Pro-face
for the best interface

Embedded Operating System:
The Right Choice for Your Application?

Shunte McMillian
Pro-face iPC Product Marketing Manager

Reasons to Choose an Embedded Operating System

An Embedded Operating System (OS) is designed to be more compact, efficient at resource usage, and reliable, while eliminating many functions that a non-embedded computer operating system provides. Unlike a desktop operating system, embedded operating systems do not load or execute applications. This means that the system can only run a single application at a time.

Choosing an embedded OS can have a big effect on the efficiency of your applications. The two operating systems offer different benefits that must be evaluated to determine the best fit in the industrial environment.

Understanding how the application will be used now and in the future is necessary for executing operations effectively.

Selecting Your Embedded Operating System (OS)

Microsoft Windows 7 and XP are commonly used in industrial PCs (iPC) for factory automation. The advantages of using these systems, such as multitasking and full capacity utilization, are advantageous for most applications; however, with the size and overhead of these operating systems, applications cannot always use the systems efficiently.

Microsoft offers embedded versions of these operating systems that help address the size and overhead concerns. In addition, the embedded operating system brings more stability and security to embedded devices. Windows-based applications can also run on Windows XP Embedded, which allows developers to easily write applications with the Windows API set.

Chart 1 - Features and Benefits of Windows 7 and Windows XP Embedded

Scalability	Overall size of the operating system can be condensed by removing unneeded features and services
Built-in networking and communication services	Allows use of TCP/IP, DHCP, WinSock, RPC, RRAS, FTP, etc.
Interoperability with existing PC and server hardware	Provides greater choices and flexibility when choosing your platform
Win32 API support	Provides consistent development environment
Windows services support	Allows greater manageability
C2-level security	Supports applications that demand secure environments
Systematic multiprocessing support	Provides one system that can be used for simple and very demanding applications
Reduces time to market	- Allows easy integration into your platform using powerful authoring tools - Requires less time developing and supporting proprietary OS code - Involves less time developing drivers, services, and applications, and getting them to work
Broad range of productive Windows-based development tools	- Many trained and experienced developers - Multitude of off-the-shelf hardware and device drivers - Large number of existing Win32 applications
Easy enterprise connectivity	- Microsoft BackOffice family applications - Allows easy integration of new opportunities with an existing IT infrastructure - Introduces and manages devices similar to other Windows-based systems Permits next generation devices to participate in enhanced management environments (examples: Microsoft Systems Management Server, HP OpenView, IBM Tivoli, CA Unicenter TNG, etc.)

Important Tools for the Transition

Embedded Operating System Component Tool Kit

Windows Embedded

Target Designer

Tool that provides a development environment that you use to create a bootable run-time image for a target device.

Component Designer

Development tool that enables you to define an application or device in a graphic development environment and save it to a disk.

Component Database Manager

Provides management functions for the component database and the repositories, which are used by both Component Designer and Target Designer tools.

Target Analyzer

Probe utility which creates a configuration that can be built into a Target Designer run-time image by analyzing the specific details about the target device and the Target Analyzer Importer.

Steps to develop an embedded operating system

The steps in the development process are as follows:

1. **Identify the hardware on your target device.**
2. **Choose the features and functionality required in your run-time image.**
3. **Identify the embedded system-specific features that need to be included in your target device.**
4. **Include custom components.**
5. **Build your run-time image.**
6. **Deploy your run-time image.**

Chart 1 shows some of the advantages of using these embedded operating systems in an application. Using these versions requires a solid understanding of the application and its future uses to ensure successful application efficiency on the operating system.

Below are examples of issues and concerns that may be associated with using an embedded OS.

Image Development

Before developing an embedded run-time image, consider the target application as well as the minimum computer requirements that will be used in the development system. Items such as additional I/O device network connections, programmable controllers, motion control systems, scanners, and many others can only be chosen by the OEM or end user of the device. For example, Pro-face can supply an industrial computer with a compact flash storage device and a reduced size operating system, but if the system has one of the previously mentioned hardware devices added, it may cause the system to not run properly. For this reason it is important that the OEM or end user be involved in developing the original run-time image.

Hardware

Before developing an embedded run-time image, consider the target application as well as the minimum computer requirements that will be used in the development system. Items such as additional I/O device network connections, programmable controllers, motion control systems, scanners, and many others can only be chosen by the OEM or end user of the device. For example, Pro-face can supply an industrial computer with a compact flash storage device and a reduced size operating system, but if the system has one of the previously mentioned hardware devices added, it may cause the system to not run properly. For this reason it is important that the OEM or end user be involved in developing the original run-time image.

Features and Functionality

Embedded operating systems are designed to do a specific task, rather than be a general-purpose computer for multiple tasks. Some have real-time performance constraints that must be met, for reasons such as safety and usability; others may have low or no performance requirements, allowing the system hardware to be simplified to reduce costs. An embedded system has fewer capabilities, as applications have been taken out of the operating system for more efficient use when building the run-time image. If an application that requires support is added, and was not included when building the run-time image, then the application might not run properly. This is another example of the importance of the OEM and end-user's involvement in the development of the original run-time image.

Embedded System-Specific Features

When using an embedded OS, it is important to pay attention to the size of the storage device. For example, storing a full operating system on a compact flash less than 16Gb for an iPC will cause your applications to run improperly. However, for an embedded OS, using a 2Gb, 4Gb, or 8Gb compact flash would be ideal. Though, when using reduced sized media with an embedded OS, you may not be able to

store data or utilize swap files. Pro-face offers industrial PCs with CF cards or SSD/HDD options, along with alternative storage devices, such as an SD card. The size of the media and the system's intended use are key factors in making the correct image.

Custom Components

It is important to install components in the developmental stage of the image. Not installing at this time may cause functions and applications not to work properly. Embedded operating systems are dedicated to specific tasks, and are not meant to be altered once the image has been developed. All utilities can be added directly to the run-time image by using Microsoft Target and Component Designers. With a non-embedded system, custom components can be added to the image after the developmental stage.

Build the Run-time Image

Building the image for an embedded operating system differs from building an image from source code. When using Target Designer, the image is generated by reassembling the individual components to create the operating system. With the use of Windows Embedded Studio, dependencies can be checked and resolved before building the run-time image. Tasks, such as assembling resources and generating directory structures, can all be accomplished before the image has been finalized. If the features or services are not functioning correctly for the OEM or end user, then more attempts to build the image will be required until the needs are addressed.

Deployment of the Run-time Image

After building the run-time image for an embedded operating system, it is ready to be deployed from the development system to the embedded device. There are a few methods you can employ to do this:

- Transfer the image using traditional methods such as disk, bootable CD-ROM, or bootable digital video disc (DVD).
- Swap a storage device (HDD, SSD) or persistent storage module (Rom, Flash, Disk).
- Transfer the image electronically over a communications line.

You can also use the deployment tools to transfer, install, and configure the run-time image into your target device. Your target device must have sufficient storage available to run the image. This storage may need to be initialized prior to receiving the image, which is done by a setup program.

Hardware Selection

Now that you have an understanding of the process to create an image, it is important to understand what hardware or target device will be used. Once the hardware platform is selected, all application software and auxiliary devices need to be determined.

First, the drive must be selected. Compact flash card and solid-state drives are more reliable than rotating media hard drives. Applications that will require levels of vibration and shock should use a compact flash card or solid state drive. HDD (hard disk drives) will have a higher capacity for storage, but will not function properly within a vibration/shock environment.

When deciding which storage device to use, it is important to have enough storage space for data. For example, an iPC with a 16Gb compact flash will fill more quickly than a 60 GB Solid State Drive. Also, when these drives reach capacity, the system and application may terminate unexpectedly. This is unacceptable in a manufacturing environment.

Swapping Files

The use of swap files is the most important item to consider when choosing between an embedded or non-embedded system. The standard non-embedded Windows 7 or XP operating systems use swap files that can significantly increase the amount of storage required. With an embedded system, you can disable the swap files if needed. But unlike hard drives, solid-state drives and compact flash cards have a limited amount of times you can "write" and "rewrite" onto the drive or card. With a hard drive, there are an unlimited number of "writes" to the drive.

It is also important to have an understanding of the image you will be receiving from your iPC provider. For example, Pro-face creates the image for the operating system and ships the product. But, if there are any other requirements needed for your operating system, it is important to know that it will be supported. If it is not supported, the hardware may not function properly and may need a new image.

Choose Wisely

Though there are many options, it is important to understand the benefits and drawbacks to embedded operating systems. To recap, the factors when considering an embedded operating system are:

- Hardware Selection
- Features and Functionality
- Custom Components
- Building and imaging the embedded operating system specifically to your needs

Choosing an embedded operating system will allow the applications for your industrial PC to run more smoothly and efficiently.

For more information, contact **Pro-face America**
profaceamerica.com | customercare@profaceamerica.com | 734-477-0600 | 800-289-9266