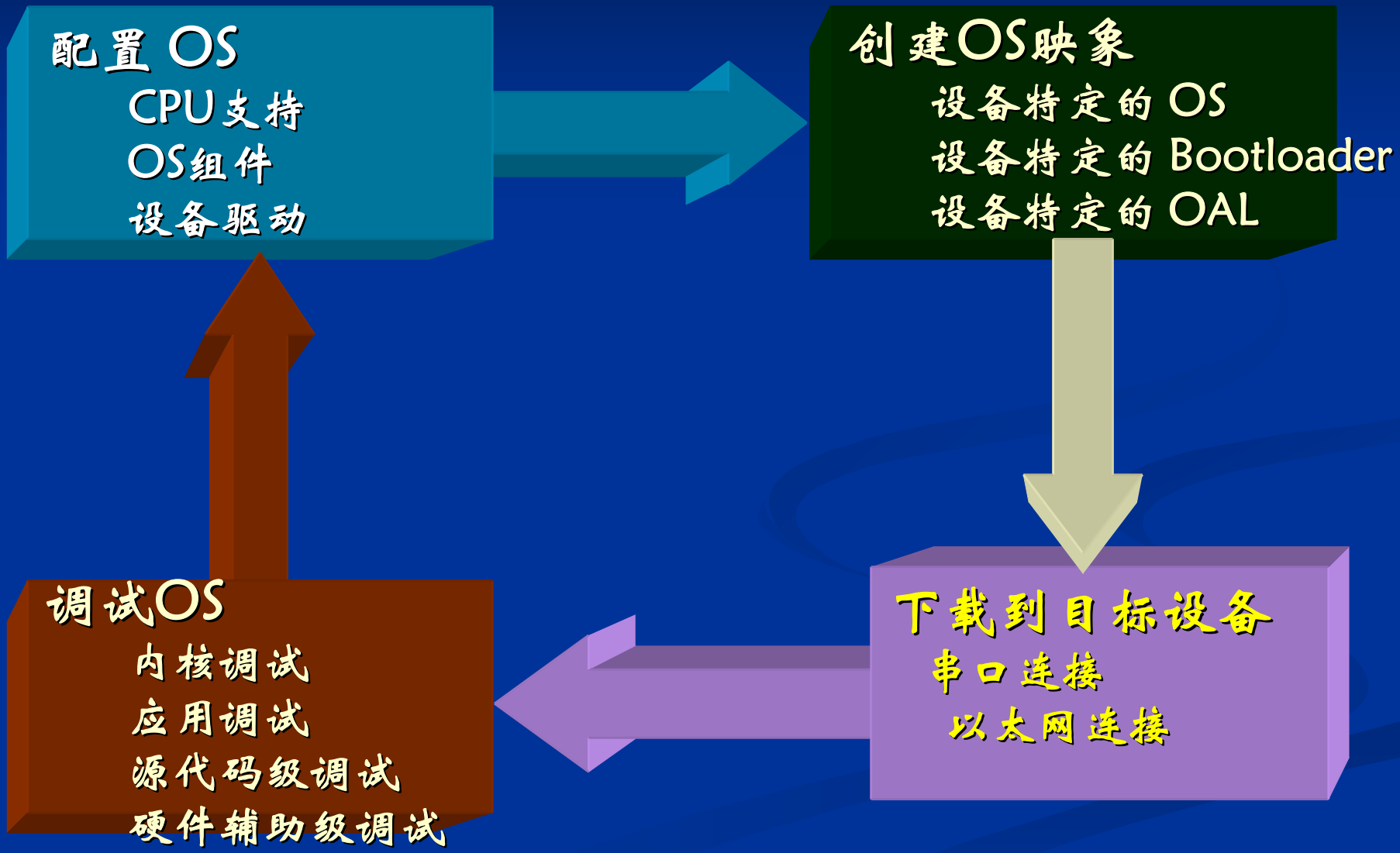
- x86
- Arm/StrongArm/Xscale
- SH3/SH4
- MIPs

Processor Family	CPU	SDB	BSP Name
ARM	Intel SA1110	Intel SA111x Assabet SDB	SA11X0BD
	ARM920	ARM Integrator AP SDB	ARMINTEGRATOR
	ARM1020		
	Intel Xscale	Intel Lubbock SDB	TBD
MIPS	NEC Vr4122	NEC DDB-Vr4122 Eagle SDB	EAGLE
	NEC Vr5432	NEC DDB-Vrc5476 Boston SDB	DDB5476
SHx	SH4-7750	Hitachi SH4 Aspen SDB	ASPEN
	SH3-7729	Hitachi SH3 Keywest SDB	KEYWEST
x86	P5/P4/PIII/PII/ CelK6x/Athlon	CEPC	CEPC
	NS Geode	National Geode Reference Platform	GEODE

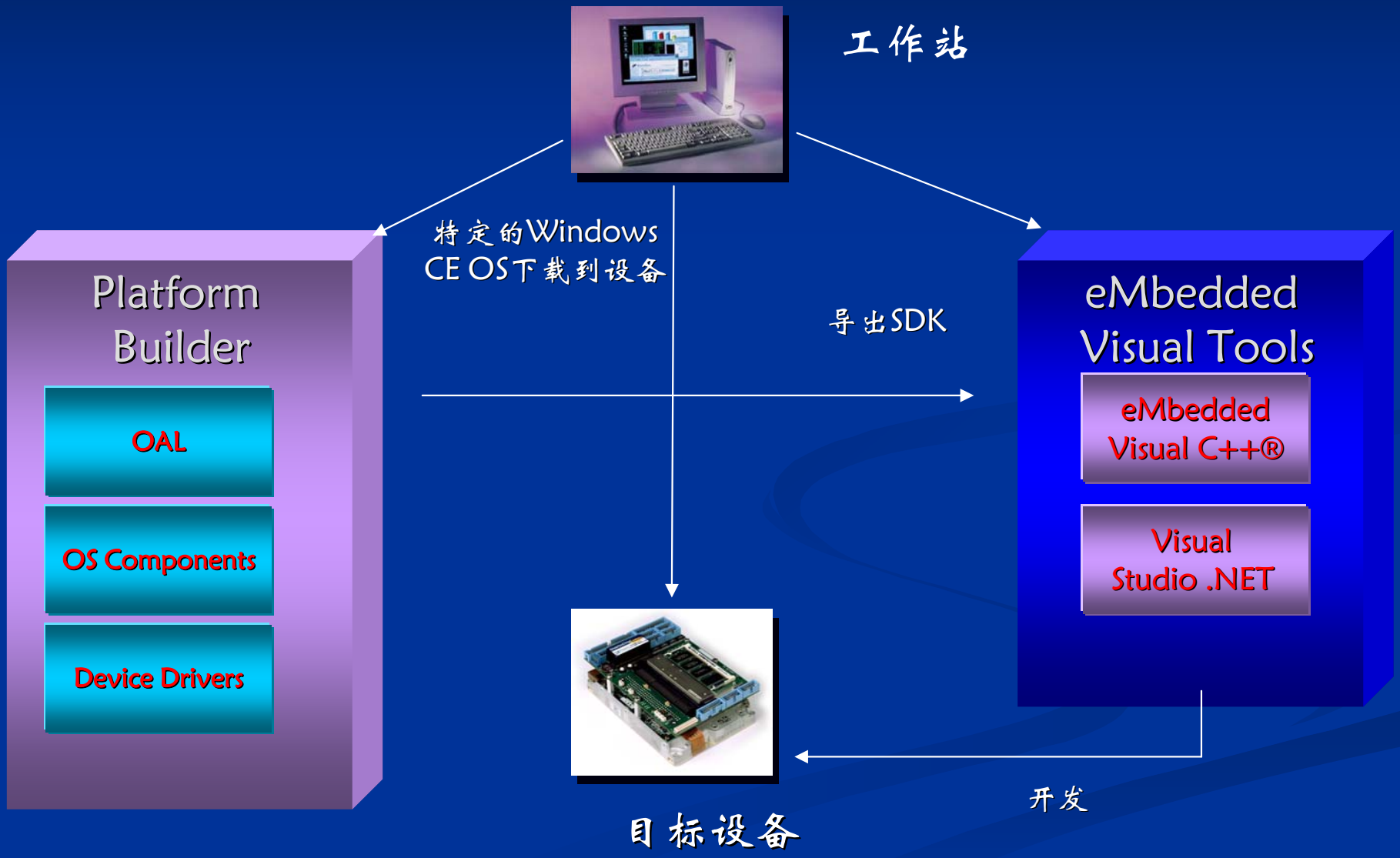
4.1.9 Windows CE .NET开发工具

- 操作系统开发(定制）工具
  - Platform Builder
    - 开发、调试、配置操作系统映像
- 应用程序开发
  - eMbedded Visual C++ 4.0 SP4
    - “本地”应用程序开发
    - C,，C++，MFC（微软基础类），ATL（活动模板库）
  - Visual Studio.NET 2003
    - Visual Basic .NET，C# .NET
    - .NET Compact Framework

Windows CE 开发周期



Windows CE.NET开发过程



4.2 进程、线程描述和调度

4.2.1 进程

进程是一个具有一定独立功能的程序在一个数据集合上的一次动态执行过程。

进程具有以下特征：

- 动态性：具有动态的地址空间，地址空间的大小和内容都是动态变化的；
- 独立性：各个线程的地址空间相互独立；
- 并发性：从宏观上，各个线程是同时独立运行的；
- 结构化：指进程的地址空间的结构划分。

进程与程序的区别与联系：

- 联系：程序是构成进程的两个组成部分之一
- 区别：
  - ✓ 程序是静态的，进程是动态的；
  - ✓ 程序可以在存储介质上长期保存，而进程有它的生命周期，被创建后存在，被撤销后消失；
  - ✓ 一个程序可对应多个进程，而一个进程只能对应一个程序。

1. Windows CE的进程描述

在Windows CE中：

- 最多只能支持32个进程同时运行；
- 系统启动时，至少默认启动四个进程：
  - ✓ NK.exe：提供有关内核的服务
  - ✓ FILESYS.EXE：提供有关文件系统的服务
  - ✓ GWES.EXE：提供对GUI系统的支持
  - ✓ DEVICE.EXE：载入和管理设备驱动程序

2. Windows CE进程结构分析

- 在Windows CE中，每一个进程由process结构来描述，即进程控制块（PCB，Process Control Block）；
- 进程所有的信息都保存在这个结构中，当系统创建一个进程时，将分配一个新的process结构，进程结束时，这个结构将被回收；
- 与 Windows 98或NT 进程比较，Windows CE进程包含较少的状态信息；
- 进程是系统资源分配的基本单位。

process结构的主要部分有：

- Procnum：当前进程ID号，用来辨识一个进程；
- pProxList：存放proxy的队列，LPPROXY结构的链表
- hProc：此进程的句柄，在调用SC_GetProcFromPtr时使用
- dwVMBase：该进程在内存中所占区域的基地址
- pTh：表示当前进程中所有的第一个线程
- BasePtr：指向加载EXE可执行文件的基指针
- lpszProcName：此进程的名称
- PfnEH：进程异常处理例程
- pMainTh：此进程所拥有的主线程
- pmodeResource：包含资源的模块指针
- oe：指向可执行文件句柄的指针

3. 进程的创建和终止

- 创建进程：CreatProcess
- 终止进程：从WinMain过程返回
  - ✓ 可以使用GetExitCodeProcess函数确定进程的退出代码
- 其他进程：
  - ✓ OpenProcess 返回已运行的进程的句柄
  - ✓ GetWindowThreadProcessId 获取窗口的句柄，并返回创建该窗口的进程的进程ID

4.2.2 线程

线程是进程的一个实体，是CPU调度和分配的基本单位，除了一些在运行中必不可少的资源，线程自身不拥有系统资源，但是线程可和同一进程的其它线程共享全部资源。一个线程可以创建和撤销另一个线程，同一进程内的线程可以并发执行。

线程具有许多进程的特征，故又被称为轻量级进程，通常一个进程都有若干个线程，至少有一个（Windows CE中是主线程）。

线程与进程的关系：

- **调度方面：** 进程是分配资源、独立调度和分派的基本单位，引入线程后，线程作为调度和分派的基本单位，进程仍是拥有资源的基本单位。
- **并发性：** 引入线程不仅在进程间可以并发执行，而且在一个进程中的多个线程之间亦可并发执行；
- **拥有的资源：** 进程是一个拥有资源的独立单位，一般线程不拥有系统资源，但可访问其所属的进程的资源；
- **系统开销：** 线程切换的开销远远小于进程切换的开销。

引入线程的优点

- 创建一个新线程花费的时间比创建进程的时间要少得多；
- 两个线程之间的切换花费的时间非常少；
- 在同一个进程内的线程共享进程所拥有的资源，所以线程之间的通信不需要额外的机制，使得通信的速度得以加快。

4.2.3 调度

Windows CE是一个可抢占的、多任务的OS，采用基于优先级的时间片轮转方法；

线程是调度的基本单位，运行一个固定的时间片（一般设置为10ms）；

在Windows CE中有许多队列（最多256个），分别对应不同目录下的进程，一个进程一个时刻只能在一个队列中，而当前运行的进程则例外；

Windows CE基于优先级来选择线程运行，拥有最高优先级的线程将在低优先级的线程前面运行

1. 线程调度的时机

- 线程状态转换；
- 可运行队列的头部插入了一个线程；
- 时间片用完或被抢占；
- 中断处理完成后。

2. 线程优先级

在Windows CE中所有的进程都是同等，单个线程可以有不同的优先级，而且线程所在的进程并不影响线程的优先级。

Windows CE的线程可分为：

- 运行态
- 可运行态
  - ✓ RUNSTATE_RUNNABLE
  - ✓ RUNSTATE_BLOCKED
  - ✓ RUNSTATE_NEEDSRUN
- 阻塞态
  - ✓ WAITSTATE_SIGNLLED
  - ✓ WAITSTATE_PROCESSING
  - ✓ WAITSTATE_BLOCKED

Windows CE中的8种优先级

优先级	描述
THREAD_PRIORITY_TIME_CRITICAL	高于正常优先级3级。具有这个优先级的线程不会被抢占。
THREAD_PRIORITY_HIGHEST	高于正常优先级2级
THREAD_PRIORITY_ABOVE_NOMAL	高于正常优先级1级
THREAD_PRIORITY_NORMAL	正常优先级。所有线程在创建时都是这个优先级。
THREAD_PRIORITY_BELOW_NORMAL	低于正常优先级1级
THREAD_PRIORITY_LOWEST	低于正常优先级2级
THREAD_PRIORITY_ABOVE_IDLE	低于正常优先级3级
THREAD_PRIORITY_IDLE	低于正常优先级4级

3. 与调度相关的函数

- MakeRunIfNeeded（HANDLE hth）函数：在需要的时候调度线程
- MakeRun（PTHREAD pth）函数：判断是否需要重新修改调度策略
- RunqDequeue（PTHREAD pth, DWORD cprio）函数：从运行队列中删除一个线程
- SleepqDequeue （PTHREAD pth）函数：把一个线程从睡眠队列中删除
- ThreadSleep（ DWORD time）函数：让线程睡眠一段时间

4.3 内存管理

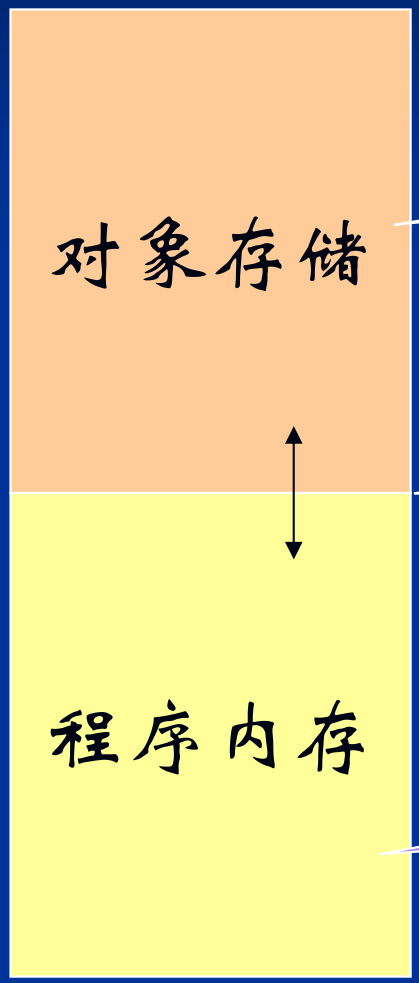
Windows CE.NET只能管理512MB的物理内存和4GB的虚拟地址空间，不同的处理器内存管理方法不同。

- 0X00000000~0X41FFFFFFH：所有应用程序使用，共33个槽。
- 0X42000000~0X7FFFFFFFH：所有应用程序共享区域。
- 0XA0000000~0XBFFFFFFFH：重复定义0X80000000~0X9FFFFFFFH之间的物理空间。
- 0XC0000000~0XC1FFFFFFH：系统保留空间。
- 0XC2000000~0XC3FFFFFFH：nk.exe使用的空间
- 0XA0000000~0XDFFFFFFFH：用户自定义的静态虚拟地址空间，只能用于非缓冲使用。
- 0XE0000000~0XFFFFFFFFH：当内核需要大的虚拟地址空间时分配该范围的空间

4.3.1 物理结构

1. RAM：分为程序空间和对象空间，
   - 程序空间：由提供应用程序运行的堆和栈组成
   - 对象空间：用于存放文件，在系统关闭后仍保留

其边界是可移动的，在进程空间RAM不足的情况下，系统会请求用户提供一些对象空间的RAM来满足运行的要求



相当于永久存储器，即使在系统关闭后其内容仍然保留

- 对象存储和程序内存的边界可以移动
- 用户可使用System Control panel应用程序来移动边界

相当于PC上的RAM，用于程序运行

2. ROM

- 存放绑定了的各种应用的整个系统
- ROM中程序的两种执行方式
  - ✓ 本地执行(XIP)：在ROM中直接运行
  - ✓ 非本地执行：如模块是压缩的，需解压到RAM中执行

4.3.2 逻辑层次结构

1. 物理内存：由一个空闲链表维护，该链表所需的内存空间则用虚拟地址映射到系统预留的区域中的。
2. 虚拟地址：Windows CE采用分页式虚拟内存，页的状态有：空闲、保留和提交3种。
3. 逻辑地址：在虚拟内存之上，用户只需将精力集中在应用程序空间的数据结构所需的内存空间，免去了复杂的页的分配回收操作。

4.3.3 Windows CE的地址空间

Windows CE为每个应用提供2GB的虚拟地址空间，前1GB划分为32个32MB的槽（slot）。0槽由当前应用程序使用（前64KB空间由系统保留），slot 1~31 每个职能包含一个进程，当这个进程为运行态时，即被映射到slot0。

4.3.4 虚拟内存

1. 虚拟内存基础

- 当MMU (Memory Management Unit)可用时，虚拟地址是CPU引用的任何地址
- 除了MMU尚未使能的情况下，CPU通常不直接存取物理地址，
- 虚拟地址必须映射到实际的物理地址来识别物理资源，如ROM、RAM、Flash、CPU寄存器、片上系统原件、总线映射原件等

2. 分页虚拟内存系统

- 微处理器管理的最小内存单元是“页”，当应用程序访问页面时，微处理器把页面的虚拟地址转换成物理资源中的物理地址。
- Windows CE.NET实现了一个分页虚拟内存系统，根据微处理器的不同，分页为1KB或4KB的大小

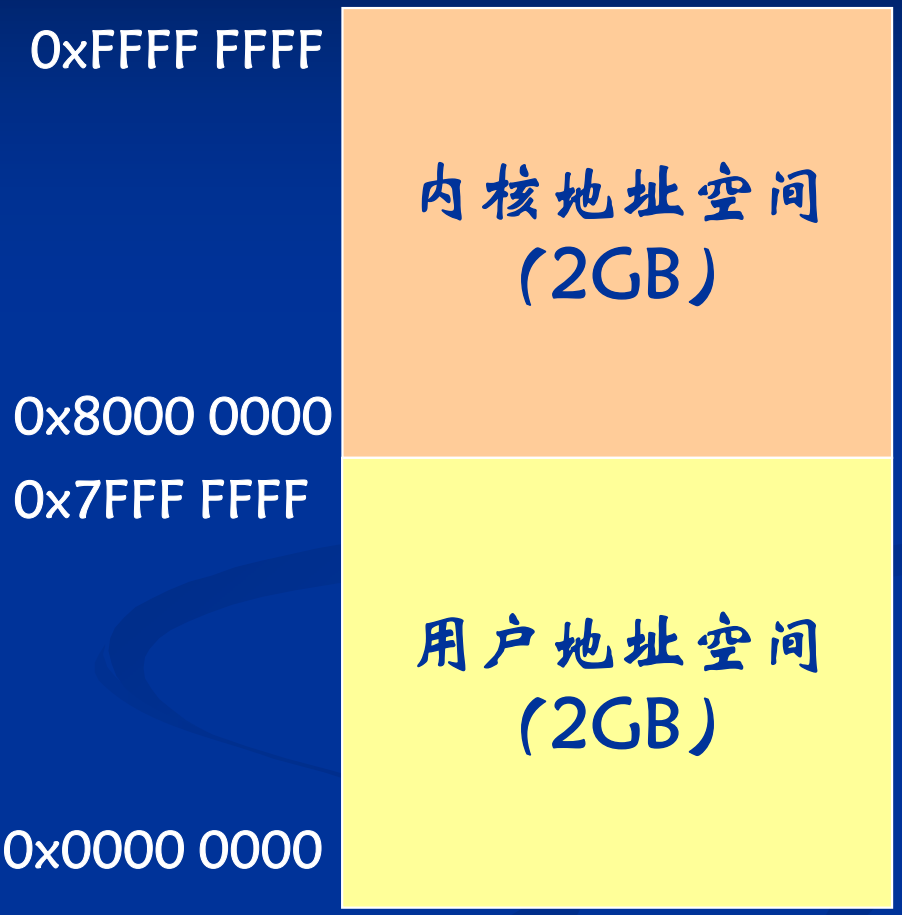
3. 虚拟内存映射到物理内存的方式

虚拟内存映射到物理内存有2种方式：

- 静态映射虚拟地址：虚拟→物理映射从不改变
- 动态映射虚拟地址：虚拟→物理映射可以改变

4. 虚拟地址空间分配

- 虚拟内存管理
  - Windows CE .NET 提供了一个4GB（32位）的虚拟地址空间
  - 大多低2GB的虚拟地址都是动态映射，一些高2GB的虚拟地址是动态映射



4.3.5内存分配方式

1. 虚拟内存分配

VirtualAlloc函数分配内存的过程分为两个步骤：

- 保留虚拟内存空间的区域
  - 不会消耗任何RAM；只是防止一部分虚拟地址空间被用于其他用途
  - 保留的空间将圆整到64KB
- 提交部分或整个区域
  - 指将实际物理内存映射到保留区域

限制因素

- 应用程序的32 MB 虚拟地址空间
- 分配时尽管是以字节为单位，但系统会圆整到下一个页面的边界提交
- 分配时保留的空间将圆整到64KB，而不是在分配函数中指定的大小

2. 使用堆

- 堆是Windows CE.NET为应用程序管理的保留的虚拟内存空间区域
- 应用程序的第一个堆由系统创建，称之为本地堆
- 默认情况下，Windows CE最初保留192KB用于本地堆，但仅当他们被分配时才会提交页面
- 应用程序可以创建一个单独的堆

3. 使用堆栈

- 堆栈是函数使用的变量的存储区域
- 每个线程都有一个堆栈，线程开始时由系统创建
- 线程堆栈的大小默认为64KB，可以通过使用/STACKSIZE链接程序开关来调整
- 创建线程是可以更改其堆栈的大小
- 不要在低内存情况下使用大量堆栈

4. 静态数据

- Windows CE为应用程序的静态数据分配了两个RAM块
  - ✓ 用于读取/写入数据
  - ✓ 用于只读数据
- 以页面为单位来分配
- 好的应用程序应该能确保只有很少或根本没有额外的剩余空间

4.4 存储管理

Windows CE提供了三种类型的文件系统：

- 基于RAM的文件系统：内置文件系统
- 基于ROM的文件系统：内置文件系统
- FAT文件系统：可安装文件系统

Windows CE.NET平台默认使用的存储设备是最多可达256MB的RAM存储器，被称为**对象存储**（object store），Windows CE内置的文件系统均属对象存储。

1. 对象存储

Windows CE中的对象存储与PC机的硬盘相似，为应用程序及其相关数据提供持久稳固的存储。非易失RAM存储器是对象存储的物理基础。

对象存储包含三部分：文件系统、数据库及系统注册表，它们共享一个内存堆。操作系统负责管理内存堆，在必要的时候对文件进行压缩和解压，同时还提供了基于ROM的应用程序与基于RAM的数据间的无缝集成。

对象存储使用基于事务的数据管理机制。

容量限制

- RAM文件系统最大256MB；
- 单个文件最大为32MB；
- 对象存储中的对象个数可达4百万个。

对象标志符（CEOID）

Windows CE为对象存储中的每个对象分分配唯一的对象标志符，用于访问对象存储中的对象。对象的类型可以是：

- 注册表中的一个键
- 注册表中的一个值
- 一个文件
- 文件数据中大小为4KB的部分
- 数据库中的一条记录，最多可保存4KB的数据
- 数据库中的一条记录的扩展信息，也可保存4KB的数据
- 一个数据库
- 数据库的一个卷，仅保证数据库的一个卷是唯一的，多个卷并不能保证，此时要与全球唯一标识（CEGUID）结合使用才能保证。

2. 存储管理器

存储管理器负责管理外围存储设备以及它们所使用的文件系统和块设备驱动程序，存储管理器的功能由fsdmgr.dll实现，包括：

- 文件系统驱动程序管理器
- 分区管理器
- 块设备驱动程序管理器

文件系统驱动程序管理器

文件系统驱动程序管理器负责管理文件系统驱动程序的模块，而只有块设备才会用到文件系统驱动程序，因此文件系统驱动程序管理器主要负责管理：

- 系统中所有可安装文件系统的交互。
- 屏蔽不同文件系统的接口差异，向用户提供标准的函数接口。
- 创建文件句柄，注册卷，安装必要的函数，将应用程序的调用映射到安装的函数上。

文件系统驱动程序的加载

当外围存储设备装入系统中时：

1. 存储管理器从注册表中读取相关配置信息，其中定义了将要使用的文件系统驱动程序（假定该驱动程序名为MyFSD）；
2. 存储管理器从设备管理器接收到一条通知，其中包含该设备的设备名及其GUID等信息；
3. 存储管理器根据配置信息加载合适的分区驱动程序，并为分区装载相应的文件系统；
4. 检查是否有合法的标志：FSD_MountDisk和MyFSD_UnmountDisk两个输出点
5. FSD管理器调用MyFSD.dll的输出函数MyFSD_MountDisk，根据MBR中的DPT以及可能的扩展分区链表中的信息定位设备上的所有卷，并向FSD管理器注册每个卷，加载过程完成（当调用MyFSD_UnmountDisk时，设备上的所有卷会被注销）。

分区管理器与分区驱动程序

- 分区管理器的作用
  - ✓ 调用分区驱动程序完成对分区的管理、装载和卸载
- 分区驱动程序
  - ✓ 对存储设备进行逻辑划分
  - ✓ 可使用不同的文件系统

块设备驱动程序管理程序

- 对设备的读写操作以块为单位，块的大小通常为：512B、1KB、2KB、4KB。
- 块设备驱动程序向上层输出流接口，应用程序通过CreateFile和ReadFile等标准的API对块设备进行访问。
- 当一个块设备驱动程序装入系统中时，它会告知存储设备管理器这一装入事件。

4.5 设备管理

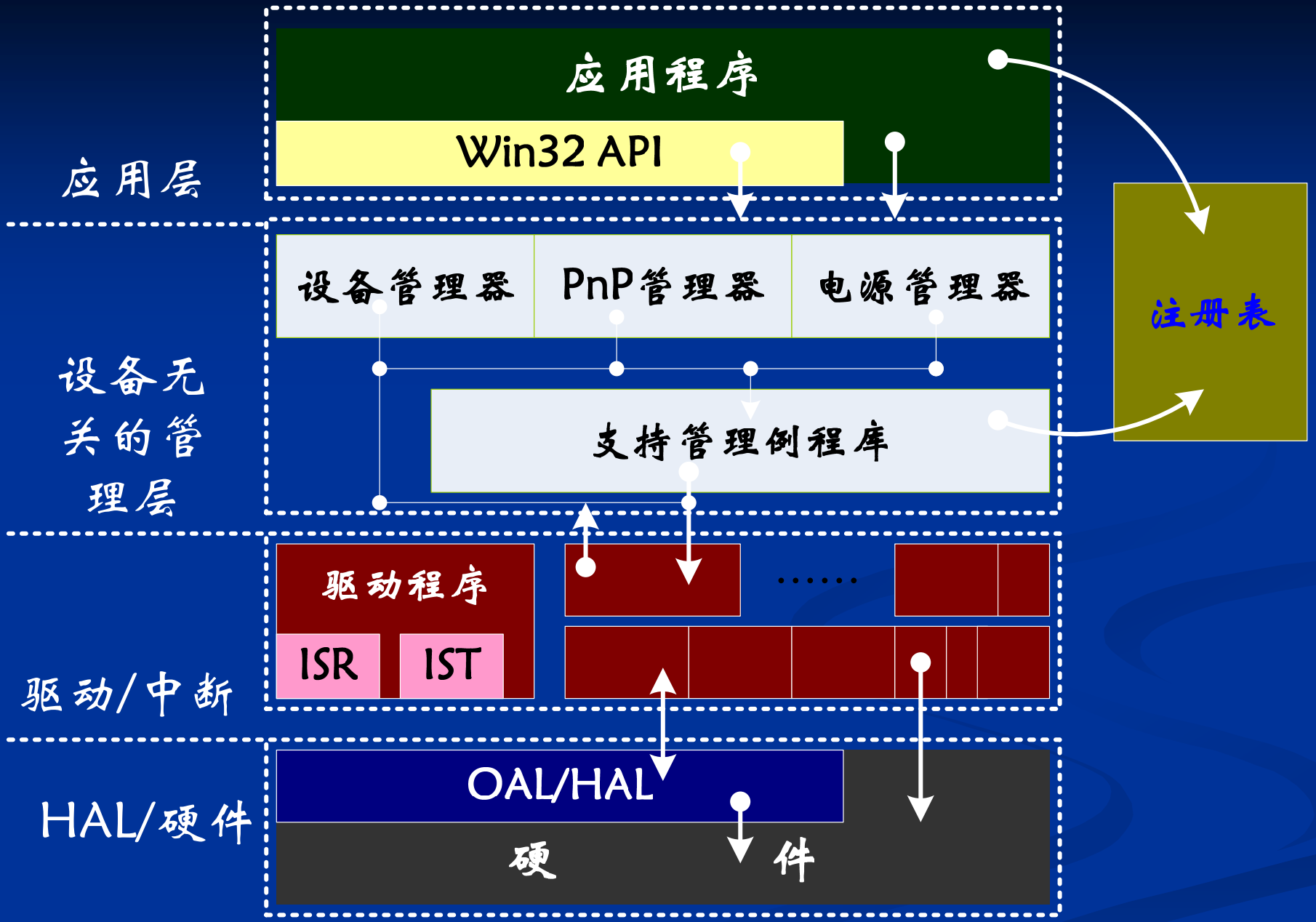
4.5.1 Windows CE.NET的设备管理模式

1.设备管理体系结构

Windows CE与其它常见系统的主要区别在于与设备无关的系统软件这一层，Windows CE.NET将这一层划分为四个独立部分（在这一点上和Windows 2000/XP颇为相似）：

- I/O管理器
- PnP管理器
- 电源管理器
- 管理支持库构成

设备管理的分层模式


应用程序
Win32 API
应用层
设备管理器
PnP管理器
电源管理器
注册表
设备无关的管理层
支持管理例程库
驱动程序
ISR
IST
驱动/中断
OAL/HAL
硬件
HAL/硬件

2. 注册表

- 注册表用于记录系统中若干重要的配置信息。
- 通过共享库RegEnum.dll实现注册表访问
- 具有层次化组织结构，由若干个键和值组成，只具有3个根节点：
  - ✓ HKEY_LOCAL_MACHINE：硬件和驱动配置信息
  - ✓ HKEY_CURRENT_USER：用户配置数据
  - ✓ HKEY_CLASSES_ROOT：OLE和文件类型配置信息
- 注册表的一些限制：
  - ✓ 键或值的名字：255个字符
  - ✓ 数据大小：4KB
  - ✓ 最大键嵌套层数：16层

3. HAL及OAL支持

- HAL和OAL都是为了Windows CE.NET为了实现支持多种体系结构和平台而用来隔离操作系统和硬件的手段。
- HAL提供一般硬件特性的封装，包括特定的硬件机制抽象和驱动程序机制，使得系统和应用程序可以顺利访问硬件特性而不必知道硬件的工作细节。HAL通常还包含HEL（硬件模拟层）。
- OAL主要是操作系统OEM开发者提供操作系统和具体硬件平台的隔离手段，OAL主要实现硬件平台初始化、各种中断服务程序例程、实时时钟、定时器、调试支持、中断开关等。

4. Windows CE.NET设备驱动集成

可支持的设备范围非常广泛，部分如下：

Audio Drivers，Display Drivers，PC Card Drivers，Battery Drivers，DVD-Video Renderer，Printer Drivers，Block Drivers，IEEE 1394 Drivers，Serial Port Drivers，Bluetooth HCI Transport Driver，Keyboard Drivers，Smart Card Drivers，Device Power Management，Network Drivers，Stream Interface Drivers，Direct3D Device Drivers Interface，Notification LED Drivers，Touch Screen Drivers，DirectDraw Display Drivers，Parallel Port Drivers，USB Drivers

4.5.2 设备管理器

设备管理器是Windows CE.NET设备管理的核心机构，它主要负责跟踪、维护系统的设备信息并对设备资源进行调配。

1. 设备信息管理

1. 设备配置信息
   - ✓ 存储于注册表中，通过注册表公用函数访问
   - ✓ 设备管理器的注册表名目
2. 核心数据结构
   - ✓ 使用简单的链表来跟踪设备情况，有4种链表节点：
     - ☞ _fsd_t：跟踪文件系统驱动程序
     - ☞ fsdev_t：跟踪一般设备驱动程序
     - ☞ fsopendev_t：文件流接口访问打开的驱动
     - ☞ fscandidatedev_t：跟踪正在加载的驱动程序
3. 核心设备信息组织
   - ✓ 通过几条链表跟踪设备以及驱动程序的信息，并在注册表里保存设备的配置和状态等
4. 设备管理器的WinMain函数

2. I/O资源管理

① I/O资源管理的任务

- I/O资源管理器跟踪设备驱动程序装载前从注册表信息中初始化所需的系统资源。这些资源包括IRQ的集合与I/O地址空间，这一般都是平台相关的。这样的资源大致可分为两类：
  - ✓ 系统预分配的设备资源
  - ✓ 是由总线驱动程序所请求系统动态分配的IRQ与I/O地址资源。
- I/O资源管理器通过系统I/O初始化来跟踪，并从注册表中找到依据；然后，它将排除任何“固化”的资源，形成可被分配的资源列表，并等待任何资源请求。
- I/O管理器中定义两种注册表键：
  HKEY_LOCAL_MACHINE\Drivers\Resources\IRQ
  HKEY_LOCAL_MACHINE\Drivers\Resources\IO

② 重要数据结构

I/O资源的两种管理模式：位图模式和稀松表模式，相关数据结构在[CEROOT] \Private \Winceos\Coreos \Device \Lib \iomgr.c中实现，具体如下：

- ResourceRange：资源范围
- SparseTableNode：稀疏表节点
- SparseTable：稀疏表表头
- DenseTable：资源位图
- ResourceTable：资源表
- ResourceDomain：资源域

③ 位图模式

对于域空间小于32位的情况，系统使用位图模式（密集表）进行管理。密集表使用位向量表示使用的空间。这样的一个向量通过宏BITRANGE(BIT, LEN)构造。

位图中的1表示连续占用资源，0表示不使用。宏BITRANGE的参数bit表示1起始的位置（从低位到高位），len表示连续1的个数。

④ 稀疏表

就是一个链表，每个节点记录一个范围，分配资源的时候在链上进行节点分割、插入等操作。

3. 设备管理接口

设备管理接口包括两部分：文件流访问接口和设备管理API（实际上就是对设备管理器的功能调用）。

这些接口函数通过WIN32_DEV_CALL和WIN32_FS_CALL等宏调用，系统在winbase.h中使用宏将这些调用封装成为标准的WIN32 API调用名。

4.5.3 电源管理

电源管理器增加整个系统的电源效率，为每个设备提供电源管理，减少设备的电能消耗，同时在重启、运行、挂起状态下在RAM中维护和保护文件系统。它的主要职责是：

- 保证电源管理的具体功能可以被枚举；
- 处理系统的电源服务请求；
- 在系统启动后或空闲过后立刻给设备家电；
- 在系统关闭和进入空闲时使设备掉电或进入睡眠；
- 如果设备支持唤醒功能，唤醒设备。

4.5.4 设备驱动程序

1. 设备驱动程序接口

- 设备接口类：提供了可以被应用程序用来获取设备驱动器特性的方法，一个设备驱动程序可以拥有多个设备接口类，也可以没有；
- IClass注册表子键表示设备接口及其相关GUID
- 访问设备接口的函数

2. 设备驱动程序结构

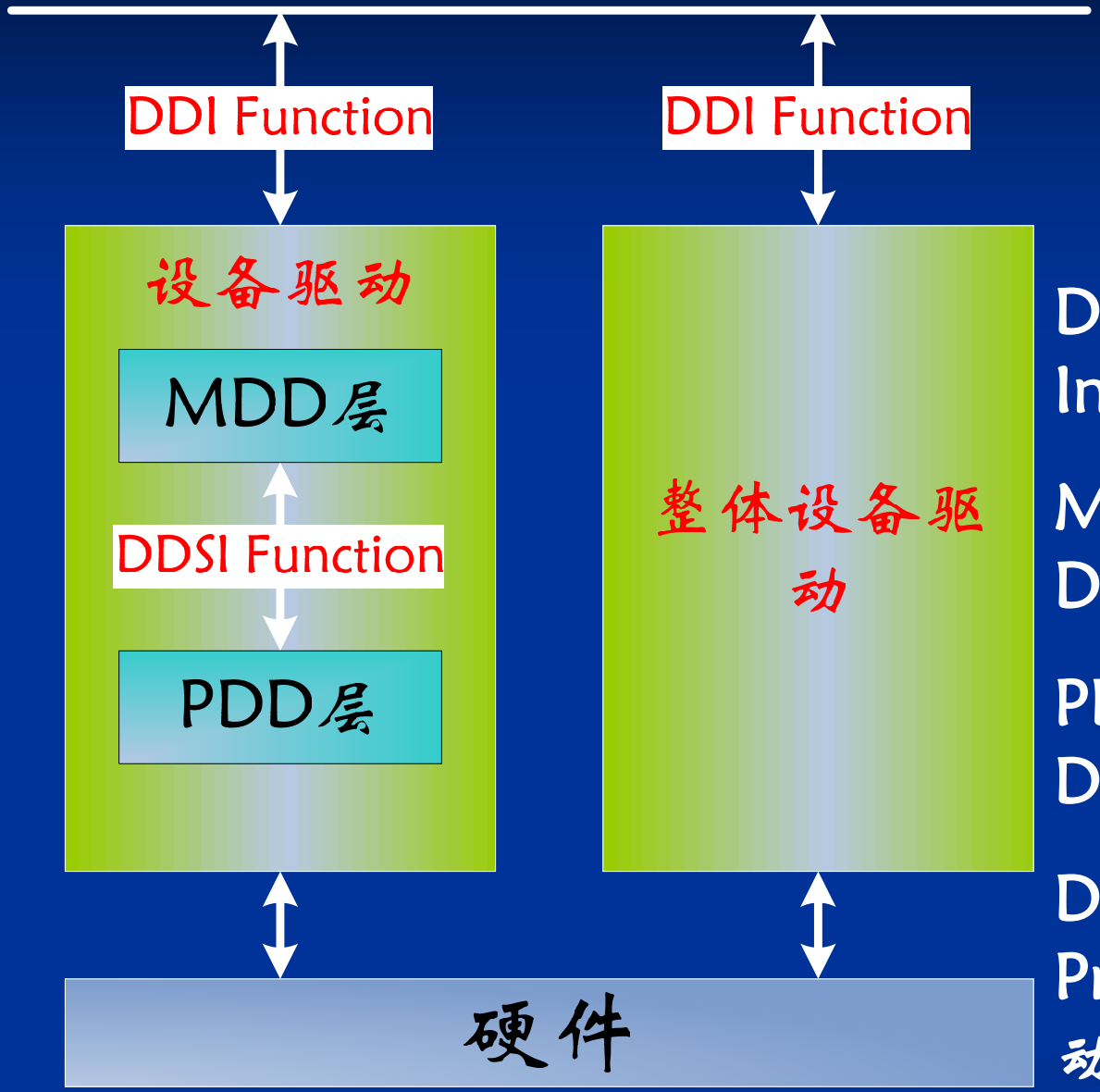
Windows CE的设备驱动模型有2种：

1. 本地设备驱动（NDD：Native Device Driver）

平台内建设备的驱动程序，如键盘、显示设备和触摸屏等是本地设备驱动，它可**根据具体设备的需求提供相应的接口**。

2. 流接口驱动（SID：Stream Interface Driver）

流接口提供一组通用的接口，其向上提供的编程接口使应用程序可以用访问文件的方式访问设备。并非所有内建设备都是本地设备驱动，如串行接口就被实现为流接口驱动，因此访问串行口就如同访问文件一般。



无论流接口驱动还是本地设备驱动都可以分层实现方式和一体实现方式。

DDI（Device Driver Interface）：设备驱动接口

MDD（Model Device Driver）：原形设备驱动

PDD（Platform Dependent Device）：平台相关设备

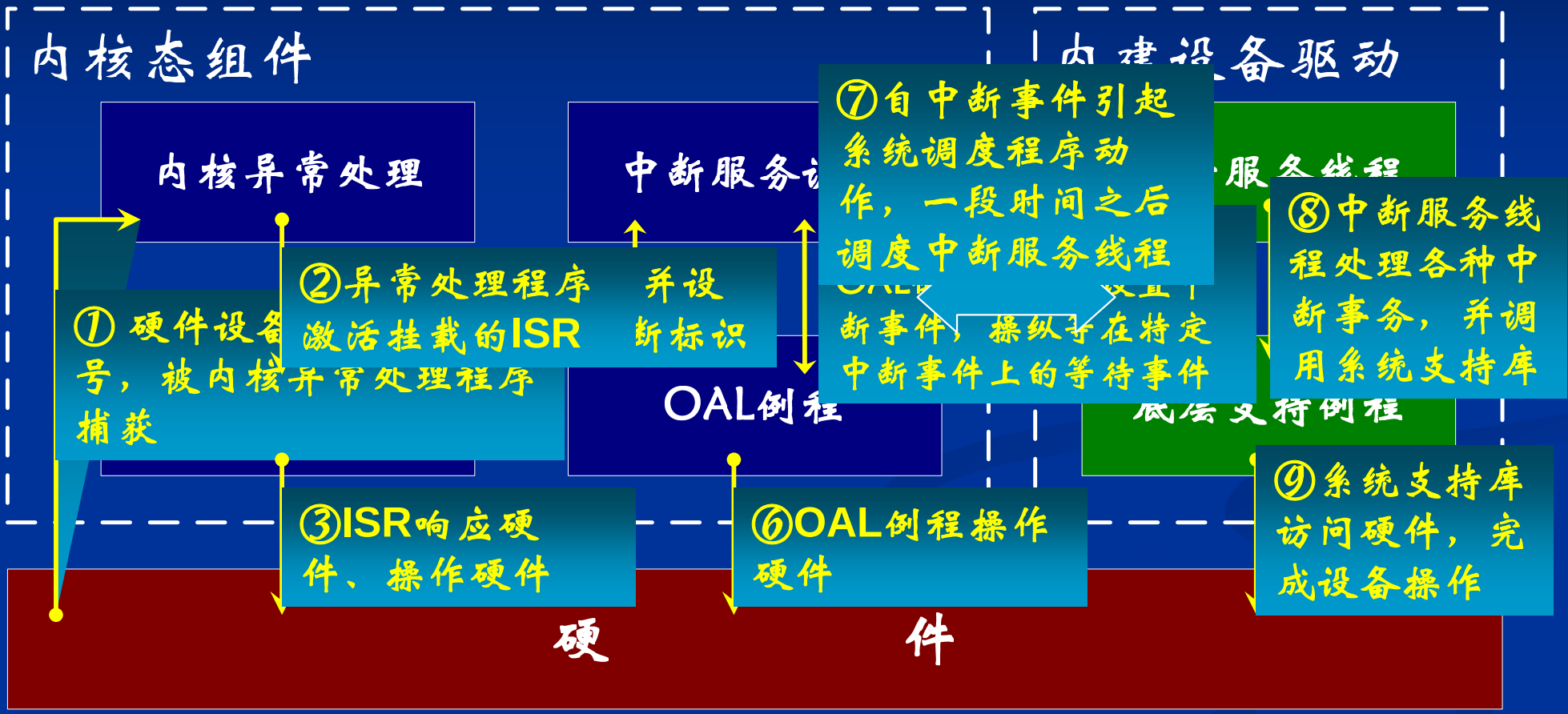
DDSI（Device Driver Service Provider Interface）：设备驱动服务提供上接口

3. 中断处理

在系统发现中断后，处理的过程分为2部分实现：内核的ISR和用户线程IST。

- ISR：一般要求短小精练，响应迅速。它们只实现最简单的功能：相应设备并返回一个中断标识给操作系统内核，Windows CE.NET支持两类类型的ISR；
  - 静态ISR：静态地编译进内核，运行时不能改变；
  - 可安装ISR：由内核管理程序从DLL中动态加载。
- IST：处理中断的事务性工作，当内核接到ISR传给自己的中断标识之后就发出一个中断事件，激活一个正等待在该事件的事件队列上的IST，一段时间之后，调度器将调度这个线程工作，处理中断的事务。

中断处理过程


内核态组件
内核异常处理
中断服务
OAL例程
①硬件设备
号，被内核异常处理程序
捕获
②异常处理程序
激活挂载的ISR
并设
断标识
③ISR响应硬
件、操作硬件
⑥OAL例程操作
硬件
⑦自中断事件引起
系统调度程序动
作，一段时间之后
调度中断服务线程
断事件，操纵等在特定
中断事件上的等待事件
⑧中断服务线
程处理各种中
断事务，并调
用系统支持库
⑨系统支持库
访问硬件，完
成设备操作
硬
件

4.6 用户界面与图形子系统

- Windows CE.NET 将Win32 API的User32和GDI32合并成一个新的模块gwes.exe，即GWES——Graphics、Window Manager、Event Manager Subsystem
  - ✓ Win32 应用编程接口(API)，用户界面 (UI)，和图形设备接口 (GDI) 库的组合
  - ✓ 是用户、应用程序和操作系统之间的接口
- GWES集成GDI,窗口管理器和事件管理器。
- GWES模块是Windows CE中最高度组件化的部分，包括两个部分：
  - ✓ USER：包含用户输入系统（User Input System）、事件管理器（Event Manager）和窗口管理器（Window Manager）3个组件
  - ✓ GDI：负责图形输出

4.6.1 用户输入系统

User部分主要包括：

- 消息队列
- 事件管理器
- 用户输入系统：将用户从键盘、鼠标及触摸屏等输入的消息通过消息传送机构送到相应的窗口

用户输入系统的主要组件包括：

- Msgque：消息队列
- Wmbase：创建窗口
- Winmgr：窗口管理器

1. 消息队列

① 消息队列的功能：

- 接收消息并将消息发送到相应的窗口
- 保存输入状态信息，如光标的大小、提示符闪烁率等

② 消息传送的两个基本函数

- SendMessage
  - ✓ 采用同步消息传送机制
  - ✓ 消息队列和线程存在一一对应的关系
- PostMessage
  - ✓ 采用异步消息传送机制
  - ✓ 每一个窗口都和一个与特定线程相关的消息队列联系在一起，窗口成为消息传送的目的地

消息的级别

级 别	消息类型
1	发送消息——调用SendMessage发送的消息，有最高的优先级
2	发送消息——调用PostMessage发送的消息，有最高的优先级
3	VM_QUIT消息
4	VM_PAINT消息
5	VM_TIMER消息

其它消息处理函数

GetMessage	从消息队列中得到消息
DispatchMessage	将GetMessage找回的消息分发给一个窗口程序
TranslateMessage	将一个键盘消息转换为字符消息
TranslateAccelerator	处理菜单命令对应的加速键
IsDialogMessage	确定一个消息是否为一个对话框所需要，如果是则处理这个消息
PeekMessage	检索一个消息，并存放这个检索信息到结构
RegisgerWindowsMessage	定义一个新的窗口消息，它保证将在整个系统中均可使用

2. 输入管理

输入管理由一整套子系统来完成，该子系统负责处理前台窗口、活动窗口和焦点窗口。

- 活动窗口：每个线程有一个特定的窗口称为活动窗口，是被特定线程拥有和激活的最高等级的窗口
- 焦点窗口：活动窗口及其子窗口可以是焦点窗口，焦点窗口能够接收来自键盘的消息
- 前台窗口：系统中一个特定的线程或者消息队列称为前台线程，前台线程中的活动窗口是前台窗口

4.6.2 图形设备接口（GDI）

- 通过把即将的使用绘图工具（画笔、画刷等）对象选入设备描述表中来完成对绘图工具的选择
- 设备描述表是所有绘图工具的集合
- 绘图操作使用所有被选入设备描述表的工具对象
- 绘图工具必须转化成与界面目标一致的形式，在选择绘图工具的同时，与绘图工具一致的界面对象也被选入了设备描述表中

1. 基本的GDI对象

- GDI中所有的东西都是一个C++对象，基类是一个被称作GDIOBJ的类。
- 与组件对象模型（COM）不同，当GDI对象的引用即使为0也不会自动删除，而需由程序员删除资源
- 基本GDI对象定义了一些抽象函数，规定了实际的GDI对象需要完成的任务
- 当应用程序退出而没有释放多余的内存空间时，操作系统负责将多余对象所占用的内存空间释放

2. 图形原语

- 主要是为了降低内存占用，Windows CE GDI不直接接触像素，而将所有信息送至驱动程序，由设备驱动程序完成像素输出
- GDI的原语有：
  - ✓ 矩形
  - ✓ 折线
  - ✓ 多边形
  - ✓ 椭圆
  - ✓ 圆角矩形
- 增加原语实际上是以空间换时间，违背了Windows CE的设计思想，对于复杂的图形宁可增加时间

3. 调色板

- Windows CE没有自己的调色板，当需要时由设备驱动程序提供
- 只须将一个调色板选入设备描述表，然后在设备上颜色就会在逻辑调色板和物理调色板之间进行一一影射
- 所有Windows CE的调色板都是一致性调色板
- Windows CE没有调色板管理器
- 可以得到最好的色彩转换
- 这种机制给了用户提供了更大的设计空间

4. 位图

- 位图是一个位数组，将其映射到输出设备上的矩形像素数组时就可以创建图像
- 用位图是可以创建、绘制、修改和存储图像
- 支持两种类型的位图：
  - 设备相关位图(DDB)：没有颜色表
  - 设备无关位图(DIB)：具有颜色表
- Windows CE支持独有的4色位图格式

5. 字体

- 字体是一组共享相同样式的图案符号，由它的字样、样式和大小表示
  - ✓ 字样决定了图案符号的特定特征
  - ✓ 样式决定了字体的颜色深度度和倾斜度
- Windows CE支持光栅字体和TrueType字体技术，但在特定系统中只能使用一种类型的字体，而且是在设计系统时决定的，应用程序无法改变
- 光栅字体和TrueType字体之间的区别与每个字符或符号的图案在各自的字体资源文件中的存储方式有关
  - ✓ 光栅字体的图案符号是一个表示单一字符的小位图，通常被认为与设备相关，不易缩放
  - ✓ TrueType字体的图案符号包含轮廓和提示，被认为与设备无关，易于缩放

- 字体的图案符号存储在字体资源文件中
  - ✓ 光栅字体的字体资源文件存储在一个.fot文件中
  - ✓ TrueType字体有两个文件：一个小的.fot头文件和一个包含实际数据的.ttf文件
- 使用字形高速缓冲存储器来减少显示字形的时间
  - ✓ 有两种控制的方法
    - ☞ 在建立Windows CE系统时就设置好它的大小
    - ☞ 当字体被丢弃时在Windows CE系统上运行独立的应用程序进行控制
  - ✓ 和字体句柄相联系
  - ✓ 默认容量为4KB

4.6.3 显示驱动程序接口(DDI)

- 是Windows NT DDI的子集，仅使用了最基本的图形引擎函数和驱动程序函数
- 显示设备驱动程序与Windows NT的差别
  - ✓ 都有相同的功能，GDI并不查询驱动程序的参数及性能信息
  - ✓ 当遇到复杂操作时显示驱动程序将回调GDI，使操作分成简单的几步
  - ✓ 被编译成DLL文件而不是LIB文件
- 显示驱动程序必须执行一套由GDI调用的DDI函数来完成初始化和显示图像的功能
- 显示驱动程序还使用图形原语引擎(GPE)类，使显示驱动程序对硬件加速更加容易