Getting Started

Microsoft® C/C++

Microsoft C/C++

Version 7.0

Getting Started

For MS-DOS® and Windows™ Operating Systems

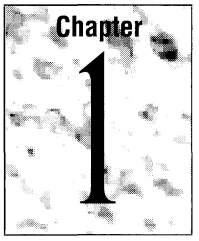
Microsoft Corporation

Printed in the United States of America.

Document No. LN24777-0192
10 9 8 7 6 5 4 3 2 1

Contents

Microsoft C/C++ Overview



This chapter offers an overview of Microsoft® C/C++. It includes:

- System requirements
- Package contents
- Components of Microsoft C/C++
- New features of this product
- Where to find information in other books provided with Microsoft C/C++

System Requirements

Microsoft C/C++ requires the following configuration:

- An IBM Personal Computer, or 100 percent compatible, running MS-DOS version 3.3 or later.
- An 80386 or higher processor.
- Four megabytes of available memory (RAM).
- One hard-disk drive with a minimum of 8 megabytes of free space. Depending on the options you want, you may need up to 27 megabytes of disk space. Setup will ask what components you want installed and then check to see if your system has enough disk space to install all the components you selected.
- One 1.2 megabyte, 5.25-inch floppy disk drive, or one 720K, 3.5-inch floppy disk drive.
- Microsoft Windows™ graphical environment version 3.*x* or another DPMI server.

Package Contents

Check your Microsoft C/C++ package to see if everything is there. If any pieces are missing, contact the retailer from whom you purchased the product. In the package, you should find the following items:

- *Registration card.* There are many advantages to being a registered owner of Microsoft C/C++, including notification of future software releases and easy access to customer assistance. Please take the time to fill out and mail the registration card now. (If you have upgraded from an earlier version of Microsoft C, a registration card is not included with the update.)
- *Disks.* The distribution disk labeled "Setup" contains a file named PACKING.TXT that lists the location and description of all disk files in the Microsoft C/C++ package. Most files on the disks are compacted. The Setup program uncompresses files as they are installed. Microsoft C/C++ is distributed on 5.25-inch high-density disks or 3.5-inch disks. If you need 360K disks to install the compiler, please send in the media order card contained in the C/C++ package, or call Microsoft Customer Service (1-800-426-9400).
- *Books.* The following books should be in your package:
  - *Getting Started*
  - *Environment and Tools*
  - *Programming Techniques*
  - *Run-Time Library Reference*
  - *C Language Reference*
  - *C++ Tutorial*
  - *C++ Language Reference*
  - *Class Libraries Reference*
  - *Class Libraries User's Guide*
  - *Source Profiler User's Guide*
  - *Comprehensive Index and Errors Reference*
- *Product Assistance Request.* If you need to contact Microsoft Product Support Services, be sure to fill out the questionnaire located in the *Run-Time Library Reference* before calling.
- *Documentation Feedback Card.* To help Microsoft improve its documentation, a postage-paid survey mailer is included in the *Run-Time Library Reference.* Please take the time to fill out the card with comments or suggestions.

Product Components

Microsoft C/C++ has all the components you need for developing C and C++, MS-DOS and Windows applications. This package includes:

- Microsoft Foundation Class Library
- Microsoft iostream Class Library
- C and C++ compilers with run-time libraries and graphics support
- QuickWin libraries
- Tools needed for Windows development
- P-code interpreter for executing p-code
- Programmer's WorkBench (PWB) with an editor, all utilities (LINK, LIB, IMPLIB, NMAKE, and HELPMAKE), other language compilers, and access to the CodeView® debugger and Help
- CodeView debugger
- Source Profiler
- MOVE, a set of libraries to allow MS-DOS programs to expand code and data space
- Online Help for the compiler, languages, utilities, Windows 3.*x* API, errors, and Microsoft Foundation classes
- Sample code
- Readme documentation for late-breaking information

New Features

If you have used an earlier version of Microsoft C, you will find many new features in version 7.0:

- Support for C++ programming
- Precompiled header files for faster compilation
- Support for p-code to reduce program size
- Remote debugging support

- Support for inline functions that allow the compiler to automatically substitute function code for function calls when appropriate for optimization
- Function allocation using based addressing
- Support for Super VGA screen modes in GRAPHICS.LIB
- Implementation of wide characters for international support
- Adjustable compiler warning levels and message display control

For information on compiling Microsoft C version 6.0 programs with Microsoft C version 7.0, see Appendix C, "Differences Between C Versions 6.0 and 7.0," in the *C Language Reference*.

Guide to Documentation

The following tables show where to get information in the Microsoft C/C++ documentation set. Most of this information is also available in the Microsoft Advisor Help system, which can be accessed from PWB, CodeView, or QuickHelp.

General Information

If you want to:	See:
Edit and debug programs	*Environment and Tools*
Use command-line utilities	*Environment and Tools*
Locate specific information	*Comprehensive Index and Errors Reference*
Look up an error message	*Comprehensive Index and Errors Reference*
Get answers to commonly asked questions	ANSWERS.TXT (on-disk troubleshooting guide)
Configure and optimize your system	*Getting Started*
Get Help for languages and tools	Microsoft Advisor Help system
Learn about using Help	*Environment and Tools*

C++ Programming

If you want to:	See:
Learn C++	*C++ Tutorial*
Use Microsoft Class Libraries	*Class Libraries User's Guide* *Class Libraries Reference*
Program with C++ and Windows	*Class Libraries User's Guide*
Understand C++ syntax and usage	*C++ Language Reference*
Compile code written for other C++ compilers	*C++ Language Reference*
Manage memory	*Programming Techniques*

C Programming

If you want to:	See:
Understand C syntax and usage	*C Language Reference*
Use the C run-time library	*C Run-Time Library Reference*
Optimize programs	*Programming Techniques*
Manage memory	*Programming Techniques*
Compare Microsoft C with C 6.0 or ANSI C	*C Language Reference*
Write portable programs	*Programming Techniques*

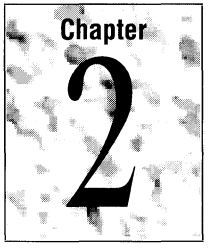
Windows Programming

If you want to:	See:
Program with Windows and C++	*Class Libraries User's Guide*
Get Help on Windows tools	Help files in the BIN subdirectory

Special Topics

If you want to:	See:
Use p-code	*Programming Techniques*
Use precompiled headers	*Programming Techniques*
Profile your program	*Source Profiler User's Guide*
Use the graphics libraries	*Programming Techniques*

Installing Microsoft C/C++



The Setup program provided by Microsoft C/C++ performs all tasks necessary for installing the C/C++ components. You can install Microsoft C/C++ under the Windows environment or under MS-DOS. This chapter gives you:

- Quick start information for running Setup.
- An overview of the installation process.
- An explanation on using the file viewer in Setup to modify your system files.

For information on configuring your system once you have installed MS C/C++, see Chapter 3 of this manual. For information on installing the Source Profiler, see the *Source Profiler User's Guide*.

Setup Quick Start

To run Setup, place Disk 1 in Drive A.

If:	Do This:	Result:
Windows is loaded	Select Run from the Windows File menu; type A:\SETUP at command line	Setup installs C/C++ for Windows and MS-DOS
Windows is installed but not loaded	Type A:\SETUP from command line	Windows loads and Setup installs C/C++ for Windows and MS-DOS
Windows is not installed	Type A:\SETUP from command line	Setup installs C/C++ for MS-DOS only
Windows is installed but you want to run the MS-DOS only Setup	Type A:\CSETUP from command line	Setup installs C/C++ or allows single-file copy

Note Extensive Help information is available from most Setup screens by clicking the Help button or pressing F1.

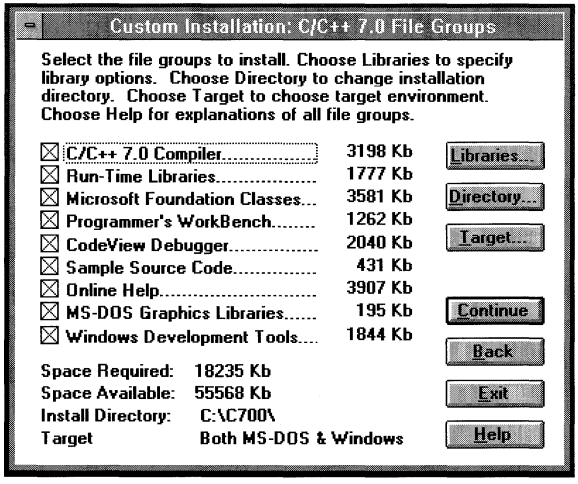
Installation Overview: Windows or Windows and MS-DOS

Setup checks system information and then asks you to confirm the following:

- Environment: MS-DOS version, Windows version, and Windows directory
- Installation directory: C:\C700 (default)
- Installation choice: Default, Custom, or Build Additional Libraries

If you select Custom Installation, Setup displays the following dialog box:

Choose file groups.



Click buttons to change libraries, installation directory, or target environment.

Click the Libraries button to select which C run-time libraries are to be built and installed. You can choose from among the following options:

Click buttons to select library options.

- Memory Models: Small/Tiny, Compact, Medium, Large/Huge
- Floating-point Math Support: Emulation, 80x87 Chip, Alternate
- Windows Targets: Windows .EXE, Windows .DLL, QuickWin .EXE files

Additional libraries can be installed later by running Setup again.

The remaining Setup screens allow you to:

- Change any selected options before copying begins.
- Review the README.TXT file while file copying takes place.
- Choose one of the following to modify system files:
  - Let Setup update your system files for you.
  - View and edit the changes now.
  - Save the changes to edit later.
- Reboot or exit Windows to enable the new system configuration

Installing C/C++ for Windows and MS-DOS

If you have Windows installed on your computer, you can run Setup from either the command-line prompt if Windows is not currently loaded or the Run dialog box, which is accessed from the File menu if Windows is currently loaded.

You can let Setup install MS C/C++ using the defaults, or you can view installation options by selecting Custom Installation.

Setup Using Defaults
: If you choose to install MS C/C++ using the Setup defaults, you will not be prompted for your installation choices. To find the list of components that are installed by default, press F1 or click the Help button from the Setup screen that prompts you to choose between Defaults or Custom Installation.

Custom Installation
: If you select Custom Installation, the Custom Installation menu shown on page 8 is displayed. From the buttons on this screen, you can also change the target (MS-DOS, Windows, or MS-DOS and Windows) or the default directory if you wish to do so. The Libraries button allows you to select the run-time libraries to be installed.

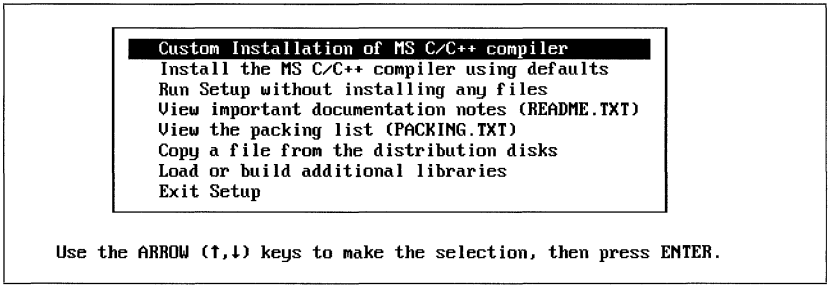
You have three options for how Setup modifies your AUTOEXEC.BAT, CONFIG.SYS, and SYSTEM.INI files:

- Let Setup modify your system files for you.
  - Setup saves current copies in same directory with .OLD as the file extension.
  - Setup automatically updates all three files.
- View and edit the changes now.
  - Setup saves current copies in same directory with .OLD as the file extension.
  - Setup automatically updates all three files.
  - Setup displays a split-screen viewer that allows you to edit the updated versions of CONFIG.SYS and AUOTEXEC.BAT. Clicking the Cancel Edits button cancels edits you have made but keeps the updates Setup made automatically.
- Save the changes to edit later.
  - Setup does not modify your system files.
  - Setup copies recommended updates that are named CONFIG.C70, AUTOEXEC.C70, and SYSTEM.C70 to your C:\C700\INIT subdirectory.

Installation Overview: MS-DOS Only

The MS-DOS Setup allows you to do any of these tasks:

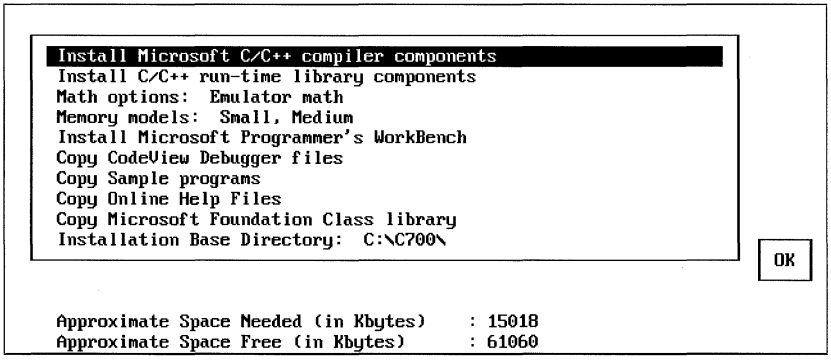
For Help on running the Setup program, press F1.



Setup then prompts for the correct destination drive (if more than one is available). If you choose installation using the defaults, Setup asks you to confirm these default settings:

To change a default or to access more information on each menu choice, use ARROW keys to make the selection, then press ENTER.

To accept choices, TAB to OK and press ENTER.



Next, Setup:

- Checks your TMP environment variable.
- Checks disk space requirements.
- Copies files and builds libraries.

You can then choose one of the following to modify system files:

- Let Setup update your system files for you.
- View and edit the changes now.
- Save the changes to edit later.

Setup reports any errors before terminating.

Installing C/C++ for MS-DOS Only

When you execute the Setup program, Setup checks for the presence of Microsoft Windows on your system. If you have not installed Microsoft Windows, the Setup program assumes you want to install MS C/C++ for MS-DOS only. Windows libraries and development tools are not installed.

Note You must use the Windows Setup if you want to install Windows libraries and tools. To do this, exit the MS-DOS only Setup program, install Windows, and Run Setup.

You can let Setup install MS C/C++ using the defaults or view installation options by selecting Custom Installation.

Setup Using Defaults
: If you choose `Install the MS C/C++ compiler using defaults` from the Main menu, Setup displays the list of default installation options and asks you to confirm your choices. The information screen on each installation option explains that component and allows you to make a different choice.

Custom Installation
: If you select `Custom Installation of MS C/C++ compiler`, Setup prompts you for your preferences for each installation option.

You have three options for how Setup modifies your AUTOEXEC.BAT, and CONFIG.SYS files:

- Let Setup modify your system files for you.
  - Setup saves current copies in same directory with .OLD as the file extension.
  - Setup automatically updates both files.
- View and edit the changes now.
  - Setup saves current copies in same directory with .OLD as the file extension.
  - Setup automatically updates files.

    Setup displays a split-screen viewer that allows you to edit the updated versions of CONFIG.SYS and AUTOEXEC.BAT. The viewer has two modes. By default, the status bars on both windows are highlighted. In this mode, both screens scroll at the same time.

    Press F3 to toggle to the other mode. In this mode, the screens scroll independently. You can move between files using TAB or F6 and scroll using PgUp and PgDn.

After viewing both versions of the files, you can either:

- Accept the changes by pressing ENTER.

  -or-

- Cancel the changes by pressing ESC. If you press ESC, no edits are saved. Setup returns to the screen that asks your preference for how to update system files.

- Save the changes to edit later.
  - Setup does not modify your system files.
  - Setup copies recommended updates that are named CONFIG.C70, AUTOEXEC.C70, and SYSTEM.C70 to your C:\C700\INIT subdirectory. Depending on your system configuration, some of the settings in these files are necessary for C/C++ to run on your system.

Getting More Information

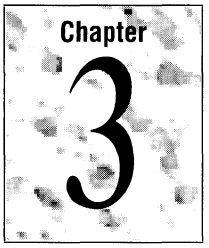
From the Windows Setup program, you can click the Help button or press F1 on any screen to access more information. Setup provides information on topics such as installation options, what to do once Setup is complete, and how to restart your system to enable a new system configuration.

From the MS-DOS Setup program, you can access more information on installation options and C/C++ components by moving the highlighting to the options and pressing ENTER.

For information on last-minute bug fixes, specialized information on particular components, or commonly asked questions, see the Readme files or the ANSWERS.TXT file.

If you have checked these sources as well as information found elsewhere in the documentation set, and you need to contact Microsoft Product Support Services, see the information on the Product Assistance Request form located in the back of the *Run-Time Library Reference* manual.

Configuring Your System



This chapter describes memory requirements for Microsoft C/C++ components. It also explains the reasons for Setup's recommended modifications to your system files (CONFIG.SYS, AUTOEXEC.BAT, and SYSTEM.INI) for Windows-environment or MS-DOS programming.

This chapter also provides information on conventional memory, extended memory, expanded memory, and memory managers. This will help you customize the available memory on your system to:

- Make more memory available for Microsoft C/C++ and other programs.
- Improve system speed as much as possible, while retaining enough memory to run your programs.

Balancing these two choices may involve experimenting with the described techniques until you find the right optimization for your system.

Note MS-DOS version 5.0 or later provides many new features that make memory configuration easier. Many of the recommendations made in this chapter require your system to have these new MS-DOS features. If you have not upgraded to MS-DOS version 5.0 or later, you may want to do so before configuring your system files for Microsoft C/C++.

Understanding System Configuration Terminology

This section defines terms you need to know to understand the configuration information in this chapter. See Figure 3.1 at the end of this section for a visual description of these memory areas.

Conventional or Real Memory
: In a computer using an Intel-compatible processor, the first 640K of a computer's memory. All MS-DOS systems have conventional memory. Programs can use conventional memory without special instructions.

Extended Memory (XMS)
: Memory above the first 1 MB of memory on systems with 80286 or higher processors. Many 80286 and 80386 computers come with 384K or more of extended memory. Extended memory requires an extended memory manager, such as HIMEM.SYS, to prevent different programs from using the same area of extended memory at the same time. Extended memory is required for Microsoft C/C++.

High-Memory Area (HMA)
: The first 64K of extended memory. On systems with MS-DOS version 5.0 or later, MS-DOS can be loaded into this high-memory area of extended memory, saving about 50K of conventional memory.

Expanded Memory (EMS)
: An area of memory accessible to programs that can access memory above 640K. Expanded memory is divided into 16K segments called "pages." When a program requests information from expanded memory, the expanded memory manager "maps" or copies the appropriate page to an area called a "page frame" in the upper-memory area. Since an expanded memory manager allows programs access to a limited amount of information at one time, expanded memory can be slower for programs to use than extended memory. Expanded memory requires special drivers such as EMM386.EXE. EMM386.EXE can also use extended memory to emulate expanded memory on 80386 and 80486 systems.

Upper-Memory Area or High DOS Memory
: The 384K of memory, called the "upper-memory area," that is above the 640K of conventional memory in most systems. Parts of this area not used by your system are called "upper-memory blocks" (UMB). If you have a system with an 80386 or 80486 processor and extended memory, MS-DOS can load itself into this upper-memory area, freeing conventional memory for programs. MS-DOS (version 5.0 or later only) has commands that enable you to store certain device drivers and programs in the upper-memory area. This memory cannot be accessed by user programs.

Device Driver
: A program that MS-DOS uses to control devices such as the keyboard, mouse, monitor, disk drives, or physical memory. Memory managers are device drivers. Device drivers are loaded into memory by statements in your CONFIG.SYS file.

Memory Manager
: A program that provides access to a particular type of memory. For programs to use extended memory, expanded memory, or the upper-memory area, your system must have a memory manager. Microsoft C/C++ provides two installable memory managers—HIMEM.SYS and EMM386.EXE. Although

memory managers take up some space in conventional memory, they make up for it by providing access to more extended memory, expanded memory, and upper memory.

HIMEM.SYS
: An installable memory manager that provides access to extended memory. HIMEM.SYS is required for Microsoft C/C++.

EMM386.EXE
: A device driver provided by Microsoft C/C++ to control expanded memory and provide access to the upper-memory area. The EMM386.EXE memory manager can also use extended memory to emulate expanded memory (see page 24).

SMARTDRV.EXE
: A device driver provided by Microsoft C/C++ for systems running MS-DOS (version 4.*x* and later) that enables faster disk access. SMARTDRV.EXE creates a disk cache in extended or expanded memory. SMARTDRV.EXE replaces the earlier version of this program, SMARTDRV.SYS.

Disk Cache
: An area in extended or expanded memory that SMARTDRV.EXE uses to store information it reads from the hard disk. This speeds up disk access because the next information the application requests may already be available in memory.

RAMDRIVE.SYS
: A device driver provided by Microsoft C/C++ for systems running MS-DOS (version 4.*x* and later) that reduces disk-access time. RAMDRIVE.SYS creates a virtual disk drive in RAM to emulate a physical disk drive.

DOS Protected Mode Interface (DPMI)
: Published specification for handling MS-DOS calls in protected-mode programs. Microsoft Windows 3.*x* provides DPMI services and is therefore called a "DPMI server." Microsoft C/C++ requires a DPMI server.

MS-DOS Extensions to the DPMI
: Provide additional functionality not required by the DPMI specification. Microsoft C/C++ requires a DPMI server that implements the MS-DOS extensions. Windows 3.*x* and Qualitas 386-Max version 6.*x* meet this requirement.

Virtual Control Program Interface (VCPI)
: Defines how multiple programs can run in protected mode on MS-DOS.

DOS-Extended Programs
Programs that have a protected-mode DOS extender bound into the executable file. This allows the program to use extended memory and to use real-mode interrupt services in protected mode.

WX SERVER
WXSVR.EXE works with WX.EXE, a real-mode MS-DOS program, to allow Windows programs to be invoked from within a Windows DOS session.

Figure 3.1 shows the location of memory areas discussed in the preceding list.

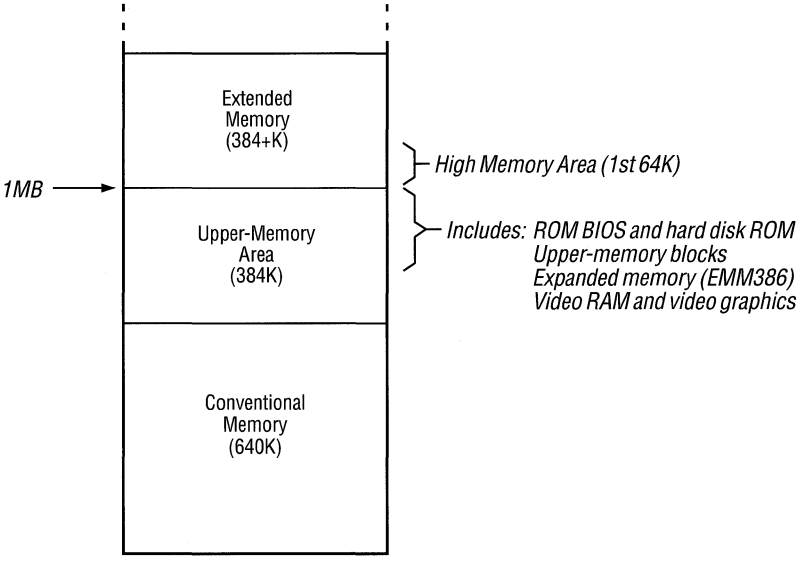


Figure 3.1 Memory Locations

Choosing a Development Environment

The Microsoft C/C++ compiler components require the presence of a DOS Protected Mode Interface (DPMI) server that supports version 0.9 or later of the DPMI specification and the MS-DOS extensions to the DPMI. The MS-DOS extensions provide additional functionality not defined by the DPMI specification. DPMI is a standard that specifies how MS-DOS should handle protected-mode calls. Windows 3.*x* in enhanced mode provides DPMI services.

If you have Windows 3.*x* installed on your computer and plan to develop Windows programs, configure your system for development under Windows. An

MS-DOS session in Windows enhanced mode provides DPMI services. Using Windows offers an important advantage: you can have multiple MS-DOS windows that can operate simultaneously, so you can edit in one window while compiling in another.

Other suitable DPMI servers, such as 386-Max version 6.*x* by Qualitas, are available. If these servers support version 0.9 of the DPMI specification and the MS-DOS extensions to the DPMI, they are compatible with Microsoft C/C++.

Revising System Files

Many suggestions in this chapter require that you modify your CONFIG.SYS or AUTOEXEC.BAT files. The following procedures update these system files safely:

- Make a backup copy of these files before modifying them.
- To disable a statement without deleting it, insert the word REM in front of an AUTOEXEC.BAT or CONFIG.SYS statement. (Note that use of REM in a CONFIG.SYS file generates a warning message in versions of MS-DOS prior to 4.0.)
- Make a bootable floppy disk in case a change to your CONFIG.SYS file makes it impossible to reboot from your hard disk.

Note When you finish making changes to the files, you must reboot your system to enable the changes.

Most of the information in this section applies to a system that is running MS-DOS 5.0 or later and Windows 3.*x*. If you use a memory manager from a vendor other than Microsoft, follow the manufacturer's instructions. For systems running MS-DOS 3.*x* or 4.*x*, you cannot load MS-DOS into high memory. This is because the DEVICEHIGH command is not available for CONFIG.SYS files, and the LOADHIGH command is not available for the AUTOEXEC.BAT file. Therefore, the upper-memory area cannot be accessed, making Microsoft C/C++ components run more slowly.

Modifying Your AUTOEXEC.BAT File

The AUTOEXEC.BAT file contains settings or definitions for several variables called "environment variables." Several environment variables need to be set correctly for the Microsoft C/C++ components to work together correctly. Setup also

adds a command to load SMARTDRV.EXE to your AUTOEXEC.BAT file. See page 25 for more information on SMARTDRV.EXE.

Required Environment Variables

The following are the environment variables required by Microsoft C/C++:

INCLUDE
Specifies the list of directories, separated by semicolons, where C/C++ INCLUDE files (.H) are located.

INIT
Specifies the directory where TOOLS.INI and CURRENT.STS initialization files are located. If INIT is set, the Programmer's WorkBench (PWB) looks for TOOLS.INI and CURRENT.STS in the directory specified by INIT. If you do not set INIT, PWB and the CodeView debugger create a CURRENT.STS file in every directory from which PWB or CodeView is invoked, making it unlikely that the correct status file will be loaded the next time PWB or CodeView is invoked.

LIB
Specifies the list of directories, separated by semicolons, where C/C++ libraries are located.

PATH
Specifies the search path for finding executable files.

TMP
Operating-system environment variable that specifies the directory where temporary files are located. Only one directory can be specified in the TMP variable. Utilities that use the TMP environment variable are NMAKE, LINK, and PWB.

HELPFILES
Required for using Help with PWB, CodeView, and QuickHelp. Specifies the list of directories where Help files (.HLP) are located.

Note Do not place Microsoft Windows Help files in the HELPFILES directory.

Optional Environment Variables

Optional environment variables for Microsoft C/C++ are:

CL
Specifies additional options for the compiler.

LINK
Specifies additional options for the linker.

SYSTEM
IF you want to move DLLs to a directory other than the one where they were installed by Setup, set SYSTEM to the new directory name.

You can use environment variables in makefiles; NMAKE automatically inherits macro definitions equivalent to every environment variable. The inherited variables can be redefined without affecting the original value of the environment variable. An environment variable can also be defined in a makefile if it has not already been defined. You can see the current settings for macros by specifying the /p option for NMAKE. See Chapter 18, "Managing Projects with NMAKE," in the *Environment and Tools* manual for more information.

Using the TMP and TEMP Environment Variables

Microsoft programming languages and the utilities included with them use the environment variable TMP. This operating-system environment variable is typically set in the AUTOEXEC.BAT file and is assigned to the drive and directory to be used for temporary file storage.

While Microsoft development tools use the TMP environment variable, Microsoft applications, such as Microsoft Word, use an environment variable called TEMP. Both variables are set to a drive and directory for temporary file storage. Since compilers and other development tools generally use more temporary disk space than applications, you may want to assign TMP to your hard disk. Applications like Microsoft Word generally use less temporary disk space, allowing you to set TEMP equal to a small RAM drive.

Since the Microsoft C/C++ Setup program uses the LIB utility to build combined libraries, there should be a minimum of 1 MB free on the drive pointed to by the TMP environment variable. LIB uses the directory specified by the TMP variable as storage for intermediate files. Files created by Setup in this directory are removed when Setup is complete. PWB also needs at least 1 MB free in the directory specified by the TMP environment variable because that directory is the location for PWB's virtual memory file.

Always set TMP to a subdirectory, as in the following example:

```
SET TMP=C:\TMP
```

MS-DOS places a limit on the number of files that can be created in a root directory. For more information on how Windows uses temporary files and how to change file settings, see your Windows documentation.

Setting Environment Space

The memory area created by the operating system to store environment variables and their values is called the "environment space." If the compiler can't find files like include files, or if you receive this error message when rebooting:

```
Out of environment space
```

the available environment space is insufficient to hold the definitions of the environment variables.

The default environment size is 256 bytes. The size of the current environment is 256 bytes or the amount of actual memory used by the environment variables rounded up to the next 16 bytes. The limits are 160 to 32,768 bytes. The "current" environment is the actual memory being used for the environment variables, not the size specified with the /e option of the SHELL command in the CONFIG.SYS file. (The size of the current environment may be less than the size specified with the SHELL command.) For correct operation, set the value for the environment size to at least 1024 with this statement:

```
SHELL = C:\DOS\COMMAND.COM /e:1024
```

See your MS-DOS documentation for more information on the SHELL command.

Modifying Your CONFIG.SYS File

The amount of memory available and the configuration for MS-DOS and Windows are controlled by your CONFIG.SYS file, your SYSTEM.INI file, and your .PIF files. If you are not familiar with these files, see your MS-DOS and Windows 3.*x* documentation.

If you choose to let Setup update your system files for you, Setup automatically updates your CONFIG.SYS and AUTOEXEC.BAT. Before doing so, Setup creates backup files with the extension .OLD in the same directory in which the files were found.

If you choose to view and edit the changes during Setup, Setup automatically updates your CONFIG.SYS, AUTOEXEC.BAT and SYSTEM.INI files. Setup then displays a split-screen viewer that allows you to view and edit both versions of your CONFIG.SYS file and your AUTOEXEC.BAT file. If you choose to cancel edits at this time, Setup does not update your system files.

If you choose to save the changes to edit later, Setup does not modify your current system files. It writes all the changes to CONFIG.C70, AUTOEXEC.C70,

and SYSTEM.C70 in the C:\C700\INIT subdirectory. See page 9 for more information.

Note Depending on your existing system configuration, some of these changes are necessary for C/C++ to run on your system.

Setting the BUFFERS, FILES, DEVICE, and SHELL Commands

Setup adds the BUFFERS, FILES, DEVICE, and SHELL commands to your CONFIG.SYS file or modifies the values set for these commands.

BUFFERS

The BUFFERS command in your CONFIG.SYS file specifies the number of buffers that MS-DOS reserves for file transfers. The greater the number of buffers (up to about 50), the faster your system runs. However, past a certain value, increasing the number of buffers uses more memory without increasing speed. Each buffer requires 532 bytes of memory. The following shows recommended number of buffers in relation to hard-disk size:

Hard-Disk Size	Number of Buffers
Less than 40 MB	20
40–79 MB	30
80–119 MB	40
More than 120 MB	50

If you have SMARTDRV.EXE installed, BUFFERS is set to 10. This is because a disk-caching program such as SMARTDRV.EXE performs much of the work that additional buffers would handle.

FILES

The FILES command sets the number of files MS-DOS can access at the same time. Each file uses 48 bytes of conventional memory. If the number of FILES is set too low, compilations may fail because include files cannot be opened.

DEVICE, DEVICEHIGH

The DEVICE command loads a device driver. The DEVICEHIGH command loads a device driver into upper memory on MS-DOS 5.0 or later. DEVICEHIGH also requires adding the DOS=UMB command and a DEVICE statement to HIMEM.SYS and EMM386.EXE. Setting DOS=UMB is necessary if you want to

load programs and device drivers into the upper-memory area. This command tells MS-DOS to maintain a link to the upper-memory area. HIMEM.SYS must be installed before you specify the DOS command.

SHELL

The SHELL command specifies the name and location of the command interpreter you want MS-DOS to use and sets the environment space to 1024 bytes. The default is 256 bytes. The /p parameter tells MS-DOS to make its associated command interpreter permanent so that you cannot type EXIT to stop the command interpreter. It also tells MS-DOS to run your AUTOEXEC.BAT file when it carries out the SHELL command. See information about environment space on page 20.

Understanding Memory Requirements

Your CONFIG.SYS file includes statements that define the memory layout of your system. This section gives a description of the memory requirements for Microsoft C/C++ components.

To run CodeView, LINK, or CVPACK, you need a DPMI, VCPI, or XMS server. Windows 3.*x* in enhanced mode provides a DPMI server; EMM386.EXE provides VCPI services; HIMEM.SYS provides XMS services. CodeView requires expanded memory when running as a VCPI application. If neither DPMI or VCPI service is available, CodeView uses extended memory provided by HIMEM.SYS.

If PWB cannot allocate enough memory, it may be unable to load all extensions, list boxes may come up empty, or its performance may be sluggish. Try to make available as much conventional and extended memory as possible when running PWB. PWB will use expanded memory if it is available, but using extended memory is faster.

Determining Memory Availability

To determine the size and type of memory in your system, use the MS-DOS MEM command. The MEM command (available for MS-DOS version 4.*x* or later) displays the amount of used and free memory, lists allocated and free memory areas, and lists programs that are loaded.

Type MEM at the command-line prompt when Windows is not loaded:

```
MEM /c | more
```

The MEM command does not report the contents of the upper-memory area if you have Windows loaded. Use the /c option to tell MS-DOS to display the status

of programs loaded into conventional memory and the upper-memory area. (The /c option is available in MS-DOS versions 5.0 or later.)

MS-DOS displays three columns of information about the programs currently using system memory: Name, Size in Decimal, and Size in Hex. You will use the information in the Name and the Size in Decimal columns in the instructions described later in this chapter. See your MS-DOS documentation for more information about the MEM command.

For MS-DOS versions prior to 4.0, you can use the CHKDSK command to check free memory. You can also examine system configuration with the MSD.EXE utility provided with C/C++. See MSD.TXT in \C700\BIN for information about running MSD.EXE.

Enabling Extended Memory with HIMEM.SYS

HIMEM.SYS, a memory manager provided in the C/C++ package, is required by C/C++ version 7.0. This program provides access to extended memory and ensures that no two programs use the same part of extended memory at the same time. HIMEM.SYS conforms to the Lotus/Intel/Microsoft Extended Memory Specification (XMS), version 2.0.

If you do not have a memory manager installed, Setup adds the following statement to your CONFIG.SYS file and copies HIMEM.SYS to your C:\C700\BIN subdirectory:

```
DEVICE=C:\C700\BIN\HIMEM.SYS
```

The DEVICE command for HIMEM.SYS enables the use of extended memory. Therefore, this command must appear in your CONFIG.SYS file before any commands that start device drivers, or programs that use extended memory such as RAMDRIVE.SYS and EMM386.EXE.

To access the maximum amount of conventional memory, use high memory and the upper-memory area as much as possible. To move MS-DOS out of conventional memory and to enable access to the upper-memory area (if you have MS-DOS version 5.0 or later), your CONFIG.SYS file needs the following statements:

```
DOS=HIGH,UMB

DEVICE=C:\C700\BIN\HIMEM.SYS
```

This loads MS-DOS into HMA (high-memory area), enables the use of the upper-memory area, and lets you load device drivers and TSRs into HMA, thus keeping a large amount of conventional memory available.

Note Memory managers from other vendors may require that you remove HIMEM.SYS from your system. Follow the manufacturer's instructions.

Using EMM386.EXE as an Expanded-Memory Emulator

The EMM386.EXE device driver, which comes with Microsoft C/C++, can use extended memory to emulate expanded memory on 80386 and 80486 systems. (EMM386.EXE can also function as an upper-memory manager (see page 33). EMM386.EXE can be run under MS-DOS 3.*x* or later. Expanded memory boards and managers conform to the Lotus/Intel/Microsoft Expanded Memory Specification (LIM EMS), version 3.2 or 4.0.

Note Do not use EMM386.EXE in addition to an expanded-memory manager from another manufacturer.

EMM386.EXE provides expanded memory for systems that have only extended memory. EMM386.EXE uses about 80K of extended memory to run, in addition to the memory used to emulate expanded memory. Thus expanded memory is provided at the cost of extended memory.

Setup adds the following statement to your CONFIG.SYS file if it does not find another DEVICE statement for EMM386:

```
DEVICE=C:\C700\BIN\EMM386.EXE NOEMS
```

If you specify the NOEMS option, CodeView and CVPACK will not run outside of Windows. If you want to run DOS-extended programs both with Windows and outside Windows, change the DEVICE statement Setup inserts to:

```
DEVICE=C:\C700\BIN\EMM386.EXE 2048 RAM
```

This statement installs EMM386.EXE and allocates 2048K of extended memory to it. The default for the size of memory area allocated when installing EMM386.EXE is 256K. Make sure this command comes after the DEVICE command for HIMEM.SYS and before any commands for device drivers that use the high memory area (device drivers loaded with the DEVICEHIGH command).

If Setup finds a DEVICE statement for EMM386.SYS, an older version of this memory manager, it replaces it with EMM386.EXE.

For device drivers (such as hard-disk cards, net cards, or video display) that use memory above 1 MB, you can tell EMM386.EXE to exclude that range of memory from the memory it manages for other clients. Use the `x=` option for this. The following example excludes the memory for a hard card or a net card. (The exact memory locations will vary for different computers.)

```
DEVICE=C:\C700\BIN\EMM386.EXE RAM X=C000-CDFF X=D800-DFFF
```

Make sure that the memory area you specify is the correct range of addresses for your system before enabling this statement in your CONFIG.SYS file.

Note Use EMM386.EXE options carefully. Always save a copy of your CONFIG.SYS file when experimenting with the `x=` option.

Increasing System Speed

There are several ways to improve system speed and the speed of the programs you use. The following sections explain how to increase system speed by optimizing disk-access time and by using the SMARTDRV.EXE disk-caching program.

Optimizing Disk-Access Time

To decrease disk-access time, arrange the directories in the PATH statement of your AUTOEXEC.BAT file so that file searches are as efficient as possible. Executable files that you use often should be in directories at the beginning of your PATH statement. Removing unnecessary files from your hard disk can also decrease disk-access time. Therefore:

- Delete obsolete Help files from previous installations of other Microsoft language products, especially UTILERR.HLP and CVW.HLP. See Chapter 23, "Using Help," in the *Environment and Tools* manual for more information.
- Have Setup build only the C run-time libraries you need.
- Use the CHKDSK /f command to recover lost disk space and then delete the files CHKDSK creates. Do not use CHKDSK when Windows is loaded. See your MS-DOS documentation before using CHKDSK.

Using the SMARTDRV.EXE Disk-Caching Program

Microsoft C/C++ includes SMARTDRV.EXE version 4.0, a sophisticated block-oriented disk-caching program that significantly improves compilation and link times. SMARTDRV.EXE is not required by Microsoft C/C++, but it can reduce the amount of time your computer spends reading data from your hard disk. SMARTDRV.EXE replaces the older version, SMARTDRV.SYS, and is compatible with all versions of Windows 3.*x*.

SMARTDRV.EXE sets aside some expanded or extended memory as a cache for its own use. SMARTDRV.EXE uses this disk cache to store the information read from the hard disk when an application needs information from your hard disk. When an application attempts to read additional information from the hard disk,

the SMARTDRV.EXE program supplies the information directly from its cache instead.

If you are using RAMDRIVE.SYS to create one or more RAM drives, and are limiting the memory assigned to SMARTDRV.EXE as a result, you can increase system speed by reassigning some or all of the memory away from the RAM drive and adding it to the memory available to SMARTDRV.EXE.

When Setup installs the compiler, it installs SMARTDRV.EXE by adding this statement to your AUTOEXEC.BAT file:

```
C:\C700\BIN\SMARTDRV.EXE
```

SMARTDRV.EXE automatically loads itself into HMA under MS-DOS 5.0 if EMM386.EXE is loaded and upper-memory blocks are available as a result of a `DOS=UMB` or `DOS=UMB,HIGH` command in your CONFIG.SYS file. SMARTDRV.EXE can also be loaded into HMA with third-party memory managers such as 386-Max.

Setup also checks your CONFIG.SYS file for a DEVICE statement for SMARTDRV.SYS. If Setup finds a DEVICE statement for SMARTDRV.SYS, it puts a REM statement in front of it and adds a DEVICE statement for SMARTDRV.EXE.

```
DEVICE=C:\C700\BIN\SMARTDRV.EXE /DOUBLE_BUFFER
```

SMARTDRV.EXE must be specified in both your AUTOEXEC.BAT file and CONFIG.SYS file because there are actually two device drivers in a single file: a double-buffer driver and a disk cache. The double-buffer driver is installed in CONFIG.SYS, and the cache component of SMARTDRV.EXE is installed in AUTOEXEC.BAT.

Some disk controllers do not need double buffering; using this option when you do not need it results in a small performance penalty. Setup does not determine whether or not your system needs double buffering. Therefore, once your system is running with SMARTDRV.EXE, type:

```
SMARTDRV
```

at the command-line prompt. SMARTDRV.EXE displays disk-cache status information about your system that looks like this:

```
Microsoft SMARTDrive Disk Cache version 4.0
Copyright 1991,1992 Microsoft Corp.

Cache size: 1,048,576 bytes
Cache size while running Windows: 1,048,576 bytes

                Disk Caching Status
    drive    read cache    write cache    buffering
```

```
-----------------------------------------
  A:        yes           no            no
  B:        yes           no            no
  C:        yes           yes           yes
  D:        yes           yes           -

For help, type "Smartdrv /?".
```

Note If every line in the Buffering column is No, then you can safely remove the DEVICE statement for SMARTDRV.EXE from your CONFIG.SYS file.

SMARTDRV.EXE always copies data to your hard disk when you turn off your computer. It also writes all data to the disk when an application calls the MS-DOS reset disk function. If you want to force data to be written to disk, use the /C command-line option. If you use a third-party program to reboot your machine from a batch file, you should make sure you have

```
SMARTDRV /C
```

in the batch file prior to the reboot command. Failure to do so may cause data loss.

You can use command-line options to control the size of the cache element (/E) and the size of the read-ahead buffer (/B). The read-ahead buffer is additional information that SMARTDRV.EXE reads when the application reads information from the hard disk. The size must be specified in bytes, and the element size must be one of the following: 1024, 2048, 4096, or 8192. The read-ahead buffer must be a multiple of the element size, cannot be less than the element size, and cannot exceed 32768. The defaults are 8192 for the element size and 16384 for the read-ahead buffer. Because these will occupy conventional or UMB memory space, making them larger reduces the memory MS-DOS applications have available.

You can start the SMARTDRV.EXE program by typing `SMARTDRV` at the MS-DOS prompt before you start Windows, or by placing a command line in your AUTOEXEC.BAT file. The syntax is:

[[*drive*:]] [[*path*]] SMARTDRV.EXE[[[[*drive*[[+|–]]]]...]] [[/E:*ElementSize*]] [[*InitCacheSize*]] [[*WinCacheSize*]]]] [[/B:*BufferSize*]] [[/C]] [[/R]] [[/L]] [[/Q]] [[/S]] [[/?]]

The following list describes the command-line options available for SMARTDRV.EXE:

Use This Value:	To Do This:
drive	Specify the letter of the disk drive for which you want to control caching. If you don't specify a drive letter, floppy disk drives are read-cached but not write-cached, hard disk drives are read- and write-cashed, and CD-ROM and network drives are ignored. You can specify multiple disk drives.
path	Specify the location of the SMARTDRV.EXE file.
+ | -	Enable (+) or disable (-) caching. Use the plus (+) and minus (-) signs to override the default settings. If you specify a drive letter without a plus or minus sign, read-caching is enabled and write-caching is disabled. If you specify a drive letter followed by a plus sign (+), read- and write-caching are enabled. If you specify a drive letter followed by a minus sign(-), read- and write-caching are disabled.
/E:*ElementSize*	Specify in bytes the amount of the cache SMARTDRV.EXE moves at a time. This must be greater than or equal to 1, and a power of 2. The default value is 8K.
InitCacheSize	Specify the size in kilobytes of the cache when SMARTDRV.EXE starts (before Windows is running). The size of the disk cache affects SMARTDRV.EXE's efficiency. In general, the larger the cache, the less often SMARTDRV.EXE needs to read information from the disk, which speeds up your system's performance. If you do not specify an *InitCacheSize* value, SMARTDRV.EXE sets the value according to how much memory your system has (see the list following this list).
WinCacheSize	Limit the amount (in kilobytes) Windows can reduce the cache size. Windows reduces the size of the cache to recover memory for its own use. Windows and SMARTDRV.EXE then cooperate to provide optimum use of your system memory. When you quit Windows, Windows restores the cache to its normal size. The default value depends on how much available memory your system has (see the list following this list). If you specify a value for *InitCacheSize* that is smaller than the value specified for WinCacheSize, *InitCacheSize* is set to the same size as *WinCacheSize*.
/B:*BufferSize*	Specify the size of the read-ahead buffer. The next time the application is to read information from that file, it can read it from memory instead. The default size of the read-ahead buffer is 16K. Its value can be any multiple of *ElementSize*.

/C	Write all cached information from memory to the hard disk. SMARTDRV.EXE writes information from memory to the hard disk at times when other disk activity has slowed. You might use this option if you are going to turn off your computer, and you want to make sure all cached information has been written to the hard disk.
/R	Clear the contents of the existing cache and restart SMARTDRV.EXE.
/L	Prevent SMARTDRV.EXE from loading into upper-memory blocks (UMBs), even if there are UMBs available. You can use this option of you are using MS-DOS version 5.0 or later and UMBs are enabled.
/Q	Prevent the display of SMARTDRV.EXE information on your screen.
/S	Display additional information about the status of SMARTDRV.EXE.
/?	Display online Help about the SMARTDRV.EXE command and options.

The following list shows the default values for *InitCacheSize* and *WinCacheSize*, depending on the amount of available extended memory on your computer.

Extended Memory	InitCacheSize	WinCacheSize
Up to 1 MB	All extended memory	Zero (no caching)
Up to 2 MB	1 MB	256K
Up to 4 MB	1 MB	512K
Up to 6 MB	2 MB	1 MB
6 MB or more	2 MB	2 MB

Note Do not put the SMARTDRV.EXE cache in the expanded memory provided by EMM386.EXE. EMM386.EXE uses extended memory to emulate expanded memory that other programs can use. Although SMARTDRV.EXE can use this emulated expanded memory for its cache, it may not make your program run as quickly as it would using extended memory.

Because the optimal cache size for SMARTDRV.EXE depends on the programs you run, and on your system configuration, there is no single best setting. You should experiment to find the best cache size for your system after saving a copy of your CONFIG.SYS file. For more information, see your Windows User's Guide.

Using the RAMDRIVE.SYS Memory-Disk Program

If you don't want to use SMARTDRV.EXE, or if you have a large amount of memory, use RAMDRIVE.SYS to create a disk partition in RAM. To use it for compiler temporary files, set the TMP environment variable to the drive and directory of the RAM disk. This is a good location for other frequently used files such as your precompiled header (.PCH) files. The minimum recommended size for a RAM disk is 1 MB. If Setup finds that you do have a RAM drive and your AUTOEXEC.BAT does not have the TMP environment variable set, a SET statement for TMP is added, giving the drive name and a subdirectory for the RAM drive.

.PIF Files

The .PIF file sets the amount of expanded memory and extended memory to be used by a non-Windows program and enables background execution by default. None of the Microsoft C/C++ programs depend on expanded memory, and Windows uses only extended memory. Therefore, the PWB .PIF file provided with C/C++ sets EMS to 0 and XMS to –1. Setting XMS to –1 makes available as much extended memory as possible. The .PIF files for an MS-DOS session should also use these settings, unless you have utilities that require expanded memory.

If you have a memory card that can configure itself either as expanded memory or extended memory, you should set it to extended memory and let the software emulate expanded memory if needed. If you have a memory card that provides expanded memory only, the card is still useful and should not be removed or disabled. In this case, set `XMS = -1`. EMS should be set to the amount of expanded memory actually available, either from a memory card or from EMM386.EXE.

If you develop under Windows 3.*x*, you can allow background execution with certain exceptions. When profiling or doing any timing-dependent task, you will want exclusive execution. If you find that background compiles interfere with your foreground work, increase the priority of foreground scheduling by changing the value for Background Priority for Multitasking Options in the PWB .PIF file.

You can also minimize the number of extensions that PWB loads by editing the .PIF file. You can specify the /DA option to suppress automatic loading of all PWB extensions, or you can put a question mark (**?**) in the Optional Parameter section of the PWB .PIF file. This specifies PWB to prompt you for command-line options each time it loads. For more information, see "About Projects," and "About PWB Extensions" from the PWB Table of Contents in the Microsoft Advisor Help system.

The SYSTEM.INI File

If you configure MS C/C++ for Windows 3.0, you will not be able to run the compiler under Windows 3.1 without changing your SYSTEM.INI file. Your SYSTEM.INI must contain this statement for running under Windows 3.0:

```
DEVICE=VMCPD.386
```

Setup adds this statement to your SYSTEM.INI file. When you use the C/C++ compiler under Windows 3.1, you must remove this statement from your SYSTEM.INI file and replace it with:

```
DEVICE=*VMCPD
```

Other DPMI Servers

If you do not use Windows, you need to use a third-party DPMI server, such as 386-Max, to run the DOS-extended C/C++ compiler.

The TOOLS.INI FILE

PWB and CodeView both require a TOOLS.INI file. The [CV] and [CVW] sections from TOOLS.C70 must be merged into your TOOLS.INI file to do remote debugging or to debug p-code. TOOLS.C70 also contains PWB customization settings. TOOLS.INI needs to be located in the directory pointed to by your INIT environment variable. If you don't have a TOOLS.INI file, copy (do not rename) TOOLS.C70 to TOOLS.INI.

After you have used PWB for a while, you may want to check the sample settings in the SAMPLES.INI file for useful PWB customization information and macros.

To load only a subset of the PWB extensions, move the extensions that are not in the load set to a directory that is not on the path. Then you can load these extensions only when necessary. You can modify your TOOLS.INI to load an individual PWB extension selectively. For more information, see the "About TOOLS.INI" entry from the PWB Table of Contents in the Microsoft Advisor Help system, or see Chapter 6, "Customizing PWB," in the *Environment and Tools* manual.

A TOOLS.INI file for the system that is the target side of a remote debugging session is also useful. See Chapter 10, "Special Topics," in the *Environment and Tools* manual for more information on remote debugging.

Optimizing Available Memory

You may need to experiment to find the right memory layout to ensure that all your MS-DOS applications have enough memory to run. The right memory layout may change according to the task to be performed. Therefore, you may need to change your CONFIG.SYS file, depending on what you need to optimize. This section provides general information for making more memory available on your system when necessary.

Freeing Conventional Memory

If you have MS-DOS 5.0 or later on your system and have followed recommendations in the section on HIMEM.SYS on page 23, you have MS-DOS running in extended memory. The MS-DOS (version 5.0 or later only) Setup program installs MS-DOS so that it runs in the first 64K of extended memory, called the "high memory area" (HMA). HIMEM.SYS or another XMS driver must be loaded before you can load MS-DOS into the high memory area.

There are several other ways to free conventional memory:

- Run device drivers and other memory-resident programs in the upper-memory area (see page 33).
- Don't start unnecessary memory-resident programs. Learn the purpose of each statement in your CONFIG.SYS and AUTOEXEC.BAT files so you know what programs are loaded (see your MS-DOS documentation).
- Include DEVICE commands in CONFIG.SYS only for device drivers you really need. If your system has expanded-memory hardware, include a DEVICE command for the expanded-memory manager that comes with the memory board. Configure memory as extended, not expanded, if possible.

Freeing Extended Memory

If you are having difficulty running a program that requires additional extended memory, you can use the MEM command or MSD.EXE to see if your system contains as much extended memory as the program needs. See MSD.TXT for a description of MSD.EXE. To minimize use of extended memory:

- Make sure your CONFIG.SYS and AUTOEXEC.BAT files are not loading unnecessary programs or device drivers that are using extended memory. To verify what a particular device driver does, see your MS-DOS or Windows documentation.
- Reduce the amount of extended memory you allocate for each device driver by modifying the DEVICE command for each. See page 21 or your MS-DOS documentation.

Freeing Expanded Memory

If a program does not run because there is not enough expanded memory, first make sure that your system contains as much physical expanded memory as the program needs. Also, make sure that it has enough extended memory to emulate expanded memory by checking your memory layout with the MEM command. Then check that your CONFIG.SYS and AUTOEXEC.BAT files aren't starting unnecessary programs that use expanded memory. If you want to free more expanded memory, try the following suggestions:

- Use EMM386.EXE to provide more expanded memory. See page 24 for more information on EMM386.EXE.
- Check that the devices loaded with the DEVICE command in your CONFIG.SYS file aren't using all of your expanded memory. Then reduce the amount of expanded memory being allocated, or disable unnecessary DEVICE commands.

Using EMM386.EXE to Manage Upper Memory

The "upper-memory area" is the region of your 80386 or 80486 computer's memory that is normally set aside for system use. Parts of the upper-memory area not used are called "upper-memory blocks" (UMB). You can use UMBs for running installable device drivers and other memory-resident programs that make more conventional memory available for running programs. The MS-DOS upper-memory area manager is EMM386.EXE. You need to use MS-DOS 4.*x* or higher for EMM386.EXE to access the upper-memory area.

Some programs cannot be moved to the upper-memory area. These include HIMEM.SYS, EMM386.EXE, and MS-DOS system data.

Some programs are good choices for loading into the upper-memory area. These include the Doskey, Share, or Fastopen programs, the RAMDRIVE.SYS memory-disk program, a console driver, and other device drivers. You can also load your TSRs and device drivers into HMA if your MS-DOS version supports it.

Note The only way to find out if a program can run in the upper-memory area is to try it. Some programs do not run properly in the upper-memory area. If the program does not execute correctly, or if the system locks up, run that program in conventional memory.

To run programs in the upper-memory area, you must include the following commands in your CONFIG.SYS file for loading HIMEM.SYS and EMM386.EXE:

```
DEVICE=C:\C700\BIN\HIMEM.SYS

DEVICE=C:\C700\BIN\EMM386.EXE NOEMS
```

The DEVICE command for EMM386.EXE installs EMM386.EXE as an upper-memory manager. The NOEMS option tells MS-DOS to run EMM386.EXE to manage the upper-memory area only. Since NOEMS prevents EMM386.EXE from emulating expanded memory, use NOEMS only if your programs do not require expanded memory.

The NOEMS option for EMM386.EXE is the most efficient setting for Windows, but DOS-extended programs in Microsoft C/C++ (CodeView, CVPACK, and LINK) fail if you run them from MS-DOS without Windows.

Use this command:

```
DEVICE=C:\DOS\EMM386.EXE RAM
```

if you want to use EMM386.EXE both for the upper-memory area manager and to emulate expanded memory.

Note Microsoft Windows will be unable to allocate expanded memory to programs that need it if you specify the NOEMS option when installing EMM386.EXE. If you use such programs, use the RAM option (or no options) instead.

Put the DEVICE command for HIMEM.SYS before the DEVICE command for EMM386.EXE. The DEVICE commands for HIMEM.SYS and EMM386.EXE must appear before any other DEVICE commands.

If MS-DOS runs in the upper-memory area, your CONFIG.SYS file will have `DOS=HIGH,UMB` instead of `DOS=UMB`.

To load programs into the upper-memory area, first check your memory layout by executing the MEM command. At the end of the output from the MEM /c command, note the size in the line `Largest available upper-memory block`.

Then look in the "Conventional Memory" section of the output and find the largest device driver or program that will fit into that upper-memory block (UMB). Change the command in the CONFIG.SYS file for that device driver from DEVICE to DEVICEHIGH. For memory-resident programs, change the command in the AUTOEXEC.BAT file from LOAD to LOADHIGH. Do this for one program at a time. You must reboot your system each time.

If you get an error with one of the programs you have loaded into upper memory, or the program or device driver is still running in conventional memory after you reboot your system, it may be that the largest UMB is not large enough. Some programs require more memory when they are loaded than when they are running. Try using the SIZE= option with the DEVICEHIGH command (see the information in your MS-DOS documentation on the DEVICE command). Modify the DEVICEHIGH command in your CONFIG.SYS file to specify the hexadecimal size of the driver from the Size in Hex column of the MEM output, and restart your computer. For example, if the information in the Size in Hex column from the MEM command output for MOUSE.SYS is 39E0, you would put this statement in your CONFIG.SYS:

```
DEVICEHIGH SIZE=39E0 C:\WIN3\MOUSE.SYS
```

The SIZE= option takes effect only if needed. If using the SIZE= option doesn't allow your program to run, or if your system locks up during startup or when running the program, it is likely that the program cannot run in the upper-memory area. Change the DEVICEHIGH command to DEVICE and remove LOADHIGH commands one at a time until the program works correctly.

Some hardware programs might attempt to use the upper-memory area after EMM386.EXE has determined this memory is available for running device drivers and programs. To avoid this conflict, you can use the x option when you load EMM386.EXE. This option prevents EMM386.EXE from allocating a specified range of the upper-memory area for its use. For example, to prevent EMM386.EXE from using the addresses D800h through DFFFh for UMB, you can include the following command in your CONFIG.SYS file:

```
DEVICE=C:\DOS\EMM386.EXE NOEMS x=D800-DFFF
```

If you think your computer is set up correctly to run device drivers and programs in the upper-memory area, but nothing appears there when you use the MEM /c command, check the following troubleshooting list:

- Make sure you are not running Windows 3.*x* in 386-Enhanced mode when you execute the MEM command. The MEM command does not report the contents of the upper-memory area when you are running Windows.
- Your CONFIG.SYS file must contain the `DOS=UMB` or `DOS=HIGH,UMB` command.
- The DEVICE command for EMM386.EXE in your CONFIG.SYS file must contain the NOEMS or RAM option. RAM is the default.
- Your CONFIG.SYS file must contain a DEVICEHIGH command, or your AUTOEXEC.BAT file must contain the LOADHIGH command for each program you want to run in upper memory.
- The DEVICE command for HIMEM.SYS must appear before the DEVICE command for EMM386.EXE; the DEVICE command for EMM386.EXE must appear before any DEVICEHIGH command in your CONFIG.SYS file.

Once programs are working successfully in the upper-memory area, you can experiment to find the most efficient way to use available memory there.

In general, load device drivers and programs in order of size, from largest to smallest. Do this because MS-DOS uses the largest remaining UMB, even if that program would fit into a smaller UMB. The optimal loading order depends on the sizes of programs you are loading and the sizes of available UMB.

Optimization Summary

The following list summarizes the methods you can use to free memory:

Method	When to Use	Memory Freed	Memory Used
Install HIMEM.	Required for MS C/C++.	Makes extended memory available; frees conventional memory.	Small amount of conventional memory.
Run MS-DOS in extended memory.	If your system has extended memory.	Conventional memory.	HMA portion of extended memory.
Use EMM386.EXE as an expanded-memory emulator.	If you have an 80386 or 80486 with extended memory and your programs need expanded memory.	Expanded memory for use by programs (even if your system has no expanded memory).	Extended memory, and a small amount of conventional memory.
Make sure CONFIG.SYS and AUTOEXEC.BAT do not start unnecessary TSRs.	If you need to free memory.	Conventional, extended, or expanded memory, depending on the programs you remove.	Depends on the programs you remove.
Run device drivers and programs in the upper-memory area.	If you need to free memory.	Conventional memory.	The upper-memory area (not accessible by programs).

The following list describes the MS-DOS programs you can use to speed up your system:

Program	When to Use	What is Made Faster
SMARTDRV.EXE	If your system has XMS or EMS memory that isn't needed by programs. Don't use in conjunction with a secondary buffer cache.	All programs.
RAMDRIVE.SYS	If your system has XMS or EMS memory and you use programs that RAMDRIVE.SYS can speed up.	Programs that use temporary files or programs that you run often.

Microsoft Corporation
One Microsoft Way
Redmond, WA 98052-6399

Microsoft®

1191 Part No. 24777