

Platform Builder 之旅（一）

作者：付林林

在以前的文章中我已经数次提到了 Platform Builder（简称 PB），并且简单的讲述了利用 PB v4.1 的定制平台向导（New Platform Wizard）创建了几个 CE 的平台。从这一篇文章起，我和 PB 爱好者们一起进入 PB 的世界，一起领略 PB 的风采，一起学习 PB。写每一篇文章对我来说都是一个进步，希望大家能够多提宝贵意见，多阐述看法。

相信看到我以前写的文章你已经会利用定制平台向导来定制简单的内核了。这次我仍然用这个向导，只不过采用自定义配置来选择平台的组件。自定义配置的优点是能够更细致的选择组件，这样定制的平台只包含我们需要的功能，而不要的全部抛弃。首先打开 "New Platform"，在 "Step 2" 中选择 "EMULATOR: X86"。在这里强调一点：能够运行 PB 创建的 CE 平台的环境有三种。第一种是**真实环境**，包括某一种 CPU 和支持的主板还有其它配置；第二种是**模拟环境**，PB 包含了模拟器，能够在 PC 上模拟运行 CE 平台；第三种是**PC 环境**，PB 可以创建能够在 PC 机运行的 CE 平台。并且提供了一组引导文件，通过这些引导文件可以加载 CE 内核文件（nk.bin）并启动 Windows CE。三种环境相比较，如果具备真实的环境那是最好了，模拟环境只能模拟简单的功能，PC 环境和模拟环境相似，只适合用来学习 PB。PC 环境需要的引导文件可以通过如下办法得到：在 PB 的安装所在位置（比如 C 盘）查找文件 Websetup.exe，运行这个文件，这个程序会解压出一个名为 WEBIMGNT.EXE 的文件。再查找名为 cepcboot.144 的文件，将 cepcboot.144 文件复制到同 WEBIMGNT.EXE 同一个目录下。插入软盘到 A 驱，然后在控制台程序（cmd.exe）下输入命令 "WEBIMGNT.EXE cepcboot.144"。在弹出的对话框中单击 "A drive"，程序把解压出来的启动文件都复制到软盘上。编辑启动文件中的 "Autoexec.bat" 批处理文件，如果 PC 配置的网卡是被支持的（微软推荐的网卡类型有 NE2000、SMC9000 兼容网卡、RealTek RTL8139 等），并且是 PCI 接口的，按如下修改：

```
set NET_IRQ=9
set NET_IOBASE=0
```

修改之后还要输入一个静态的 IP 地址。这个 IP 地址要和运行 PB 的电脑的 IP 地址处于一个子网内。实际上 PC 环境还是无法实现 CE 平台的大多数功能，所以对于要学习 PB 而又不具备真实环境的人来说，模拟器是比较合适的。

回到定制平台向导，在 "Step 3" 中单选 "Custom configuration"，输入平台名称和路径。在 "Step 4" 中必须单选 "Custom Device with Shell and Graphical User Interface"。表示平台将加入外壳程序和 GUI。有了资源管理器这样的外壳程序我们操作就方便多了。在 "Step 5" 中列出的是 "Application & Services"

Development"。这些都是用于软件开发的库。大多数支持库我们在 PC 上开发时早已熟悉了,有几个是 CE 独有的。比如"Simple Network Management Protocol" (简单网络管理协议),这是用在网络设备上的协议;"Pocket Outlook Object Model API",用于读取 Pocket PC 中"Inbox"软件中的数据;"Beta .NET Compact Framework"是.NET 的支持框架。这个在 PB v4.1 中不要选取,它在 PB v4.2 中才是正式版。其余的选项说明参见 CE 帮助文档(位置: **operating system development\windows ce.net overview\catalog features\applications and services development**)。"Step 6"中列出了 Windows CE 自带的应用程序。"Step 7"中列出了操作系统内核支持的服务。包括串口支持、并口支持、USB 口支持、调试工具、电源管理,还有一些其它特征。这里的调试工具不仅仅指能够用于调试的应用程序,还包括用于调试的 API 函数。Toolhelp 就是专门用于查看当前操作系统的进程及进程包含的线程、DLL 的信息。"Kernel Features"(内核特征)中的 Fiber API 是用于支持线程的 API。"Keyboard & Touch Driver Debugging Sample Applications"包括四个调试工具,用于调试触摸屏和键盘的驱动程序。"LMemDebug memory debugging hooks"用于查看当前操作系统正运行的程序的内存信息。在"Step 8"中列出的是网络特征(我有时称特征为组件,实质一样)。包含 CE 支持的所有网络协议。"Networking Features"中大多数的子项都要加入的,其它项按需加入。比如要支持红外线一定要加入 PAN 中的 IrDA。要拨号上网的一定要加入 WAN 所有项。"Step 9"中列出了存储特征。包含和存储设备相关的支持。具体包括存储管理器(支持 CDFS、FAT 文件系统)、数据库支持、ROM 和 RAM 文件系统、注册表存储支持。关于 ROM 和 RAM、FAT 我在以前的文章中提到过,这里就不再详细讲解了。"Step 10"中列出了 CE 包含的所有英文字体。要了解这些字体中每种字体所占空间大小,可以将鼠标放到字体名字的上 面,系统就会显示这种字体的大小。一般来说一个定制的 CE 平台其中字体尤其是中文字体占用空间是很大的,所以选择字体要慎重。除非必要,否则不要添加太多的字体。"Step 11"列出了国际化选项。包括各种语言支持包。在这里选择 "National Language Support[NLS]",再选择一种本地区语言,比如"Chinese {Simplified}"。"Agfa AC3 Font Compression"建议选择,这是一种字体压缩技术,适合中文字体。在"Chinese {Simplified}"第一子项中列出了字体。这一项选择至关重要,因为中文字体占用的空间太大了,直接影响 CE 平台的总体大小。具体选择哪些字体请参见 CE 帮助文档(位置: **operating system development\windows ce.net overview\catalog features\ International**)。从帮助文档中才能了解具体哪个子集包含哪些中文字体和字符集。"Chinese {Simplified}"其它子项包括中文输入法、输入法编辑器建议全部选择。"Step 12"列出了 CE 包含的 Internet 客户端程序、支持的组件和脚本。第一项"Browser Application"中建议选择第一子项"Internet Explorer 5.5 for Windows CE Components",而第二子项"Pocket Internet Explorer"是用于 PDA 上的功能较少的 Internet 浏览器。第二项"Internet Explorer 5.5 for Windows CE

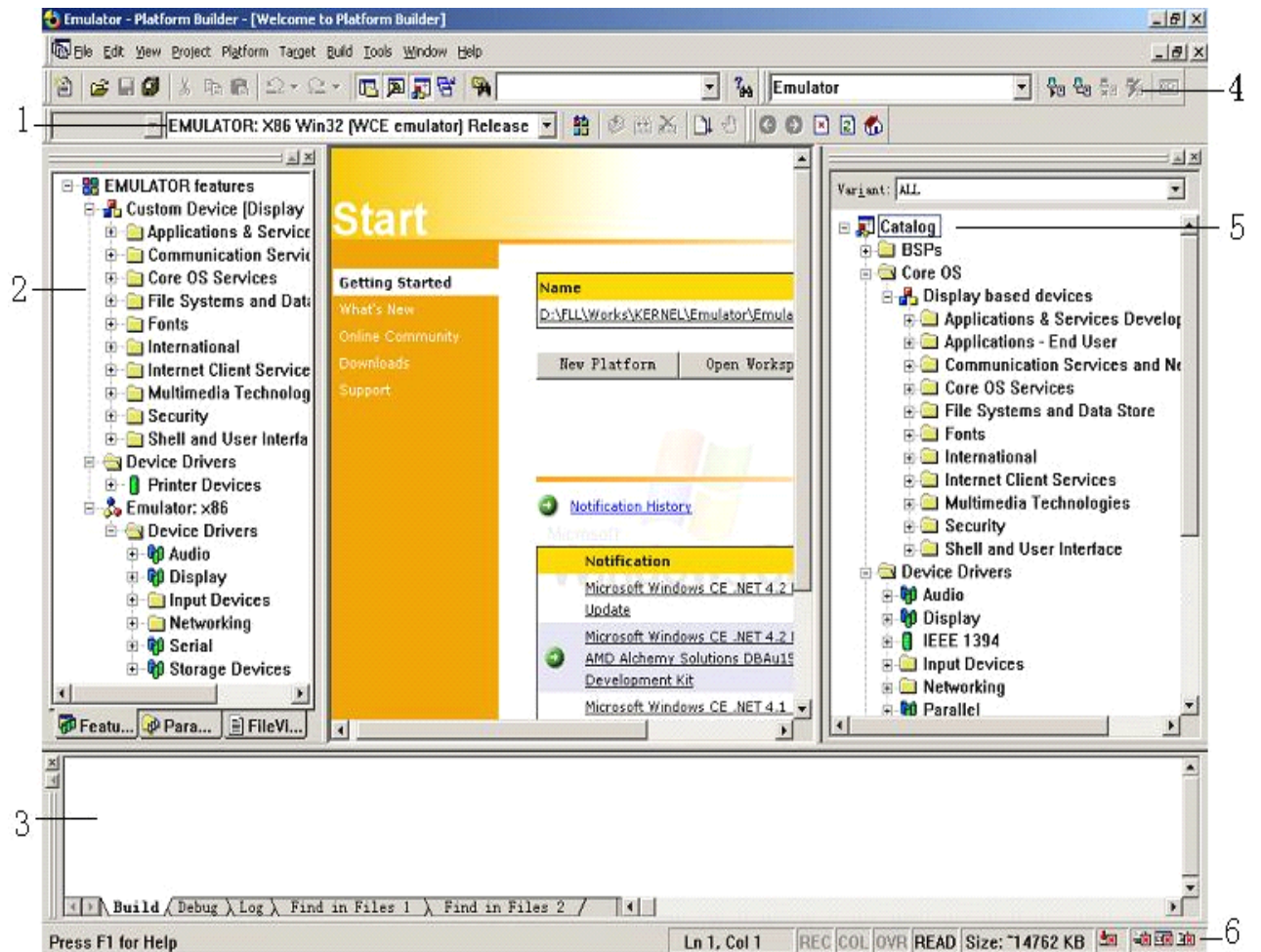
Components"中建议选择全部子项。这些子项都是开发 IE 的 API 函数、Active X 控件。第四项"Sample IE 5.5 Internet Options Control Panel"是"控制面板"中的用于修改 IE 选项的程序。和 Windows 桌面操作系统下"控制面板"中的"Internet 选项"一样。"Step 13"列出了 CE 支持的多媒体服务。可以任意选取要播放的各种媒体和具体媒体的格式。Direct X 也可以分拆选取。"Step 14"列出了 CE 支持的安全服务。"Step 15"列出了各种外壳和用户接口。外壳程序一般常常选择"Graphical Shell"中的标准外壳程序"Standard Shell",也就是资源管理器。"Step 17"是最后一步。单击"Done"按钮, PB 将用户选择的所有选项保存到脚本文件(*.wce)中。当编译 CE 平台时, sysgen.bat 批处理文件会调用 cesysgen.bat 批处理文件, cesysgen.bat 读取*.wce 文件的内容, 根据此文件的内容来设置全部的环境变量。cesysgen.bat 处理之后 sysgen.bat 调用 nmake.exe 程序根据环境变量来编译模块(针对源码文件, 如 C 语言文件)、复制模块(针对非源码文件, 如 DLL 等)。具体编译操作以及编译时 PB 所做的工作我将在下一篇文章中详细讲述。

Platform Builder 之旅（二）

作者：付林林

上一章所说，当用 PB 的"定制平台向导"选取了一个平台的所有特征（feature）后，接下来的工作就是编译了。即使你从来未曾编译过，你也能想象得到编译的时间一定很长。因为 PB 是在编译一个操作系统，而非一个应用程序。当然，越快的硬件环境编译的速度就越快。一般我编译一个内核需要 5 分钟到 10 分钟之间。

在编译之前你可能还要对这个 CE 平台的某些特征进行修改。为此，你必须熟悉 PB 的操作环境。下面图 1 是 PB 打开一个平台工程文件的界面。



图一 PB 主界面

图一中数字所指示的内容描述如下：

1. "Build"工具栏。此工具栏上按钮、下拉框都用于编译、调试。数字 1 指向的下拉框是编译指令集，可以指定不同的指令集来编译 CE 平台或者应用程序。
2. "Workspace"窗口。此窗口有三个子视图，分别为 FeatureView、ParameterView、FileView。当打开一个平台工程文件后，FeatureView 显示这个平台所有的特征。如设备驱动程序、各个软件组件等。ParameterView 显示所有平台通用的配置文件和当前平台的配置文件，这些配置文件扩展名为 *.bib、*.reg、*.db、*.dat。FileView 显示在当前 CE 平台上建立的应用程序源码文件、资源文件、资源脚本文件等。也就是说如果在当前 CE 平台上建立一个应用程序工程，那么所有的文件都在 FileView 中显示出来。类似 EVC、VC 的 "Workspace"窗口中的 "FileView"。注：关于 FeatureView 和 ParameterView 包含的内容在以后的文章中讲解。
3. "Output"窗口。用于显示输出信息。类似 EVC、VC 的 "Output"。
4. "Target"窗口。此工具栏上按钮分别用于下载内核文件到模拟器或实际平台、连接、断开。当一个 CE 平台编译好了之后，就可以按下载按钮将平台 (nk,bin) 文件下载到模拟器上运行。
5. "Catalog"窗口。这个窗口包含所有的 CE 支持的特征。
6. 状态栏图标。位于状态栏最右端的四个图标中，最左边的图标表示当前下载状态。另外三个表示三种服务状态，这三种服务运行在目标机 (target device) 上。在这里就是模拟器。

要向当前 CE 平台添加特征，首先在右边 "Catalog"窗口中找到要添加的特征，找到后移动鼠标光标到此特征上，然后单击右键，在弹出的菜单中单击 "Add to Platform"，PB 就将此特征添加到左边当前 CE 平台中。如果无反应说明此特征已经被添加进去了。要删除当前 CE 平台中某一个特征，移动鼠标光标到此特征上，单击右键，在弹出的菜单中选择 "Delete"。并非所有的特征都可以任意删除，因为有些特征是彼此关联的。这部分将在以后的文章中讲述。

对平台进行一些基本的设置，打开菜单 "platform" - "Settings"，检查 "Locale" 选项卡中地区和语言设置。这一点很重要，它决定着编译的平台采用的语言种类。再在 "Environment" 选项卡中添加 "IMGRAM64" 环境变量，值设置为 1。单击 "OK"，PB 开始将此环境变量加入到平台中。单击工具栏上 "Build Platform" 按钮开始编译。

在讲述编译过程前必须先了解环境变量以及如何读取和设置环境变量。一个环境变量包含了一个 CE 操作系统某一方面的信息。例如一个驱动程序、一个路径、一个配置文件、一个特征等。当 PB 编译 CE 平台时，先做的工作就是收集所有的环境变量供编译器使用。读取和设置环境变量的最好方法是单击 PB 菜单 "Build" - "Open Build Release Directory"，PB 会弹出一个控制台窗口，也就是命令行外壳。键入 "set" 命令，当前平台所有的环境变量就显示出来了，不过要多屏显示。为了看清楚每个变量的值，可以键入 "set |more"，这样就可以分屏

查看了。还可以将所有环境变量信息保存到硬盘上，比如键入"`set >C:\envi.txt`"。要查看单个环境变量值，键入"`set 环境变量名`"。要修改原环境变量的值键入"`set 环境变量名=值`"。有些环境变量无需键入值就可以达到修改目的。比如前缀为"`BSP_`"和"`SYSGEN_`"的变量，键入"`set 环境变量名=`"就取消了这个环境变量。在 IDE 中也可以修改环境变量，如上面所说的设置"`IMGRAM64`"的值，就是在 IDE 中修改的。利用环境变量也可以添加和删除特征，如 `BSP` 变量。`BSP` 变量分两种，一种以 `BSP_NO` 为前缀，一种以 `BSP` 为前缀。以 `BSP_NO` 为前缀表示当前平台不支持某一特征，以 `BSP` 为前缀表示支持这一特征。例如 `BSP_SERIAL2` 表示此 CE 平台支持串口 2；`BSP_NOSERIAL` 表示此 CE 平台不支持串口。如果在 PB 的"`catalog`"中找不到要添加的特征，可以通过设置 `BSP` 变量来实现。具体 `BSP` 环境变量参见 CE 帮助文档。

下面讲述整个编译过程中 PB 所做的工作：

- 执行 `cebuild.bat` 批处理文件。
- `cebuild.bat` 调用 `sysgen.bat` 批处理文件。
- `sysgen.bat` 调用 `cesysgen.bat` 批处理文件，`cesysgen.bat` 负责在*.wce 文件中搜索用户选择的特征，然后形成一系列环境变量。之后 PB 会显示这些变量，下面几个步骤就是显示收集的变量。
- 产生 `SYSGEN` 变量。每个 `SYSGEN` 变量对应一个特征。
- 产生 `CE_MODULE`、`COREDLL_COMPONENTS`、`FILESYS_COMPONENTS`、`DEVICE_COMPONENTS`、`GWE*_COMPONENTS`、`DCOM_MODULES`、`FONT_COMPONENTS` 等环境变量。其中每个环境变量包含某一个特征具体的内容。从环境变量名称就可以看出来是哪种特征。
- 对 `_DEPTREES` 环境变量指定的每个目录分别执行 `sysgen.bat` 批处理。`_DEPTREES` 这个变量的值是一些目录名(例如 `DCOM`、`IE`、`SERVERS`、`DIRECTX`、`WCESHELLFE` 等)，这些目录名位于 `%WINCEROOT%\public`。如果安装 PB v4.1 时默认安装路径，那么此目录路径为 `C:\WINCE410\Public`。
- 编译 `BSP`。因为我们采用的 `BSP` 是 `Emulator`，所以 PB 会编译 `%WINCEROOT%\Platform\Emulator` 目录下三个子目录 `KERNEL`、`DRIVERS`、`GWE` 中的源码文件。
- 清除 `_FLATRELEASEDIR` 环境变量指定的目录下的所有文件、子目录。假如我们定制的平台路径为 `C:\Emulator`，那么这个环境变量的值为 `C:\Emulator\RelDir\Emulator_X86Release`。
- 复制 `%_PROJECTROOT%` 所有文件到 `_FLATRELEASEDIR`。
- 根据本地地区环境变量，寻找所有与本地语言相关的*.str 文件复制到 `_FLATRELEASEDIR` 中。*.str 文件中包含了字符串资源，将字符串与 ID 关联。在 `_FLATRELEASEDIR` 目录下你可以看到以地区码为目录名的子目录。

- 处理 NLS（国家语言支持）数据。
- 执行 `fmerge.exe`。合并所有 `*.bib` 文件为一个文件 `ce.bib`，合并所有 `*.reg` 文件为一个文件 `reginit.ini`。
- 执行 `cebuild.bat` 批处理文件。
- 执行 `fmerge.exe`。合并所有 `*.db` 文件为一个文件 `initdb.int`，合并所有 `*.dat` 文件为一个文件 `initobj.dat`。
- 运行 `regcomp.exe` 压缩 `reginit.ini`。
- 运行 `txt2ucde.exe`。将整个 CE 平台涉及到的所有字符串转成 unicode 码。
- 运行 `res2exe.exe`。将所有 `*.dll`、`*.exe`、`*.cpl` 文件中的资源更新。资源更新部分主要和语言相关。
- 运行 `Romimage.exe`。将所有文件合并压缩成一个文件 `nk.bin`（默认文件名）。

整个编译过程被调用的批处理文件和 EXE 文件主要包括：`cebuild.bat`、`sysgen.bat`、`cesysgen.bat`、`nmake.exe`、`txt2ucde.exe`、`makeimg.exe`、`fmerge.exe`、`regcomp.exe`、`res2exe.exe`、`romimage.exe`、`build.exe`。在这里声明一点，我不保证所讲述的 PB 的编译过程一定是准确无误的。从总体上讲 PB 所做的工作就是这样。

Platform Builder 之旅（三）

作者：付林林

这篇文章主要讲解 PB 的配置文件。从用途方面分析，PB 包含两种配置文件。分别是源码配置文件和镜像配置文件。下面分别讲解这两种配置文件。

一、源码配置文件

源码配置文件用于编译源码时使用。这里的源码是指 Windows CE 公开的源码，如驱动程序、系统应用程序等。PB 在编译平台时将这些公开的源码即时编译并将编译链接后的文件复制到平台工程子目录里。记得前面讲过 PB 在开始编译时调用 `cebuild.bat` 批处理文件，`cebuild.bat` 执行的一个步骤是针对 `_DEPTREES` 变量指定的所有目录执行 `build.exe` 和 `sysgen.bat`。`build.exe` 在编译源码文件时会寻找当前目录下存放的源码配置文件，根据配置文件的信息来编译和链接，产生 EXE、DLL、LIB 文件。CE 的源码文件所在的目录中都包含了相应的配置文件，这些配置文件只对当前目录或者子目录的源码有效，具体分为三种：

- **DIRS** 文件：文件内容和解释如下：
 - DIRS**：指定哪个子目录的源码要被编译
 - DIRS_CE**：只有为 CE 编写的源码才被编译
 - OPTIONAL_DIRS**：指定可选的目录（很少使用这个选项），可以只编译指定目录而不是全部编译。
- **SOURCES** 文件：通过宏定义来指定编译和链接涉及到的文件，文件内容和解释如下：
 - TARGETNAME**：指定编译链接产生的主文件名
 - TARGETTYPE**：指定编译链接产生的文件的类型（决定了扩展名）。文件共分三种：`.lib`（LIBRARY）和 `.dll`（DYNLINK）和 `.exe`（PROGRAM）。
 - TARGETLIBS**：定义 `.lib` 链接文件，链接时需要这个文件。
 - SOURCES**：源码文件。包含扩展名为 `*.c` 或 `*.h` 或 `*.cpp` 的文件。
 - EXEENTRY**：`.exe` 文件的执行代码入口点。
 - sources.cmn** 文件是通用的 **SOURCES** 文件。在这个文件中可以指定作用于所有源码文件的配置选项。
- **MAKEFILE** 文件：包含默认的编译和链接选项
整个编译和链接过程：`build.exe` 收集编译和链接需要的数据（源码文件、链接文件、编译和链接选项）产生一系列的内部环境变量，然后调用 `nmake.exe`，`nmake.exe` 根据内部环境变量执行编译、链接，最后产生最终文件（`*.lib *.exe *.dll`）。

二、镜像配置文件：

镜像配置文件用于在制作 CE 镜像文件时使用。CE 的镜像文件扩展名为 .bin。制作镜像工具 romimage.exe 除了能够产生 .bin 文件外，还能够产生 .abx 和 .sre 文件。整个镜像的制作过程由 makeimg.exe 控制，它调用 cenlscmp.exe、fmerge.exe、res2.exe、txt2ucde.exe、regcomp.exe、romimage.exe 等。这些工具大部分在前面已经介绍了。镜像配置文件类型有 .bib、.reg、.db、.dat、.str。如果主文件名为 Common，表示是通用的配置文件。如果主文件名为 Platform，表示是某一个 BSP 的配置文件。如果主文件名是 Project，表示是定制的一个平台的配置文件。在 PB 中修改配置文件前如果没有把握最好先做好备份。

.bib (Binary image builder)

定义包含在内核镜像中的文件和模块的名称、加载位置。主要的 bib 文件有 Common.bib, Config.bib, Project.bib, Platform.bib 等。 .bib 文件内部分为几个部分：

【MEMORY】用于定义有效的物理内存块，在此将整个 RAM 分为几个部分。

格式： 名称 首地址 大小 内存类型

名称： 内存区域的唯一名称（RESERVE 是预定义名称，可以用多次，表示此区域保留）

首地址： 内存区域的首地址（十六进制表示）

大小： 内存区域的大小（十六进制表示）

内存类型：分为三种。

RAM: 运行所有进程的内存区域（整个区域必须是连续的，且不能含空洞）

RAMIMAGE: 专用于保存镜像的内存区域。（每个 .bin 中只能指定一个 RAMIMAGE）

RESERVED: 保留内存区域（这样的区域一般用于驱动程序使用，如显卡缓冲区、DMA 缓冲区）

举例：

```
;名称 首地址 大小 内存类型
IF IMGRAM64
NK 80220000 009E0000 RAMIMAGE
RAM 80C00000 03000000 RAM
UMABUF 83C00000 00400000 RESERVED
ENDIF
```

注：整个内核的地址都是从 0x8000 0000 开始的。如果是 x86 系列的 CPU，那么物理内存地址与虚拟地址映射关系在 oeminit.asm 中指定。

【CONFIG】 类似环境变量，PB 预设置了一些配置变量。常用的配置及说明如下：

AUTOSIZE:

格式：AUTOSIZE = OFF | ON

默认值为 OFF。在 config.bib 中的 MEMORY 部分定义了有效的内存区域，其中两部分 RAM、RAMIMAGE 分别用于进程使用区域和保存镜像区域。如果为 ON，romimage.exe 在创建 nk.bin 时将 RAM 和 RAMIMAGE 两部分合并成一个部分，然后从最低地址开始保留 RAMIMAGE 大小的内存，其余都作为 RAM 使用。

BOOTJUMP:

格式：BOOTJUMP = address | NONE

默认值为 NONE。每次重新启动 CE 内核，默认执行的代码从 RAMIMAGE 的首地址开始。如果在 BOOTJUMP 指定一个地址（必须在 RAMIMAGE 范围内），那么将从指定的地址开始执行。

COMPRESSION:

格式：COMPRESSION = OFF | ON

默认值为 ON。romimage.exe 在创建内核时默认压缩所有可写部分。对于文件，默认全部压缩。对于模块（.exe、.dll），默认压缩可写部分。模块的可写部分包括数据段，也就是在模块运行时一定加载到内存中的部分。如果模块在 .bib 中定义时具有 C 属性（表明压缩模块所有部分），那么当前这个选项就忽略了。

FSRAMPERCENT:

格式：FSRAMPERCENT = number

默认值为 0x80808080。指定为文件系统分配的内存的百分比。number 分为四个字节，由十六进制表示。

byte0 的值（单位为 4KB）表示在第一个 2MB 中，其中每 1MB 包含的 4KB 的整数倍。

byte1 的值（单位为 4KB）表示在第二个 2MB 中，其中每 1MB 包含的 4KB 的整数倍。

byte2 的值（单位为 4KB）表示在第三个 2MB 中，其中每 1MB 包含的 4KB 的整数倍。

byte3 的值（单位为 4KB）表示在剩下的内存中，每 1MB 包含的 4KB 的整数倍。

计算一下默认值 `0x80808080` 表示的百分比： $0x80*4K/1M = 0.5$ ，因为每个字节都等于 `0.5`，所以整个占用的百分比是 `50%`。

`KERNELFIXUPS`:

格式：`KERNELFIXUPS = OFF | ON`

默认值为 `ON`。如果为 `ON`，`romimage.exe` 创建内核前重定位内核到 `RAM` 的开始位置。

`OUTPUT`:

格式：`OUTPUT = path`

指定 `romimage.exe` 将创建完成的内核文件 `nk.bin` 放置到的路径。一般放置到 `%_FLATRELEASEDIR%` 下。

`ROMFLAGS`

格式：`ROMFLAGS = Flags`

设置内核选项的位掩码，多个位掩码可以组合使用。

- `0x0001` 禁止按需分页：`EXE` 和 `DLL` 默认是按需分页的。
- `0x0002` 禁止全内核模式：进程运行在两种模式下，用户模式和内核模式。全内核模式下所有线程运行在内核模式。全内核模式能够提高执行效率，但会增加系统的不稳定性。如果允许执行用户程序，那么不适合采用全内核模式。
- `0x00000010` 只信任来自 `ROM` 的模块（`DLL`、`EXE`）。默认 `ROM` 中的模块和所有文件系统的模块都是内核信任的。`OEM` 能够在 `OAL` 层实现对所有运行模块的检查，这个标志将忽略对来自 `ROM` 保存的模块的检查。
- `0x00000020` 停止刷新 `TLB`。这个标志仅用于运行在 `x86CPU` 上的内核。`TLB`（`Translation Look-aside Buffer`），有人翻译成变换索引缓冲区，它的作用是在虚拟地址和物理地址之间转换。对于具有实时性的内核，这个标志应该设置。
- `0x00000040` 按照 `/base` 链接选项中的地址加载 `DLL`。这样内核将不采用重定位加载 `DLL`。不建议采用。

`ROMSIZE`

格式：`ROMSIZE = size`

指定内核镜像的大小

`ROMSTART`

格式：ROMSTART = address

指定内核镜像的首地址

ROMWIDTH

格式：ROMWIDTH = width

指定数据宽度，一般为 32 位

ROMOFFSET

格式：ROMOFFSET = address

指定偏移地址。

SRE

格式：SRE = OFF | ON

指定 romimage.exe 是否产生 .src 文件，一般烧录 ROM 的程序能够识别此文件。

注：config 中绝大多数【CONFIG】选项不需要修改。凡是配置文件都可以使用 IF/ENDIF 条件语句。

【MODULES】定义镜像要包含的模块并指定模块（DLL、EXE）如何被加载到内存表中。

格式：模块名称 路径 内存块 类型

模块名称一般为模块的真实名称；路径为当前文件所处的位置（路径中指定的文件名和前面模块名称最好一致）；内存块是指这个模块将被存放到哪个内存块中，内存块的定义见前面 MEMORY 部分；类型指定这个模块将被存放的属性，具体类型如下：

- S: 系统文件
- H: 隐藏文件
- R: 只压缩模块的资源部分（默认模块是不压缩的）
- C: 压缩模块所有部分
- D: 禁止调试
- N: 标志模块是非信任的
- P: 忽略 CPU 类型
- K: 指定 romimage.exe 修正模块（仅用于调试或者内核跟踪）
- X: 指定 romimage.exe 对此模块验证签名
- M: 运行时加载整个模块，而不是按需分页

- L: 不分离 DLL 在进程地址空间和 Slot 1

举例：

```
MODULES
init.exe %_WINCEROOT%\RELEASE\INIT.EXE NK SH
nk.exe $(_FLATRELEASEDIR)\kitlnokd.exe NK SHD
nk.exe $(_FLATRELEASEDIR)\kitlnokd.exe NK SHN
```

【FILES】定义镜像要包含的文件并指定文件如何被加载到内存表中。
格式：模块名称 路径 内存块 类型

具体类型如下：

- S: 系统文件
- H: 隐藏文件
- U: 不压缩文件（默认是压缩的）

举例：

```
FILES
initobj.dat %_WINCEROOT%\RELEASE\INITOBJ.DAT NK SH
```

【.dat File System File】定义目录和指定文件位置。当冷启动 CE 平台时，fileysys.exe 用这些数据创建目录、快捷方式、文件（在 RAM 文件系统）。

举例：

```
;;创建根目录下子目录 Program Files
root:-Directory("Program Files")
;;创建目录 Program Files 下一个子目录 My Projects
Directory("\Program Files"):-Directory("My Projects")
;;复制文件从\Windows\Myproj.exe 到\Program Files\My
Projects\My Project.exe
Directory("\Program Files\My Projects"):-File("My
Project.exe", "\Windows\Myproj.exe")
;;复制文件从\Windows\control.lnk 到\control.lnk
root:-File("\control.lnk", "\Windows\control.lnk")
```

快捷方式的运用：如果要在 CE 平台的桌面上显示一个程序的快捷方式，实现步骤为：

1. 创建一个快捷方式。在开发机上用记事本一类的文字编辑软件写入字符格式为：长度#路径。其中长度为路径的字符个数。

例如"16#\windows\abc.lnk"，注意路径中空格也算在内。完成后保存为

ASCII 码的扩展名为 .lnk 的文件。

将此快捷方式文件 *.lnk 复制到 %_FLATRELEASEDIR% 下,也就是 PB 编译的所有文件存放的目录。

2. 在 project.bib 中的 FILES 部分下按 FILES 的格式键入字符。例如:

FILES

abc.lnk \$(_FLATRELEASEDIR)\abc.lnk NK S

3. 在 project.dat 中指定 abc.lnk 文件所存放的位置。例如:

root:-Directory("\Windows")

Directory("\Windows"):-Directory("LOC_DESKTOP_DIR")

Directory("\Windows\LOC_DESKTOP_DIR"):-

File("abc.lnk","\Windows\abc.lnk")

【.reg Registry file】设置注册表项。关于注册表见我的文章《开发实例二：保存信息》。

数据类型	格式
REG_SZ	"my string"
REG_DWORD	DWORD: NNNNN (十六进制)
REG_MULTI_SZ	multi_sz: "my string"
REG_BINARY hex:	xx, xx, xx, xx ...
HEX hex (xxxxxxxx):	xx, xx, xx, xx

例如:

```
[HKEY_LOCAL_MACHINE\init]
```

```
"Launch60"="myproc.exe" ///REG_SZ 类型
```

```
"Depend60"=hex:14,00,1e,00 ///REG_BINARY 类型
```

```
[HKEY_LOCAL_MACHINE\System\StorageManager\Profiles\TRUEFFS_UN  
AND\FATFS]
```

```
"MountFlags"=dword:2 ///REG_DWORD 类型
```

【.db The database files】数据库文件保存在对象存储中。实际应用的不多,在这里不再过多讲解。

【.str string files】类似 EVC 中的字符串资源。

用于指定 ID 与字符串的关联。CE 支持很多国家语言,所以内核使用的字符串可能采用不同国家的语言。为此,CE 将字符串用 ID 来定义,在 .str 文件中指定 ID 对应的字符串。包含 .str 文件的目录名采用国家码来设置,例如 "C:\.....\0410\cepc.str"。

Platform Builder 之旅（四）

作者：付林林

本篇文章是对前面讲过的关于 PB 的系列文章做一些补充，因为 PB 包含的知识面太广也太杂，所以针对一些杂项归纳在一起写成这一篇文章。当然一篇文章绝不可能包罗万象，有时间我会陆续写出来。

- **【Windows CE 安装目录】**

Windows CE.NET v4.1 默认安装路径为 C:\wince410。其子目录名及目录包含内容如下：

Others 包含 ATL、MFC 运行时文件等。

Platform 包含所有 **BSP** 子目录。每个目录包含 **BSP** 文件。

Private 包含大多数 CE 公开的源码

Public 包含大多数 **SDK** 文件、配置文件、编译工具。

SDK 包含 PB 使用的工具

public 目录下一些子目录名及包含内容如下：

Common 核心操作系统模块

Datasync 同步数据传送模块

Dcom DCOM 模块

DirectX directx、DVD-VIDEO 模块

IE IE 模块

NetCF .NET 框架模块

RDP 远程桌面模块

Servers HTTP 服务扩展模块

Script JavaScript 和 VBScript 脚本模块

Speech SAPI 模块

Viewers FileView 模块

- **【build options】**

PB 菜单 "Platform" - "Settings" 中常见的编译选项。

Enable CE Target Control Support: 支持对实际平台的控制。可以执行 CESH 命令。

Enable Event Tracking During Boot: 在引导过程中支持事件跟踪。

Enable Full Kernel Mode: 支持全内核模式。全内核模式参见配置文件。

Enable Kernel Debugger: 允许调试内核。




Enable KITL: KITL (内核独立传输层) 用于在开发平台和实际平台之间相互通信。

Enable Profiling: 能够评测内核的性能。评测内核性能的工具在以前讲

过。

- **【Object Icon Types】**

Object Icon 是指 PB 中 "Workspace" 和 "Catalog" 列出的对象对应的图标。当定制了平台后，如果在 "Workspace" 中删除某些特征时，PB 很可能弹出对话框说明此特征无法删除。本节将讲述其中原因，之前要介绍 "Workspace" – "FeatureView" 中相关图标。

 特征组  锚定特征  非锚定特征

1、把一个特征组缩减成一个特征叫 **Resolve** (分解)。点击菜单 "platform" – "resolve feature(s)", 弹出一个对话框显示整个 CE 平台所有能够分解的特征组，可以在每一个特征组中选择单个特征。比如显卡驱动组，当从 PB 的 "catalog" 中加入显示驱动时，即使加入一个驱动，PB 也会在 "FeatureView" 显示一个特征组。通过 **Resolve** 可以任意选择一个需要的显卡驱动。

2、锚定特征。在 PB 创建一个 CE 平台时，有一些特征默认被加上锚定特征 (anchor feature)。用户从 PB "catalog" 中选择特征加到左边项目里时，这个加入的特征被自动加上锚定特征。同时 PB 将检查 CE 平台中的所有特征是否与新加入的特征冲突，然后根据 **cesysgen.bat** 中包含的从属规则，将和用户选择的特征相关的特征加到项目里。**由 PB 根据从属规则带到项目里的特征是非锚定特征 (非用户选择)。**

3、查看特征从属关系。通过右键菜单 "feature dependencies" 选项来查看指定特征的从属关系。"Depends on" 表明当前特征需要哪些其它特征，"Dependency of" 表明哪些特征需要当前特征。

4、每次从定制的 CE 平台中删除或者加入特征，PB 都会在执行删除或者添加特征后重新整理平台的所有特征。在 "Output" 窗口中可以看到平台所有增加的非锚定特征。

5、删除特征：**锚定特征可以直接删除，而非锚定特征必须先删除它的父特征 (父特征在加入到平台时将子特征带进来) 后才能被删除。**另外锚定特征也可能和另一个锚定特征属于从属关系。所以**锚定特征也可能不允许被直接删除。**

通过以上 5 点读者一定会了解为什么有些特征不能被直接删除了。

- **【cec 文件】**

.cec 文件是 **目录特征** 文件。这个文件用于将自定义的目录特征导入到 PB 的 "catalog" 中。适合于将编写的驱动程序、BSP 等给其它开发商使用，开发商只需加载 cec 文件后就能够将驱动、BSP 等加入到自己的平台上。用记事本就可以打开查看 .cec 文件的内容。一般编写 .cec 文件采用 PB 的工具 "cec editor", 单击 PB 菜单 "tools" – "cec editor", 弹出一个窗口，窗口

标题为"platform Builder CEC Editor"。

1、编写.cec文件：假如我们要加入一个驱动程序，这个驱动程序包含两个文件，一个驱动文件 mydll.dll，另一个驱动注册信息 mydll.reg。单击"cec editor"的"create a new cec file"按钮。在新窗口左边树型控件中"catalog"项位置单击右键，在弹出的菜单中单击"insert feature grout..."，在弹出的对话框"insert feature group"的"name"中输入特征名字例如"mydll"，带"*"的必须填写，其它可以不添。在窗口左边树型控件中"mydll"项位置单击右键，在弹出的菜单中单击"insert feature"，在弹出的对话框的"name"中再次输入"mydll"。接着在窗口左边树型控件中"mydll"项位置单击右键，在弹出的菜单中单击"insert build method"，在弹出的对话框中选择支持的 CPU 类型。在此对话框中的"Step"中共有 12 个选项，将 PB 编译平台的过程分为四个部分，每个部分又加入"Pre"和"Post"表示此前和此后。这四个部分分别为 CESYSGEN（执行 cesysgen.bat 过程）、BSP（编译 BSP 等源码文件的过程）、BuildRel（复制文件的过程）、MakeImg（制作镜像文件的过程）。按照我举的例子，我只需复制这两个文件到 %_FLATRELEASEDIR%中，所以在 MakeImg 之前执行即可，选择"PreMakeImg"。在窗口左边树型控件中"PreMakeImg"项位置单击右键，在弹出的菜单中单击"insert action"—"copy"，在弹出的对话框中指定源文件路径和目的路径。目的路径可以输入 "\$(_FLATRELEASEDIR)"。因为有两个文件，所以要重复操作一遍，将另一个文件也复制过去。最后保存。

2、导入到 PB 的"catalog"中：单击工具栏上"add the cec file to the catalog"按钮将此 cec 文件导入到"catalog"中。

3、删除.cec文件：如果加入到"catalog"后，单击 PB 菜单"File"—"Manage Catalog Feature"，找到你的 cec 文件，单击"remove"。

- **【loadcepc.exe】**

loadcepc.exe 是一个 MS-DOS 程序，它是一种 Boot Loader。Boot Loader 的主要工作是加载 CE 平台（nk.bin），将 nk.bin 解压后的所有文件加载到内存，然后将 CPU 的控制权交给 CE 内核，CE 内核执行初始化工作，运行 nk.exe 实现操作系统内核功能、运行 device.exe 管理常用的设备驱动程序、运行 filesys.exe 加载文件系统、运行 gwes.exe 管理图形窗口事件子系统等等。采用 x86 CPU 的硬件系统共有两种启动模式，一种是采用 BIOS(基本输入输出系统)实现硬件的检测和初始化，之后启动 MS-DOS 操作系统，运行 loadcepc.exe 加载 nk.bin。另一种方法是采用 The x86 ROM boot loader (romboot)，它是一个很小的引导程序，有 256KB 大小。可以将它存放到 Flash/EEPROM 中替换 BIOS 程序，它能够实现硬件的检

测和初始化，在这之后如果系统采用硬盘等 IDE 接口存储设备，那么 romboot 会自动寻找活动分区上的 nk.bin 文件并加载。romboot 的优点是检测速度和加载速度都很快，但是在支持的硬件系统方面不如 BIOS 全面。

loadcepc.exe 支持通过并口、串口、网卡从开发机上下载 nk.bin 文件。在 loadcepc 后需要指定要加载的 .bin 文件的文件名，如果为 nk.bin，那么可以不指定文件名，如果不是 nk.bin，那么必须指定文件名，例如 "loadcepc abc.bin"。下面简单介绍几个常见的参数：

- /B: 指定串口的波特率。例如 /B:19200
- /C: 指定串口的端口。1 指 "COM1:"，2 指 "COM2:"。 例如 /C:1
- /D: 指定显示分辨率。0 指 320 x 200，1 指 480 x 240，等等。
- /E: 指定网卡 IO 地址和 IRQ。例如 /e:300:5
- /L: 指定显示分辨率和色深。它需要指定具体的分辨率，所以能够指定不标准的分辨率。例如 /l:768x576x8，
表示分辨率为 768 x 576，颜色位数为 8 位。
- /P: 指定使用并口传递数据。
- /Q: 指定使用串口传递数据。
- /V: 指定当 loadcepc 加载时添加状态信息。

• 【调试平台和运行程序】

要调试平台，先要配置远程连接。单击 PB 菜单 "Target" — "Configure Remote Connection"，在弹出的菜单中将 "Download" 和 "kernel" 下拉框都设置为 "Emulator"，单击 "Download" 后面的 "Configure"，在弹出的菜单中将 "Memory (MB)" 设置为 64MB，因为前面我们设置了 "IMGRAM64"。分辨率设置为 800x600。退出 "Configure Remote Connection" 后单击 "Target" — "Download/Initialize" 将 nk.bin 下载到模拟器上执行。如果向定制的平台添加或者删除特征后，当执行 "Build Platform" 时，PB 不会全部重新编译，而是将修改的部分重新编译。要在定制的 CE 平台上运行应用程序或者 DLL，先要将 EXE 或者 DLL 复制到 %_FLATRELEASEDIR% 中，再单击 PB 菜单上 "Target" — "Run Programs"，在程序列表中找到指定的 EXE 文件，再单击 "Run"，在模拟器上就可以看到你所运行的应用程序了。因为 EVC 附带的模拟器不支持中文，所以有些应用程序需要到 PB 的模拟器上运行。CE 的帮助文档中介绍了如何在 PB 的模拟器上对应用程序设置断点进行调试，但是我没实验成功。另外凡是基于 x86 指令集编译的 EXE、DLL 都可以在模拟器上运行，除非 EXE、DLL 包含的功能模拟器无法模拟（模拟器的限制在我以前的文章中有说明）。

Platform Builder 之旅（五）

作者：付林林

随着 CE 的发展，对象存储（Object Store）的作用越来越小，而大容量的永久存储设备被越来越多地采用，这一章将针对 CE 的文件系统阐述相关的知识，让 PB 开发者除了能够加入对永久存储设备的支持，还能做一些优化。记得在以前讲过的文章中提到了如何在 PB 中向定制的内核加入对硬盘、光驱的支持（包括 ATA 设备驱动和各种文件系统），在这里就不再重复了。

CE 提供了三种文件系统，基于 ROM 的文件系统、基于 RAM 的文件系统、FAT 文件系统。FAT 文件系统使用的范围最广，能够应用在 ATA 设备、Flash 存储设备、SRAM 存储设备上，另外 CE 还允许开发者自己编写并注册一套文件系统，只要接口符合 Win32 文件系统 API 即可。

CD/UDFS 文件系统

这两种文件系统被用于读取 CD、DVD 等。除了通过在 PB 的“catalog”中加入这个文件系统外，还可以在 PB 中添加 SYSGEN_UDFS 环境变量来实现。CDFFS 和 UDFS 在注册表中的注册信息如下：

```
; Default values for udfs. These can be overridden per
profile.
[HKEY_LOCAL_MACHINE\System\StorageManager\UDFS]
    "FriendlyName"="CDFFS/UDFS FileSystem"
    "Dll"="udfs.dll"
    "Paging"=dword:1
[HKEY_LOCAL_MACHINE\System\StorageManager\Profiles\CDProfile]
    "Name"="IDE CDROM/DVD Drive"
    "Folder"="CDROM Drive"
"DefaultFileSystem"="UDFS"
"PartitionDriver"=""
```

上面注册表信息在文件 common.reg 中。注册表数据是从上至下有效，也就是说下面的数据可以覆盖上面的数据。从注册表数据可以看出 udfs.dll 包含了 UDFS 文件系统的驱动程序，CDROM 的驱动器名为“CDROM Drive”，采用的文件系统为 UDFS，没有分区驱动程序。如果我们要访问 CDROM 的目录或者文件就要在名字前加“\CDROM Driver\”。注意，可能你的 common.reg 文件中的数据在“Folder”处不同于上面，比如为“Folder”=LOC_STORE_CD_FOLDER，那么你可以直接按照上面数据更改，或者在*.str 文件中查找 LOC_STORE_CD_FOLDER，找到这个 ID 对应的字符串再更改（查找到的文件可能很多，应查找以本国家码

为目录名的目录)。

FAT 文件系统

除了安全性外，FAT 文件系统是一个很优秀的文件系统，很适合在嵌入式设备中使用。CE 也把 FAT 作为外部存储设备的通用文件系统。添加 FAT 文件系统的环境变量为 `SYSGEN_FATFS`。随便列出 CE 提供的操作 FAT 文件系统的函数：

<code>DefragVolume</code>	碎块整理，在碎块整理前先进行磁盘扫描。
<code>DefragVolumeUI</code>	同上，但是包含一个选项对话框。
<code>FormatVolume</code>	按要求格式化分区。
<code>FormatVolumeUI</code>	同上，但是包含一个操作对话框。
<code>ScanVolume</code>	扫描一个分区的 FAT 和目录。
<code>ScanVolumeUI</code>	同上，但是包含一个操作对话框。

FAT 文件系统在注册表中的注册信息如下：

```
; Default values for fatfs. These can be overridden per profile
```

```
[HKEY_LOCAL_MACHINE\System\StorageManager\FATFS]
```

```
    "FriendlyName"="FAT FileSystem"
```

```
    "Dll"="fatfsd.dll"
```

```
    "Flags"=dword:00000024
```

```
    "Paging"=dword:1
```

```
    "CacheSize"=dword:0
```

从注册表数据可以看出 `fatfsd.dll` 包含了 FAT 文件系统的驱动程序。对“Flags”值的描述如下：

标志	值	描述
<code>FATFS_UPDATE_ACCESS</code>	<code>0x00000001</code>	更新访问时间
<code>FATFS_DISABLE_AUTOSCAN</code>	<code>0x00000004</code>	不能自动调用 <code>ScanVolume</code>
<code>FATFS_VERIFY_WRITES</code>	<code>0x00000008</code>	检验所有写操作
<code>FATFS_ENABLE_BACKUP_FAT</code>	<code>0x00000010</code>	备份 FAT 表
<code>FATFS_FORCE_WRITETHROUGH</code>	<code>0x00000020</code>	让系统可以直接将任何缓

	0	缓冲区中的数据写到磁盘上，这样系统将加快写数据到磁盘的速度
FATFS_DISABLE_AUTOFORMAT	0x0000004 0	禁止自动格式化未格式化的分区
FATFS_DISABLE_COMPCHECK	0x0000008 0	禁止自动检测压缩分区

“Paging”用于指定是否能够分页，值为 1 可以分页，0 不能分页。“CacheSize”用于指定 FAT 文件系统缓冲区大小。具体的值是用 16 进制数表示的扇区数量。假如“CacheSize”的值为 400，那么用于 FAT 缓冲的字节数为 $1024 * 512 = 512KB$ 。

存储管理器默认的文件系统是 FAT 文件系统，所以硬盘、USB、PCMCIA 等存储设备的注册表信息默认都没有指定文件系统。

文件系统过滤器

文件系统过滤器是一个 DLL。存储管理器在调用文件系统 API 之前先调用文件系统过滤器的过滤函数，通过过滤函数能够实现对文件数据的加密、解密、压缩甚至扫描文件是否存在病毒。实现文件系统过滤器的步骤是先编写 DLL，CE 提供了例子程序，位于 `%_WINCEROOT%\Public\Common\OAK\Drivers\FSD` 下。添加代码之后再修改注册表数据。注册表位置如下：

1. HKEY_LOCAL_MACHINE\System\StorageManager\Profiles\ProfileName\FileSystem\Filters
2. HKEY_LOCAL_MACHINE\System\StorageManager\FileSystem\Filters
3. HKEY_LOCAL_MACHINE\System\StorageManager\Filters

注册表键及其键值例子如下：

```
"Dll" := "fsdspy.dll" ////DLL 名称
Order = 0 ////顺序
```

这里 ProfileName 是指 Profile 的名称，比如 HDProfile。FileSystem 是指具体的文件系统，如 FATFS, UDFS, RELFSD。数字 1 指出的文件系统过滤器对指定存储硬件的文件系统有效；数字 2 指出的文件系统过滤器对指定的文件系统有效；数字 3 为所有文件系统多有效。

存储管理器

存储管理器 (Storage Manager) 是 Windows CE .NET 的新功能, 主要管理存储设备驱动程序、文件系统驱动程序、分区驱动程序、文件系统过滤器。存储管理器根据注册表数据来加载需要的模块。具体注册表数据如下:

```
[HKEY_LOCAL_MACHINE\System\StorageManager]
"Dll"="fsdmgr.dll"
"PNPUnloadDelay"=dword:1000
```

`fsdmgr.dll` 包含存储管理器的代码。“PNPUnloadDelay”是指存储管理器在接收到即插即用设备的卸载通知后的延时时间。具体存储管理器需要管理的存储设备的注册表信息在

`HKEY_LOCAL_MACHINE\System\StorageManager\Profiles` 下。例如硬盘的注册表信息如下:

```
[HKEY_LOCAL_MACHINE\System\StorageManager\Profiles\HDProfile]
"Name"="IDE Hard Disk Drive"
"Folder"="Hard Disk"
"AutoMount"=dword:1
"AutoPart"=dword:0
"AutoFormat"=dword:0
"MountFlags"=dword:0
"FileSystem"="fatfsd.dll"
"PartitionDriver"="mspart.dll"
```

“Name”指名称; “Folder”指目录名, 可以在此更改目录名称; “AutoMount”指如果检测到分区后就自动安装文件系统; “AutoPart”指如果没有分区则自动将最大可用空间划分成一个分区; “AutoFormat”指如果没有格式化则自动格式化; “FileSystem”指定这个存储设备采用的文件系统, 如果不指定就采用默认的文件系统; “PartitionDriver”指定分区驱动程序; “MountFlags”指文件系统如何被安装。值为 1 指定一个隐藏文件系统, 如果文件系统被隐藏, 那么这个文件系统将不会被查找文件的 API 发现, 但是如果指定文件的绝对路径, 还是可以访问的。值为 2 指定当前文件系统能够包含系统注册表。一个存储设备第一个分区将包含基于 HIVE 的系统注册表 (关于基于 HIVE 的注册表见以前讲过的文章)。值为 4 指定这个文件系统作为整个文件系统的根 (\), 这里要说明一下, CE 默认对象存储作为整个文件系统的根 (\), 当在根下放置一个文件时, 例如 `\a.dat`, 这个文件实际存放在对象存储中。如果指定其它文件系统作为根, 那么文件将存放在这个文件系统所在的存储设备中。值为 8 指定当指定值为 4 时隐藏 ROM。

Platform Builder 之旅（六）

作者：付林林

增加对大容量物理内存的支持和永久存储注册表是在定制内核工作中常遇到的问题。本篇文章将对这两个方面阐述相关的知识并指导读者如何在 PB 中实现。

对大容量物理内存的支持

在 PC 上增加物理内存是很方便的，插上内存条后只要自检程序识别，那么桌面操作系统就能够支持。而在基于 CE 的产品上就没那么简单了。如果物理内存大于 64MB，就要在定制内核时做一些工作。

一旦内存管理单元（MMU）开始工作，CPU 就不再直接访问物理内存了，对于运行在 x86 和 ARM 系列 CPU 上的 CE 内核来说，必须先确立物理内存地址同虚拟内存地址的映射关系。这种关系实际是在一个名为 OEMAddressTable 的表中定义的。这个表在前面的文章中已经提到过。CE 提供了两种虚拟地址映射方法，分别为静态映射和动态映射，这个表属于静态映射方法。静态映射的虚拟地址空间只能由内核访问，而动态映射的地址空间可以由用户模式的应用程序访问。OEMAddressTable 在文件 %_WINCEROOT%\Public\Common\Oak\Csp\i486\Oal\OEMInit.asm 中。在这个文件的最后有一段代码：

```
; RAM 0x80000000 -> 0x00000000, size 64M
dd 80000000h, 0, 04000000h
dd 0, 0, 0
```

这段代码表示将虚拟地址 80000000 映射到物理地址 0，大小为 64MB。将 04000000h 改成实际的物理内存大小，然后保存。接着单击 PB 菜单"Build"—"Open Build Release Directory"，在命令行中先用 cd 命令进入上述目录，如：

```
cd %_WINCEROOT%\Public\Common\Oak\Csp\i486\Oal
```

然后键入下列命令：

```
build -c
```

```
sysgen i486oal
```

build 命令根据配置文件内容编译整个目录，**sysgen** 批处理将 **build** 编译的文件 **i486oal.lib** 文件复制到 CE 的安装目录和内核工程目录下。我安装的 **BSP** 是基于 **x86** 的，所以相应目录为

%_WINCEROOT%\PUBLIC\COMMON\OAK\LIB\X86\RETAIL 和

%_PROJECTROOT%\cesysgen\oak\lib\x86\retail 两个目录。上一步做完之后，接着开始修改 **config.bib** 文件。在以前的文章中讲过在 **config.bib** 文件中定义内存区域。在 **config.bib** 中预设的配置没有超过 **64MB** 的，所以要自己手工添加。

可根据原有的 **IMGRAM64** 配置更改，原有的 **IMGRAM64** 如下：

```
; 64 MB of RAM (note: AUTOSIZE will adjust boundary)
```

```
IF IMGRAM64
```

```
    NK          80220000  009E0000  RAMIMAGE
```

```
    RAM         80C00000  03000000  RAM
```

```
    UMABUF      83C00000  00400000  RESERVED
```

```
ENDIF
```

假如要支持 **128MB**，更改如下：

```
IF IMGRAM16 !
```

```
IF IMGRAM32 !
```

```
IF IMGRAM64 !
```

```
    NK          80220000  009E0000  RAMIMAGE
```

```
    RAM         80C00000  07000000  RAM
```

```
    UMABUF      87C00000  00400000  RESERVED
```

```
ENDIF
```

```
ENDIF
```

```
ENDIF
```

在这里没有更改 **NK** 的大小，只是修改了 **RAM** 的大小。在 **config.bib** 定义之后，还可以在 **OAL** 层通过变量或者函数更改物理内存的大小，适合设备可能出现增加或减小内存的情况。**CE** 的帮助文档介绍了几种方法，这里只提一下

CreateStaticMapping 函数。**CreateStaticMapping** 函数作为 **config.bib** 文件的补充，适合在用户模式的应用程序或驱动程序中调用，调用这个函数能够将指定首地址的物理内存块映射到虚拟地址空间，函数返回虚拟地址。虚拟地址范围在 **C400 0000** 到 **E000 0000** 之间，这是内核的地址空间，只能由内核访问。相比较 **VirtualCopy** 函数用于动态地将指定首地址的物理内存块映射到虚拟地址空间，这个虚拟地址空间一般为用户进程的地址空间，因为 **VirtualCopy** 函数被设计专用于驱动程序调用，它常被用于将一个物理内存块映射到不同的虚拟地址空

间。

实现永久保存注册表数据

关于注册表在前面的文章中已经介绍过了，这里只讲述如何实现永久保存注册表数据。

注册表类型分为基于对象存储的注册表和基于 **HIVE** 的注册表，在定制内核的时候只能选择其中一种。从理论上讲这两种注册表都能够实现永久保存注册表数据，但是采用不同的类型会影响 **CE** 的启动顺序和启动速度，还会影响内存的使用量。我还是趋向于采用基于 **HIVE** 的注册表来实现永久保存注册表数据，这也是一个发展趋势。在讲解之前先简单描述如果 **CE** 采用基于 **HIVE** 的注册表，那么在启动时如何加载已经保存的注册表数据：

1. **nk.exe** 执行，启动 **fileSYS.exe**。
2. **fileSYS.exe** 加载引导 **HIVE**，此时引导 **HIVE** 位于 **nk.bin** 解压之后的文件中。
3. **fileSYS.exe** 启动 **device.exe**，之后处于等待状态，等待 **device.exe** 将包含系统 **HIVE** 的文件系统和存储设备的驱动程序加载完毕。而这个文件系统和存储设备的驱动程序存在于引导 **HIVE** 中。
4. **device.exe** 加载上述所说的文件系统驱动程序和存储设备驱动程序，使之开始工作。之后 **device.exe** 处于等待状态。
5. **fileSYS.exe** 被唤醒，加载并且安装系统 **HIVE**。之后 **fileSYS.exe** 处于等待状态。
6. **nk.exe** 按照系统 **HIVE** 的信息开始执行初始化工作。其中包括加载驱动程序和启动一些应用程序。其中加载驱动程序一般由 **device.exe** 执行，而启动应用程序由 **fileSYS.exe** 执行。这时 **device.exe** 和 **fileSYS.exe** 已经被唤醒。

因为引导 **HIVE** 和系统 **HIVE** 肯定有重复的地方，所以可能出现重复加载了驱动程序或者重复启动了应用程序。为此，**CE** 允许在描述驱动程序的注册表信息中加入防止重复的标志，而应用程序可以采用事件对象来防止重复启动，如 **device.exe**。

下面讲述如何设置基于 **HIVE** 的注册表(假如保存系统 **HIVE** 的是 **FAT** 文件系统)：

1. 在 **PB** 中加入 "**Hive-based Registry**"，如果是 **Geode** 平台，再加入 **BSP_ENABLE_FSREGHIVE** 环境变量。
2. 打开 **platform.reg**，找到如下信息：
3. ; **HIVE BOOT SECTION**
4. [**HKEY_LOCAL_MACHINE\init\BootVars**]

5. "SYSTEMHIVE"="Documents and Settings\\system.hv"
6. "PROFILEDIR"="Documents and Settings"
7. "Start DevMgr"=dword:0
8. IF BSP_ENABLE_FSREGHIVE
9. "Start DevMgr"=dword:1
10. ENDIF
11. ; END HIVE BOOT SECTION

"SYSTEMHIVE"的值为系统 HIVE 文件的路径。"Start DevMgr"是一个布尔值，指示是否开始就执行设备管理器 **device.exe**，按照 CE 帮助文档的说法，只有想把系统 HIVE 存储在对象存储中才在此设置为 0，所以一般都要设置为 1。

- 12.如果是多用户，可以在上述的注册表位置下输入"DefaultUser"="", 指定默认的用户名。如果是单用户系统，可以不设置。
- 13.保证将包含系统 HIVE 的文件系统驱动程序的注册表信息和存储设备的驱动程序的注册表信息被包含在"; HIVE BOOT SECTION"和"; END HIVE BOOT SECTION"之间，在这两个语句之间的注册表数据全部属于引导 HIVE。假如我们将系统 HIVE 文件 **system.hv** 存放在硬盘上，并采用 FAT 文件系统。那么就要将
[HKEY_LOCAL_MACHINE\System\StorageManager\FATFS]和
[HKEY_LOCAL_MACHINE\System\StorageManager\Profiles\HDProfile]
移动到"; HIVE BOOT SECTION"下。
- 14.在"; HIVE BOOT SECTION"和"; END HIVE BOOT SECTION"之间的所有驱动程序的注册表信息中都加入下列一个标志：
15. "Flags"=dword:1000

这个标志是一个位掩码，它可以和其它已经存在的"Flags"或运算。值 1000 表示此驱动程序只加载一次，这样 **device.exe** 就不会把当前驱动程序加载两次了。

- 16.在包含系统 HIVE 的存储设备的驱动程序的注册表信息中，加入如下标志（假设是硬盘）：
17. [HKEY_LOCAL_MACHINE\System\StorageManager\Profiles\HDProfile]
18. "MountFlags"=dword:2

这个标志表示这个存储设备包含系统 HIVE 文件。

按照如上所述设置后的内核就能实现永久存储注册表数据了。对于保存注册表数据的执行动作在此必须阐述清楚：

正常情况下，CE 能够保证重要的注册表数据能够从内存刷到（Flush）永久存储器上。但是这并不能完全保证所有数据都能完整地保存而不丢失，所以要保证万无一失，应该主动地调用 **RegFlushKey** 函数强制将内存中的数据刷到永久存储器上。这个函数的参数只有一个，就是注册表分支。CE 还增加一个注册表项（如下所示），它的作用是每当函数 **RegCloseKey** 被调用时都自动调用 **RegFlushKey** 函数。

```
[HKEY_LOCAL_MACHINE\init\BootVars]
    "RegistryFlags"=dword:1
```

如果 CE 在启动过程中发现系统 **HIVE** 出现错误，它会自动删除文件并创建一个默认的系统 **HIVE** 文件，如果出现下面的注册表项，说明发生了这种事情。

```
[HKEY_LOCAL_MACHINE]
    "RegPersisted"=dword:1
```