Minimizing the Windows 95 Footprint for Embedded Systems Applications

By Sean D. Liming, Engineering Manager, Annasoft Systems

The introduction of Windows 3.1 marked a paradigm shift from the simple, single-tasking, text-oriented environment provided by MS-DOS. With Windows 3.1 came a friendlier GUI, primitive multitasking, a standard driver interface, and WYSIWYG printing. Microsoft's latest operating system offering, Windows 95, builds on the baseline provided in Windows 3.1, paving the way for robust 32-bit applications and providing advanced features such as true multitasking, built-in driver support, extended file names, and integrated multimedia support.

Though Windows 95 was developed as a desktop operating system, its development capabilities and third party software support also make it an attractive platform for embedded systems. Many embedded developers may be reluctant to consider Windows 95 because of its large footprint. However, a closer look at Windows 95 reveals that its core facilities occupy just a fraction of the overall code. By stripping away facilities that are not required for the target system, designers can cut the size of the Windows 95 image in half, producing relatively lean configurations that are well suited to embedded environments that utilize Flash-based hard drives.

Flash Drives Provide Ideal Embedded Storage Medium

Many embedded systems designers prefer to execute their operating system and applications out of ROM. Unfortunately, while it is possible to fit the bulk of the Windows 95 files into an INT 13h ROM drive, Windows 95 requires a read/write disk to run. Even if the Windows 95 files were placed in a ROM drive, they would still need to be loaded from ROM and executed out of RAM. Many of the Windows 95 files (particularly files in the Root and Windows directories such as the Plug-and-Play configuration, registry, INI, and swapfile) must still reside on a read/write disk.

Many embedded systems applications can make do with a traditional hard drive with rotating media. However, solid state drives based on Flash memory provide a better solution for embedded systems applications that are constrained by rigid speed, size, power, and environment requirements.

Flash drives offer faster read rates than their rotating media counterparts. Fast loading, in turn, reduces power consumption, which is important for extending battery life in mobile applications. Flash-based drives are also extremely rugged, an advantage in harsh embedded environments.

Companies such as M-Systems offer Flash hard drives in a variety of convenient formats well suited to embedded applications, including PCMCIA, PC-104, and even ISA Bus. These drives come with file system software (such as M-Systems' TrueFFS) that enable the Flash hard drive to emulate the Windows 95 file system.

Evaluate Target System Memory Resources

Before deciding which Windows 95 facilities should be incorporated into the target system, programmers need to assess the target system's memory resources and determine how much hard drive space is available for storing Windows 95 files. Windows requires a minimum memory pool (RAM plus hard drive) of 12 Mbytes. The bulk of this memory is used to store a paging or swap file (WIN386.SWP) that Windows 95 needs to support virtual memory. The portion of this file that must be stored on the hard disk is 12 Mbytes minus the amount of RAM that is provided in the system. For example, if the system is equipped with 8 Mbytes of DRAM, then the hard disk must provide 4 Mbytes of storage for the swap file.

To determine the amount of Flash drive space that is available for storing Windows files, designers need to first calculate the size of the Flash-based swap file, the application, and the accompanying data files. The available disk space can then be calculated using the simple formula:

> Available disk space for files = total disk space - (Flash swap file size + application size + data size).

Consider, for example, a system equipped with 4 Mbytes of RAM and a 20-Mbyte hard disk, in which the application and data files require 2 Mbytes of storage. The size of the swap file is:

> Swapfile = 12 Mbytes (total memory pool) - 8 Mbytes (DRAM) = 4 Mbytes.

And the amount of space available for storing Windows files is:

> 20 Mbytes (disk size) - 4 Mbytes (disk space for swap file) - 2 Mbytes (application code plus data space) = 14 Mbytes.

Integrating Flash Drive with ROM-Based Components

When designing Flash-based Windows 95 applications, designers need to be aware of the components that are stored in ROM, and where to they should be located. The system BIOS generally requires 64K of memory and resides at memory location F0000h. BIOSs that support functionalities such as Advance Power Management, Extended IDE, and Plug-and-Play, often require 128K of memory and must be loaded at memory location E0000h.

If the system needs to support a VGA monitor, then it will need a ROM-based Video BIOS Extension. This extension typically requires 32K of memory and is located at memory location C0000h. If a bootable PCMCIA solid state drive is used instead of a hard disk, then a 32K BIOS Extension (located somewhere between C8000h and EFFFFh) will also be needed. A bootable PCMCIA solid state drive will most likely require a memory page window (usually 32K), which facilitates access to the drive. This window often resides in the same location as the BIOS Extension, as is the case with the Flash disks supplied by M-Systems.

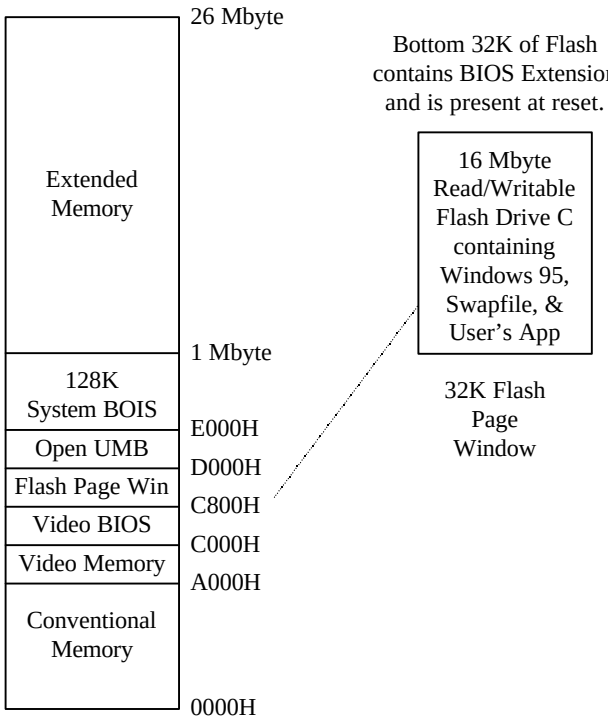

Figure 1: ROM Configuration for a system equipped with Flash-based hard disk

Creating a Minimal Windows 95 Image

Once designers have determined how much Flash drive space they have available for storing Windows files, they must analyze their application to determine which Windows 95 services are required and therefore, which files are necessary for the application to work correctly. A simple data collection application, for example, can probably do without the OLE services required by multimedia application. Similarly, an inexpensive palmtop computer can probably get by without a full set of TrueType fonts. Correspondingly, target systems that are designed to run only Windows 95 applications can most likely survive without the Windows files that are used to support MS-DOS and Windows 3.1.

Having identified the Windows functions they will need to support their applications, designers are then ready to begin the process of shrinking the Windows 95 image. The basic

strategy is to remove all the files determined unnecessary until the Windows image fits into the available disk space, while constantly testing the system to be sure the application will run without the files that have been removed.

Regardless of the application, a few core components must be maintained. Aside from these few files however, shell components such as the Explorer, system utilities such as the Control Panel, and end-user applications such as Microsoft can be cut. Designers may also be able to omit space-hungry Dynamic Link Libraries (DLLs) for functions such as Object Linking & Embedding (OLE), which are used by a broad range of Windows applications that aren't likely to appear in the target system.

After deciding which Windows 95 services their application will need, designers should configure a test system by performing a custom Windows 95 installation wherein the desired components are selected for installation and the remainder are deselected. Once this installation has been completed, designers should install and test their application. After verifying proper operation of the application, designers can begin to remove unwanted files from the installed Windows 95 image. Periodically rebooting the system to verify proper operation is highly recommended. As a safety precaution, designers should create a separate dump directory (not in the system path) to store deleted files in case these files prove critical to the application and need to be reloaded.

After the desired files have been removed, the SYSTEM.INI, SETUP.INI and Registry files need to be edited in order to remove all references and connections to the files that have been deleted. Once this is completed, designers are ready to perform a final test and install the image (Windows 95 files plus application files) on the target system.

Installation of the Minimized Windows 95 Image

Mass distribution of the Windows 95 files and application to target systems can be performed in one of three ways. One of these installation approaches involves the use of a hard disk duplicator. If the image will be downloaded over a network, the best choice is likely to be the Windows OEM Preinstallation Kit. Lastly, there is the DOS-based Interlink and Interserve method that involves downloading the image to the target via a serial or parallel port.

Removing Windows 95 Components by Group

Designers who want to produce an embedded version of Windows 95 will need to purchase an OEM Preinstallation Kit (OPK). The OPK contains most of the files found in the desktop version of Windows 95, including multimedia extensions, accessories, help files, and so on. Files that are not likely to be used in an embedded system, such as Tutorials, Demos, MSN, and Internet support, are not shipped with the EDK.

Windows 95 services can be deleted by one group or one file at a time. The most efficient approach is to eliminate whole groups wherever possible, and then individual files as necessary. The following list of files, accompanied by their locations, provides an overview of the basic software components in Windows 95.

BASE Files:

COMMAND.COM	;Root	WIN.INI	;Windows
IO.SYS	;Root	WIN.COM	;Windows
MSDOS.SYS	;Root	WIN386.SWP	;Windows
SUHDLOG.DAT	;Root	ADVAPI32.DLL	;System
IOS.LOG	;Windows	COMM.DRV	;System
HIMEM.SYS	;Windows	DDEML.DLL	;System
IFSHLP.SYS	;Windows	GDI.EXE	;System
KEYBOARD.DRV	;Windows	GDI32.DLL	;System
MMSOUND.DRV	;Windows	KERNEL32.EXE	;System
MOUSE.DRV	;Windows	SHELL.DLL	;System
SETVER.EXE	;Windows	SHELL32.DLL	;System
SYSTEM.DAT	;Windows	USER.EXE	;System
SYSTEM.DA0	;Windows	USER32.DLL	;System
SYSTEM.INI	;Windows	VGA.DRV	;System
TTFCHACHE	;Windows	VGAFULL.3GR	;System
USER.DAT	;Windows	VMM32.VXD	;System
USER.DA0	;Windows		

MS-DOS Drivers

Windows 95 includes several MS-DOS drivers and TSR (Terminate Stay Resident) drivers that may be omitted. For example, HIMEM.SYS, IFSHLP.SYS, and SETVER.EXE are automatically loaded by Windows 95 and therefore do not need to be listed in CONFIG.SYS. EMM386.EXE is the MS-DOS expanded memory manager included with Windows 95. If the target system does not support expanded memory for EMS-compliant MS-DOS applications and does not require that MS-DOS TSRs and drivers be loaded "high" in upper memory blocks, then EMM386 can be removed. After doing so, designers should ensure that EMM386.EXE is not loaded in CONFIG.SYS. Similarly, if the target system does not require a RAM disk, then designers can remove the RAMDRIVE.SYS driver. Again, after removing this driver, designers should ensure that RAMDRIVE.SYS is not loaded in CONFIG.SYS.

Windows 95 also includes an MS-DOS TSR known as SMARTDRV.EXE that provides read and write disk caching. If the target system has limited RAM for Windows applications, designers may wish to eliminate disk caching by removing this driver. After removing it, designers will need to eliminate its reference in the AUTOEXEC.BAT batch file.

Removing the Shell

Windows 95 comes with two shells, EXPLORER and PROGMAN. It is common practice for embedded systems developers to replace the shell with their own application. To do so, the Shell=EXPLORER line in SYSTEM.INI must be replaced with the following information: Shell=NameOfYourApplication. This ensures that when the system boots up, it goes directly to the new shell, and that upon exit, the system shuts down. EXPLORER.EXE can be removed if designers plan to use their own shell.

Program Manager, File Manager, and Task Manager are the main applications used to run programs in Windows 3.1. these applications are included in Windows 95, but they can be removed. To make Windows 95 look and feel like its predecessor, designers can replace the Shell=EXPLORER parameter in SYSTEM.INI with Shell=PROGMAN. They must also change the BootGUI=parameter in MSDOS.SYS (only a text file) to zero. Finally, the LOGOS.SYS file should be removed. This will produce the MS-DOS prompt when the system exits Windows 95. Selecting this option enables users to switch between Windows and MS-DOS.

The Windows 3.1 Program Manager, File Manager, Task Manager and all Windows 3.1 applications require the SHELL.DLL module. Designers can remove this module if the target system does not include any of these files. To remove the Program Manager, File Manger, Task Manager, and other shell components, designers should delete the following files:

ACCESSOR.GRP
MAIN.GRP
PROGMAN.EXE
PROGMAN.INI
SHELL.DLL
STARTUP.GRP
TASKMAN.EXE
WINFILE.EXE

To replace the shell application, the Shell=PROGMAN.EXE setting in the [boot] section needs to be changed to reflect the name of the application that will be run at startup. This setting is required. The replacement shell application can be located on disk or in ROM.

Help

Most Windows 95 features and accessories include Help information. Help information is stored in files with the .HLP extension and normally use the same filename as the application they are associated with. Any or all of these files may be removed. However, removing these files will not remove the Help menu in the application. Moreover, the F1 key and Help menu will still run the Windows Help system (WINHELP.EXE) if it is present in ROM or on the hard drive. An error message will be displayed if a requested Help file cannot be found. To remove all Help support, designers can simply delete the *.HLP and WINHELP EXE files.

MS-DOS Support

Windows 95 supports MS-DOS-based applications and commands. If the target system does not need to support MS-DOS, the files used to support MS-DOS applications within Windows 95 can be deleted. To remove support for MS-DOS-based applications, the following files can be deleeted.

*.PIF
*.FON
VGAFULL.3GR ;From the System directory
W1NOA386.MOD ;From the System directory
DELTREE /Y COMMAND ;From Windows directory

Designers will also need to edit the SYSTEM.INI file to remove the value from the 386grabber=setting.

Multimedia Extensions

Windows 95 includes many extensions that support multimedia applications. If the target system will not be running such applications (including those installed by the end user), multimedia extensions can be removed by deleting the following files.

```
MMCI.DLL
MMDEVLDR.VXD
MMFMIG32.DLL
MMSYS.CPL
MMSYSTEM.DLL        ;Also remove it from SYSTEM.INI
MMTASK.TSK
MIDI*.*
MCI*.*
TIMER.DRV
MPLAYER.*
SOUND.DRV
```

Once these files have been deleted, a few changes must be made to the SYSTEM.INI file. First, MMSYSTEM.DLL must be removed from the drivers=setting. Then, they'll need to change the WaveAudio=mciwave.drv, Sequencers=mciseq.drv, and CDAudio=mcicda.drv settings in the [mci] section. Finally, the timer=timer.drv and midimapper=midimap.drv settings in the [drivers] section must be removed.

Accessories, Screen Savers and Fonts

Accessories are small application programs included with Windows 95. Accessories can be removed by simply deleting the corresponding executable and Help files. Windows screen savers have the file extension .SCR, subsequently, to remove screen savers, delete files with the .SCR extensions. Plotter fonts can be removed by deleting the *.SCPlotter Fonts files in the WINDOWS\SYSTEM directory. Plotter fonts such as Roman and Script can be removed by deleting the ROMAN.FON and SCRIPT.FON files.

Eliminating or minimizing the fonts (system fonts, bitmaps application fonts, and TrueType fonts) used in the target system can yield significant memory savings. Bold and italicized fonts, for example, require separate storage for each character. Whenever possible designers should make do with one or the other. TrueType fonts can be eliminated by deleting the .TTF file associated with that font. All of the TrueType fonts can be removed at once by deleting the FONTS directory.

Conclusion

Producing a slimmed-down version of Windows 95 for a Flash-based embedded system is a straightforward process. For most embedded applications, paring unnecessary files should enable designers to produce a Windows 95 image that, together with a swapfile and Visual Basic application, can fit comfortably in a system equipped with a 20-Mbyte Flash card and

4 Mbytes of RAM. The Microsoft Windows 95 Resource Kit, available separately from Microsoft, provides a comprehensive explanation of Windows 95 features and files, and is indispensable for producing lean Windows 95 configurations.