

Windows Embedded CE

Build Your Picture Frame with Windows Embedded CE

VIGNESH DORAISWAMY
PRINCIPAL CONSULTANT
AXIOM ETC
VIGNESH@AXIOMETC.COM



Agenda

- Building an embedded device
 - Devices examples
 - Picture Frame
- Windows CE Review
 - OS layout
 - Memory architecture
 - Tools
- Demo
 - Building the Picture Frame OS
- Building an embedded application
 - Framework Choices
 - Demo!



Build a Windows Embedded CE Device

What does it take to get a Windows Embedded CE device to market?

Digital Picture Frame example

1 Select the Chipset

SV

- Often first part chosen
- MIPS, ARM based mostly
- X86: Intel, AMD, Via
- Renesas, STM only major SH4 SVs

Company examples

- Texas Instruments, NXP, ST Microelectronics, Samsung, and Freescale

2 Build the Core Hardware

Board

- In house
- SV
- SI
- IHV
- Design house
- Through distributors' packages

BSP

3 Get the Software License

Acquired through:

- Distributor
- Direct channel

Acquired by:

- OEM
- ODM

In rare cases:

- IDH
- IHV
- SV

4 Create the Experience

Application and UE

- In house
- Design house
- ISV
- SI
- SVs provide more and more application level software (to a certain extent)

5 Build and Manufacture

- ODM**
 - Mostly high-volume consumer devices
 - Tatung, Wistron, Foxconn
- OEM**
- IDH Sub-Contractors**

6 Brand and Sell

- OEM**
 - Builds and ships to end-user channel
- OEM: Not final buyer**

Windows Embedded Family

- Windows Embedded Compact
 - Portable media
 - Consumer Thin Client
 - Entertainment
 - Medical
 - Dedicated servers
- Windows Embedded Standard
 - Industrial Automation
 - Telematics
 - Robotics
 - Industrial Automation
 - Thin Client
- Windows Embedded for Point of Service
 - Personal Navigation
 - Point of Service
 - Books
 - Entertainment

What is Windows Embedded CE6.0 ?

Windows Embedded CE 6.0 is **NOT** Windows Mobile 6.x (based on CE 5.0)

Windows Embedded CE 6.0 is...

- 32-bit, real-time, multitasking OS
- Highly componentized (~700 components in CE 6.0)
 - Delivered as a granular set of components
 - Use CE 6.0 Platform Builder tools to configure image
- Scalable
 - Footprint scales with functionality selected
 - 350KB minimum - Kernel & FileSystem
 - 500KB - Adding Networking
 - 1MB - Webserver & Webcam driver
- Wide variety of CPU support
 - Runs on x86, ARM, MIPS and SH4

Introducing the CE 6.0 Kernel

- 2 GB of Virtual Memory per process
- 32K processes
 - 64K Global Handle Table
- Unified Kernel
 - Critical OS components moved into kernel space
- Improved system performance
- Increased security and robustness
- High degree of application compatibility

CE 6.0 System Architecture

User Mode

- Shell
- Services Manager
- UM Driver Manager
- User Mode Drivers
- Applications

OS DLLs (CoreDll, Winsock, CommCtrl, ...)

Kernel Mode

- Kernel.DLL
- FileSys.DLL
- GWES.DLL
- Device.DLL
- Kernel Drivers

Hardware

Bootloader

Kernel Drivers

Tools and Frameworks

Demo

Building a Picture Frame OS

How Does The Application Fit into the Picture?

USER: MyPictureFrame.EXE

O/S: GDI, Display Driver, NDIS

H/W: Display, NIC

Application Building

- Goal
 - How to build Apps for Windows Embedded CE
 - Information to help choose the right tools
- How we will achieve that
 - Tools
 - Win32
 - MFC
 - Managed

Win32 Pros and Cons

| Pros | Cons |
|---|---|
| Size Speed (of execution) Real-time development Drivers & Services | Speed of development Preventing & Debugging memory/handle leaks Complexity (no structure to APIs) |

Demo

Win32

Demo Highlights

- Flexible framework based on your project
- Looks the same as a desktop application
- Leverage your Win32 skills

MFC Pros and Cons

| Pros | Cons |
|---|--|
| Helper classes for String, Array, List... Cleaner Class structure Tools support for WM_Handlers Unicode 'hidden' by CString! | Thin wrapper over Win32, often need to drop to Win32 Easy as Win32 to leak memory/handles |

Managed

- .NET Compact Framework
 - MSIL, GC
 - C#, VB, currently no C++ (__gc)
 - ~4MB
 - 8 percent the size of the full .NET
 - 30 percent of the full .NET Framework class library

Managed Pros and Cons

| Pros | Cons |
|--|---|
| Processor independent (policy) Full .NET Common Language Runtime Garbage Collection Sandwich: Focus on the filling, not the bread P/Invoke to escape the box | Size of Framework No real-time development (typically work) Main focus is UI based applications JIT vs. Native Managed doesn't mean you can code like the desktop |

Demo

Managed Code

Summary

What you have learned...

- Build your own OS
- Tools fit for purpose
- Build Native and Managed Apps
- Deploy and Test

