Module 1-2: Overview of OS Build Steps and Phases

Contents

Overview 1
Directory Structure of Platform Builder 2
The Build Process 9
Modifying Configuration Files 12
Lab 1-2: Instrumenting the Build Process 19
Review 20
Instructor Notes

Presentation: 30 minutes
Lab: 45 minutes

This module introduces the build system and configuration files. It is important to explain the %ENVVAR% syntax as a command line environment variable expansion. There are many such variables and the labs directly reference them so no assumption of installation folders or drives is made. It’s also important as with V5.0 it is now possible to point the IDE at a different %WINCEROOT% other than the default one from installation. It is usually helpful to actually show the file structure via the Windows Explorer on the instructor machine after discussing the various environment variables and folder structure.

In this module, students will learn:

- Windows CE build system folder structure
- Build system is command line (IDE is a GUI layer on top of command line)
- SYSGEN – Creates filtered headers, libs and modules (EXEs, DLLs)
- Build – compiles BSP and Project code files using build.exe, DIRS and SOURCES files
- BuildRelease – copies files to the %_FLATRELEASEDIR%
- Makeimg – merges config files and generates final OS run-Time Image

Required materials

To teach this module, you need the Microsoft PowerPoint file Module 1-2.ppt.

Preparation tasks

To prepare for this module:

- Read all the material for this module.
- Complete the lab.
Overview

In this module, you will learn:

- Windows CE build system folder structure
- Build system is command line (IDE is a GUI layer on top of command line)
- SYSGEN – Creates filtered headers, libs and modules (EXEs, DLLs)
- Build – compiles BSP and Project code files using build.exe, DIRS and SOURCES files
- BuildRelease – copies files to the %_FLATRELEASEDIR%
- Makeimg – merges config files and generates final OS run-Time Image

Introduction

Objectives
Directory Structure of Platform Builder

- The Platform Directory
- The Public Directory
- The SDK Directory
- The Others Directory
- Private Directory
Environment Variables in the Build

- `%WINCEROOT%` - Root of the Windows CE build install
  - Typically C:\WINCE500
- `%PUBLICROOT%` - `%WINCEROOT%\PUBLIC`
- `%PLATROOT%` - `%WINCEROOT%\Platform`
- Many others
  - Use set in a command prompt build window to see a the ones currently set
The Platform Directory

- Contains Hardware Platform-specific BSP files
- Is located at the path %WINCEROOT%\Platform
- Contains subdirectories
  - Each subdirectory represents a Windows CE-specific platform adaptation
  - Each directory contains a dirs file and one or more batch files for unique BSP settings
- Place your Hardware Platform-specific files in this directory
The Public Directory

- Contains a platform-independent set of components and configurations
- Is located at the path %WINCEROOT%\Public
- Contains three types of subdirectories:
  - Module and component subdirectories- Common, Wceshellfe, Wceappsfe, IE, MSMQ
  - Reference configuration subdirectories- CEBASE
  - Custom configuration subdirectories
The SDK Directory

- Is located at the path %WINCEROOT%\SDK
- Contains tools to support Platform Builder
  - Processor compilers
  - Development tools
  - Miscellaneous utilities
The Others Directory

- Is located at the path %_WINCEROOT%\OTHERS
- Contains the following subdirectories:
  - ATL: Contains ATL headers, libraries and source code for debugging
  - DOTNET: Contains the .NET processor specific binaries and portable .NET Compact Framework files
  - MFC: Contains MFC Headers, Libraries and Source Code for debugging
  - SAMPLES: Contains the MFC and ATL sample applications
  - WCETK: Contains the Windows CE Test Kit (CETK) processor specific binaries
It is important to clarify that Shared Source is not “Open Source” or “FreeWare” commercial use requires a paid OS run time license. Any access or use of the Shared Source code is subject to the Microsoft Shared Source License Agreement available from the MS website and presented before installing the shared source code. The license is also installed to X:\Program Files\Windows CE Platform Builder\5.00\Source.rtf
The system generation stage filters the various header and configuration files based on the set of catalog items selected for the OS Design. It also links the various OS component libraries into the final DLL or EXE modules.

The Build stage compiles and links the BSP and other workspace projects you provide.

The BuildRelease stage copies files into the %_FLATRELEASEDIR% folder. This may be done with actual full copies or with hard links. Using hard links shortens the process but can have subtle problems if you try to edit files in the %_FLATRELEASEDIR%

Makeimg stage uses the contents of %_FLATRELEASEDIR% to generate the OS Run-Time image.
System Generation

Sysgen.bat is the command that executes the sysgen phase of the build. It processes all the header and configuration files to filter out functionality not supported by the OS Design. The sysgen process has the following basic internal steps:

1. Run cesysgen.bat to establish component and module dependencies
2. Filters header and configuration files
3. Links the component libraries into DLL or EXE modules
Building the Release Directory

- **BUILDREL.BAT**
  - Copies all platform files to the release directory
  - Copies all binaries to the release directory
  - Uses either full copy or Hard links depending on user settings
Modifying Configuration Files

The build system uses a number of configuration files that you need to edit or customize to meet your system requirements. This topic goes into the details of each of the various file types and their internal syntax.

It’s good practice to place comments at the beginning and end of any configuration files (BIB, REG, DAT and DB) as they are all merged together into one final file of each type that is used in the makeimg stage. For BIB files this should be bracketed in each section (MODULES, FILES) as they are merged by section. This allows you to look at the merged file and see a problem and easily identify where it came from.

Delivery Tip

It’s a good idea to show one of the final merged files to illustrate the value of bracketing the configuration files with comments.
The Configuration Files: .BIB Files

Binary Image Builder (BIB) files provide the list of modules and associated data files to include in an OS Run-Time image ROM. The Modules section contains modules that are located for eXecute In Place (XIP).

The syntax of this file is a simple single line of text the sections are delineated by the word MODULES or FILES on a single line of its own. Each entry in the sections has the following white space separated fields:

- Name of the file as it will appear to the OS
- Location of the file on the development workstation
- The memory section it should place the module or file into
- The file type flags for a complete list of the type flags see the online help at 
  ms-help://MS.WindowsCE.500/wceosdev5/html/wce50conMODULESSection.htm
  ms-help://MS.WindowsCE.500/wceosdev5/html/wce50conFILESSection.htm
The Configuration Files: .BIB Files

- **MEMORY section**
  - defines the platform memory information
  - Partitions the physical memory into Data memory and Program memory

<table>
<thead>
<tr>
<th>Name</th>
<th>Start address</th>
<th>Size(bytes)</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>NK</td>
<td>8C800000</td>
<td>00800000</td>
<td>RAMIMAGE</td>
</tr>
<tr>
<td>RAM</td>
<td>8C050000</td>
<td>007AFFFF</td>
<td>RAM</td>
</tr>
</tbody>
</table>

- **The CONFIG section**
  - Is not mandatory
  - Is used to set properties, such as compression or ROM size

The memory section specifies the memory regions used in creating the OS Run-Time image. RAMIMAGE is the actual OS executable section. The RAM region is the system dynamic RAM.

The CONFIG section contains a number of parameters for FLASH/ROM builds and some configuration flags for the OS build.
The Configuration Files: .REG Files

- Define default registry settings
  - COMMON.REG, IE.REG, MSMQ.REG, WCEAPPS.REG, WCESHELL.REG: define registry settings for Windows CE modules
  - PLATFORM.REG: platform-dependent registry settings, such as device drivers entries
  - PROJECT.REG: defines registry settings relative to your project

[HKEY_LOCAL_MACHINE\init]
“Launch10”=“shell.exe”
“Launch20”=“device.exe”
The Configuration Files: .DAT Files

- Define folder structures of your image
  - COMMON.DAT, IE.DAT, MSMQ.DAT, WCEAPPS.DAT, WCESHELL.DAT
  - PLATFORM.DAT
  - PROJECT.DAT
- Example:

  ```
  Root:–Directory("Program Files")
  Directory("\Program Files"):–Directory("My Projects")

  Root:–Directory("My Documents")
  Directory("\My Documents"):–File("MyFile.doc","\Windows\Myfile.doc")
  ```

FileSys.exe processes the DAT file entries at system boot to initialize the ram based file system. They are not used in systems not using the RAM file system.
The Configuration Files: .DB Files

- Define default image databases
  - COMMON.DB, WCEAPPS.DB, WCESHELL.DB
    These .DB files are the databases relative to the Common, Wceapps, and Wceshell modules
  - PLATFORM.DB
    The PLATFORM.db file contains the platform-specific default databases
  - PROJECT.DB
    The PROJECT.db file contains the project-specific default databases
Making an Image

- Merges release folder files to make the Windows CE image
- Process:
  - Merge configuration files:
    All .bib files into CE.BIB
    All .reg files into REGINIT.INI
    All .dat files into INITOBJ.DAT
    All .db files into INITDB.INI
  - Compress reginit.ini in a binary registry file (DEFaulT.DBF)
  - Replace resources in .exe and .dll for language adaptation
  - From ce.bib, combine binaries and files into a Windows CE image (NK.BIN)
Explaining Build Errors

- **Errors during the Sysgen**
  - Caused by missing files, missing configuration of the operating system features, and applications built during the Sysgen phase

- **Errors during the Module build phases**
  - Compilation errors or unresolved link errors

- **Errors during the building the release directory phase**
  - File copy errors

- **Errors during the making an image phase**
  - Romimage.exe failed in CE.BIB
  - Romimage.exe failed in reginit.ini
  - Warning: Image exceeds
Lab 1-2: Instrumenting the Build Process

Introduction

This lab adds instrumentation to the build for an OS Design by adding extra “code” lines to the BAT files used in the build that logs the use and parameters of each bat file. This allows you to see the exact call sequence and parameters used and it also helps in understanding all of the BAT files used in the build. For anyone used to the command line this will be fairly familiar. However to the IDE users this will likely be new and even foreign even though it was used under the hood before! The IDE is now a thin layer on top of the command line build and uses the same command line build system that you might use manually so there is no longer a difference between building form the command line and the IDE.
1. How is the IDE build different from the Command line build?
   Trick Question! It isn’t! In V5.0 the IDE uses the command line underneath without adding anything additional to the build.

2. Which part of the build system is responsible for linking OS components into modules?
   SYSGEN the last step of sysgen will link the components into modules.

3. What is the difference between the MODULES and FILES section of the BIB files?
   The MODULES section defaults to uncompressed so the modules can be located for XIP. The FILES section is compressed by default and never located for execute in place.

4. What is the name of the combined registry file created by the makeimg process?
   REGINIT.INI You’ll see syntax errors in your reg files appear as errors in REGINIT.INI with line numbers indicating the area of the file with an error.