Getting Started

Microsoft Corporation

Filename: LMATUTTL.DOC Project:
Template: FRONTWA1.DOT Author: Mike Eddy Last Saved By: Mike Eddy
Revision #: 6 Page: 1 of 1 Printed: 10/02/00 04:07 PM

Microsoft® MASM

Assembly-Language Development System
Version 6.1

For MS-DOS® and Windows™ Operating Systems

Information in this document is subject to change without notice. Companies, names, and data used in examples herein are fictitious unless otherwise noted. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Microsoft Corporation.

©1992 Microsoft Corporation. All rights reserved.

Microsoft, MS, MS-DOS, XENIX, CodeView, and QuickC are registered trademarks and Windows and Windows NT are trademarks of Microsoft Corporation in the USA and other countries.

U.S. Patent No. 4955066

IBM is a registered trademark of International Business Machines Corporation.
Intel is a registered trademark of Intel Corporation.
Printed in the United States of America.

Document No. DB35753-1292

Contents

Chapter 1 Microsoft Macro Assembler (MASM) Overview 1
System Requirements 2
Package Contents 2
Product Components 2
New MASM Features 3
Document Conventions 4

Chapter 2 Installing and Using MASM 5
Using Setup 5
Reviewing Installation Settings 7
System Files 9
Installing MASM for Use With Other Programming Languages 11
Running MASM 13
Running MASM from the MS-DOS Command Line 13
Running MASM Within the Windows Operating System 13
Getting More Information 14

Chapter 3 Configuring Your System 15
Understanding System Configuration Terminology 15
Choosing a Development Environment 19
Revising System Files 19
Modifying Your AUTOEXEC.BAT File 20
Modifying Your CONFIG.SYS File 23
Modifying Your .PIF Files 24
Modifying Your SYSTEM.INI File 25
Modifying Your TOOLS.INI FILE 25
Using Your DOSXNT.EXE File 26
Increasing System Speed 26
Optimizing Disk Access Time 26
Using SMARTDRV.EXE 27
Using RAMDRIVE.SYS 31
Optimizing Available Memory 31
Understanding Memory Requirements 32
Determining Memory Availability 32

Freeing Conventional Memory 33
Enabling Extended Memory with HIMEM.SYS 33
Freeing Extended Memory 34
Freeing Expanded Memory 35
Using EMM386.EXE as an Expanded Memory Emulator 35
Using EMM386.EXE to Manage Upper Memory 36
Other DPMI Servers 39
Optimization Summary 39

CHAPTER 1

Microsoft Macro Assembler (MASM) Overview

This chapter describes the features of MASM version 6.1. The following topics are included:

- System Requirements
- Package Contents
- Product Components
- New MASM features
- Documentation Conventions

System Requirements

MASM version 6.1 requires the following system configuration:

- An IBM Personal Computer or 100 percent compatible, running MS-DOS version 3.3 or later
- An 80386 or later processor
- 4 megabytes of available memory (RAM)
- One hard-disk drive with a minimum of 5 megabytes of free space (Depending on the options you select, you may need up to 9 megabytes of disk space. The SETUP program 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 1.44 megabyte, 3.5-inch floppy disk drive. (For information on the 720K MASM disk set, see “Package Contents.”)

Package Contents

Your MASM version 6.1 package should include the items listed below. If any pieces are missing, contact the retailer from whom you purchased the product.

- *Registration card.* There are many advantages to being a registered owner of MASM, 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 are already a registered owner (from an earlier version of MASM) and have upgraded, your upgrade kit will not include a registration card.
- *Disks*. Disk 1 of the MASM disk set contains a file named PACKING.TXT that lists the name, location and a brief description of each disk file in the MASM package. Most files on the disks are compressed; the SETUP program decompresses files as they are installed. MASM is distributed on five 5.25-inch high-density, or four 3.5-inch high-density disks. If you need 3.5-inch 720K disks to install MASM, please send the media order card contained in the MASM package, or call Microsoft Customer Service (1-800-426-9400).
- *Books*. Your package should contain following books:
  - *Getting Started (this book)* *Getting Started* includes information on system requirements, tells you how to set up the software, and provides instructions on optimizing your system for use with MASM.
  - *Environment and Tools* This book describes how to use the Programmer's WorkBench (PWB), the CodeView (CV) Debugger, and all the other utilities included with your MASM package.
  - *Programmer's Guide* This advanced programming text describes the enhanced features and technical details of MASM version 6.1.
  - *Macro Assembler Reference* This quick-reference book lists the utilities along with a brief description of their command-line options, directives, symbols and operators and include-file macro names. Complete information on processor and coprocessor instructions is also included.

Product Components

MASM version 6.1 includes all the components you need to develop Assembly Language programs for the MS-DOS and Windows operating systems. The following components are included:

- ML Assembler version 6.1

- Programmer's WorkBench version 2

- CodeView version 4 Debugger
- The latest versions of LINK, LIB, IMPLIB, NMAKE, BSCMAKE, CREF, H2INC, EXEHDR, CVPACK, SBRPACK, HELPMAKE, RM, UNDEL, and EXP utilities
- On-line Help for the assembler and all utilities
- Sample code
- Readme documentation for information unavailable at the time of printing.

New MASM Features

MASM version 6.1 includes the following new features:

- MASM can run as a 32-bit application under MS-DOS version 3.3 and later, and the Windows operating system version 3.*x* (see *Programmer's Guide*).
- The new command-line option sequence /Fl /Sc causes the assembler to show instruction timings in the listing file (see MASM *Reference*).
- The assembler accepts '*@file*' in the command line to specify a response file to extend the command line (see MASM *Reference*).
- The new /COFF command-line option causes the assembler to convert .OBJ output to COFF (see MASM *Reference*).
- Updated versions of the programming utilities are included (see *Environment and Tools*).
- Improved compatibility with MASM version 5.10 (see Appendix A in *Programmer's Guide*).
- Allows creation of Windows-based DLLs without the Microsoft Windows Software Development Kit (see *Programmer's Guide*).
- Includes sample code for writing DLLs for calling from the Windows operating system version 3.*x* (see *Programmer's Guide).*

Document Conventions

The MASM document set uses the following conventions:

Example	Description
COPY TEST.ASM C:	Uppercase letters represent MS-DOS commands and filenames.
INVOKE	Boldface letters indicate standard features of the MASM language: keywords, operators, and standard library functions.
expression	Words in italics indicate place holders for information you must supply, such as a filename. Italics are also occasionally used for emphasis in the text.
ML /Zi HELLO.ASM	This typeface is used for example programs, program fragments, and the names of user-defined functions and variables. It also indicates user input and screen output.
SHIFT	Small capital letters denote names of keys on the keyboard. A plus sign (+) indicates a combination of keys. For example, SHIFT+F5 tells you to hold down the SHIFT key while pressing the F5 key.
“bookmark”	The first time a new term is defined, it is enclosed in quotation marks. Since some knowledge of programming is assumed, common terms such as *memory* or *branch* are not defined.

CHAPTER 2

Installing and Using MASM

This chapter describes the MASM version 6.1 installation. It includes detailed information about:

- Using SETUP
- Reviewing Installation Settings
- Running MASM
- Getting More Information

Before running SETUP, back up the distribution disks and make sure you have enough disk space (see "System Requirements," page 1). For information on configuring your system after you have installed MASM, see Chapter 3 of this book.

Using Setup

To install Microsoft MASM version 6.1, run the SETUP.EXE program (located on Disk 1 of your installation set). The SETUP program performs all tasks necessary for installing the MASM components. You must run SETUP to install MASM, as the files on the distribution disks are compressed. SETUP decompresses the files and copies them to your hard disk. SETUP runs under MS-DOS and under the Windows operating system version 3.*x*.

You can use SETUP to perform the following:

- View the documentation notes, README.TXT, packing list, and information for users of MASM version 5.10.
- Preview the installation prompts and their defaults before installing any files.

- Install the Macro Assembler using the defaults or while modifying the loading options.
- Copy individual files from the distribution disks.

You can use the interactive installation, which is the default, to set the following options:

- Load utilities for use with the Windows operating system (default = yes).
- Load the Programmer's WorkBench (default = yes).
- Configure PWB with BRIEF-compatible environment commands (default = no).
- Load MASM.EXE for MASM version 5.10 compatibility (default = yes).
- Copy Help files, README.TXT, and other documentation files (default = yes).
- Copy the sample programs (default = no).

Note If you plan on using the Tutorials in the *Programmer's Guide*, you should load the sample programs.

- Copy a mouse driver (default = yes).
- Select the drive where you want the MASM files to reside (default = highest drive letter).
- Select the directories for the MASM component files you choose to install. These include:
  - Executable files (default = C:\MASM61\BIN)
  - Library files (default = C:\MASM61\LIB)
  - Include files (default = C:\MASM61\INCLUDE)
  - Initialization files (default = C:\MASM61\INIT)
  - Help files (default = C:\MASM61\HELP)
  - Sample files (default = C:\MASM61\SAMPLES)

Note If you are using more than one Microsoft programming language, you may want to direct the files to other than the default directories to avoid duplicating utilities. For more information on installing MASM in a multi-language environment, see "Installing MASM for Use With Other Programming Languages" on page 11.

- Check your TMP environment variable and available disk space installing any files. If the TMP environment variable is not set when you run SETUP, SETUP will propose C:\MASM61\TMP as the temporary directory on your hard disk to use during installation. This directory will not be deleted when installation is complete. SETUP will place the line

  ```
  SET TMP=C:\MASM61\TMP
  ```

  in the NEW-VARS.BAT file if a TMP environment variable is not defined.

  If there is a problem during installation, SETUP will report the error and terminate without loading MASM. For information on how SETUP uses your system's TMP environment variable, see page 21.

You can run SETUP from either the MS-DOS command line or within the Windows operating system version 3.*x*.

Running SETUP from the MS-DOS Command Line

1. Insert the disk labeled Disk 1 in the appropriate disk drive.
2. At the MS-DOS command prompt, type

   DRIVE`:\ENTER`

   where *DRIVE* is the disk drive into which you just put Disk 1.
3. Type SETUP and press ENTER to begin installation.
4. Press ENTER again to display the Main Menu screen in Figure 2.1.

Running SETUP from the Windows operating system

1. Insert the disk labeled Disk 1 in the appropriate disk drive.
2. Open the File Manager and view the contents of Disk 1.
3. Run the SETUP.EXE file by double-clicking on it with the mouse, or by selecting it and pressing ENTER.
4. Press ENTER to display the main menu.

Reviewing Installation Settings

After SETUP is complete, the SETUP Main Menu screen, shown in Figure 2.1, appears.

To review the default settings for installing MASM, choose Run SETUP Without Installing Any Files from the SETUP Main Menu. SETUP will go through the prompting screens and then return to the Main Menu screen.



Figure 2.1 SETUP Main Menu screen.

Follow the instructions on the screen. Press ENTER to proceed with a selection you have made. Use ARROW KEYS to make a selection. Press F1 for information on a selection. Press CTRL+C to quit SETUP.

If you are running SETUP interactively, the Confirm Your Choices screen shown in Figure 2.2 appears when you have viewed all the prompts. This screen allows you to change any of the installation selections you made.



Figure 2.2 SETUP Confirm Your Choices Screen

To change a setting or to access more information on each menu item, use the ARROW KEYS to make the selection, then press ENTER. To accept all the settings, select No Changes and press ENTER.

After SETUP is complete, the Environment Settings screen shown in Figure 2.3 appears.

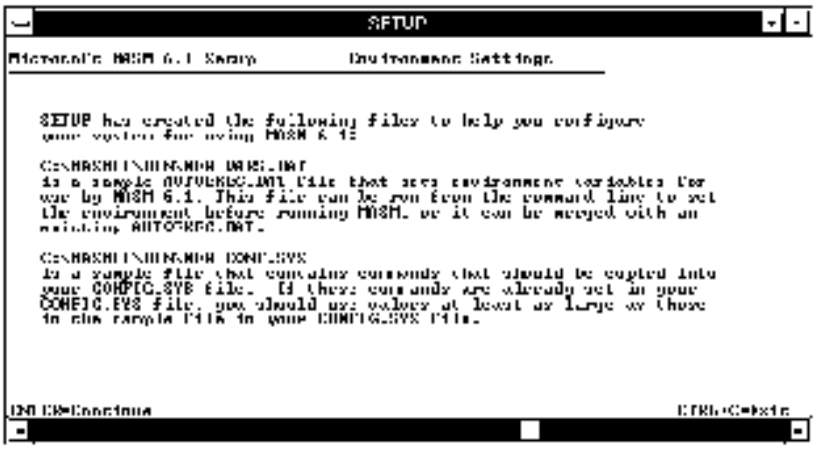


Figure 2.3 SETUP Environment Settings screen

System Files

SETUP does not modify your system files. Instead, SETUP copies recommended updates that are named NEW-VARS.BAT, NEW-CONF.SYS, and NEW-SYS.INI to your \MASM61\BIN subdirectory. Depending on your system configuration, some of the settings in these files are necessary for MASM to
run on your system.

NEW-CONF.SYS is a sample CONFIG.SYS file containing system commands for your CONFIG.SYS file. If your CONFIG.SYS file has the same commands, then make sure they are set to the same or greater values as in NEW-CONF.SYS.

NEW-VARS.BAT is a batch file that sets the MASM environment variables. You can run NEW-VARS.BAT from the command line or merge it with your existing AUTOEXEC.BAT file.

If you chose to install the Windows operating system utilities during SETUP, the SYSTEM.INI Setting screen shown in Figure 2.4 will be displayed.

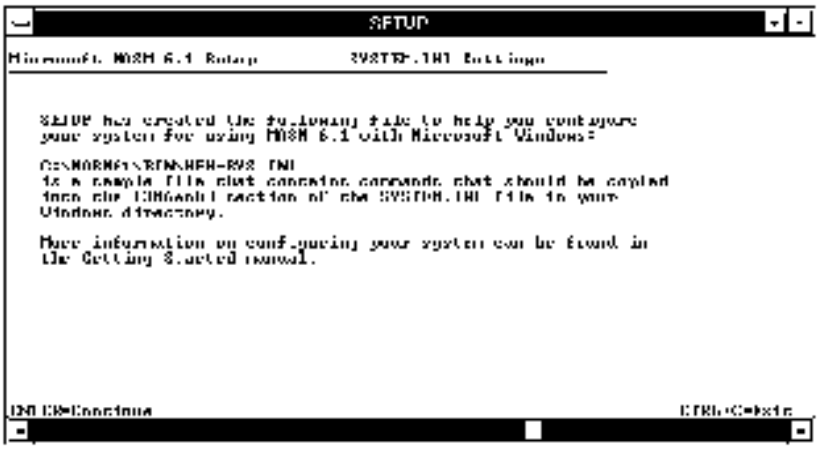


Figure 2.4 EM.INI Settings screen

NEW-SYS.INI contains commands that should exist in the [386.enh] section of your Microsoft Windows SYSTEM.INI file. It also lists any [386.enh] section lines you should delete to complete the update.

The Sample PWB Settings screen shown in Figure 2.5 is displayed next. It tells you the location of the TOOLS.PRE file that contains the default PWB setting. You can rename this file TOOLS.INI, or merge the information in it with your existing TOOLS.INI file.

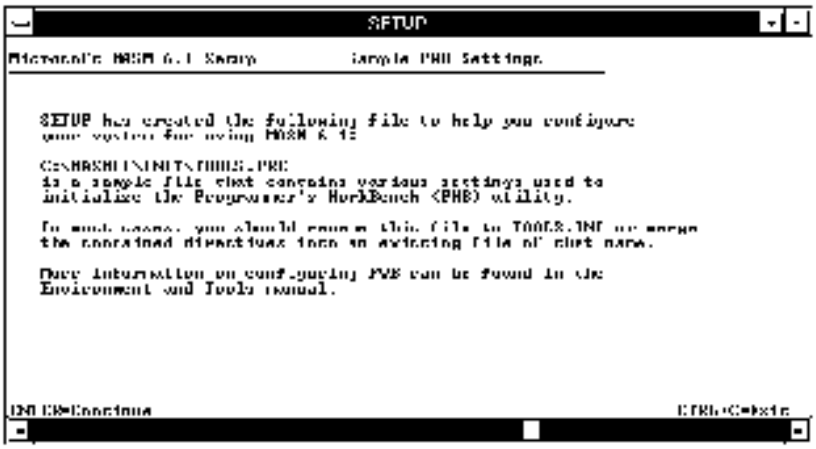


Figure 2.5 Sample PWB Settings screen

The Memory Utilities screen shown in Figure 2.6 is displayed next. It tells you the location of HIMEM.SYS, RAMDRIVE.SYS and SMARTDRV.EXE. For more information about these memory utilities, see Chapter 3 of this book.

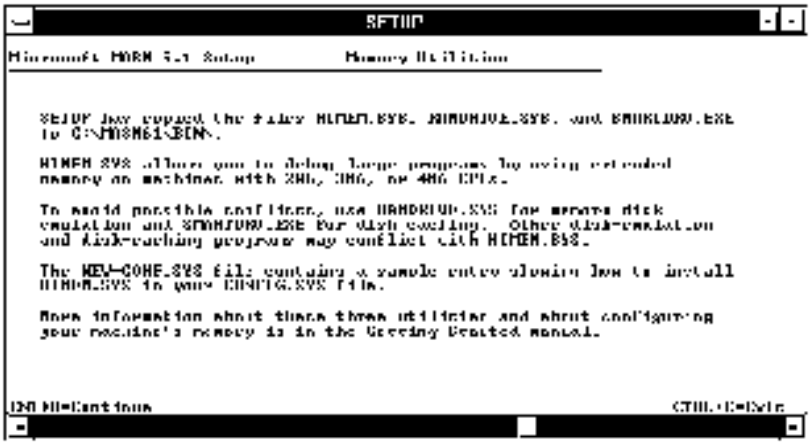


Figure 2.6 Memory Utilities screen

Pressing ENTER displays SETUP's Main Menu screen again. From here you may exit SETUP, view documentation files, or run the installation process again. You may run SETUP again at any time to load any files you chose to exclude during this installation.

Installing MASM for Use With Other Programming Languages

If you will be using MASM with other Microsoft programming languages, such as Microsoft C/C++, you may already have versions of PWB, CV and the other programming utilities loaded. You can install MASM in one of three ways, depending on your working requirements and available hard disk space:

- Install MASM in your existing MASM tree structure, if one exists. This will update all identically named and placed files in the MASM tree. The SETUP program defaults to this option. If a copy of MASM version 5.10 is located in the \MASM directory, it will be renamed to OLDMASM.EXE.
- Install MASM in your high-level language tree structure. This will also update identically named and placed files in the high-level language tree. For example,

if you have Microsoft C/C++ version 7.0 on drive D:, the root of your C/C++ tree is \C700. If you wanted to install MASM in your \C700 tree, you would:

- Select Install the Microsoft Macro Assembler from the Main Menu.
- Specify C700 instead of MASM in each of the target directory prompts.

- Install MASM in an independent tree. MASM will have its own complete tree structure, so any identically named utilities or files that exist between the new MASM tree and your high-level language (or previous MASM version) trees will be preserved.

  If you select this option and want to use the newest versions of the Programmer's Work Bench, CodeView, and other programming utilities loaded during MASM setup, you will need to make sure the directory that contains your executable MASM files (default \MASM61\BIN) comes before other language \BIN directories in your MS-DOS PATH statement.

If SETUP detects files that are named and located identically to files it is about to install, it checks the time/date stamps of those files. If the files are newer than the files SETUP is about to install, and the file is supplied by another Microsoft language product (other than MASM), you are warned and given three options:

- Copy new files over old files. This deletes the old files from your hard disk.
- Do not copy new files, and keep the older versions. The new versions are not copied to your hard disk.
- Exit SETUP. If you exit SETUP you can save your old files in another location.

There are two exceptions:

- If you choose to install the new MASM.EXE utility, your old MASM.EXE is renamed OLDMASM.EXE and left in the same directory.
- You are warned if SETUP is about to overwrite any version of LINK.EXE that is different from the one SETUP is about to install.

If you will be using Microsoft FORTRAN, BASIC, or C/C++ with MASM, you need to activate the appropriate PWB extensions (PWBFORT.XXT for FORTRAN, PWBBASIC.XXT for BASIC, and PWBC.XXT for C/C++). These are located in the C:\MASM61\BIN directory. To activate an extension, change the .XXT extension to .MXT. Language extensions provided with earlier versions of PWB are not compatible with PWB version 2.0, so you must use the new .XXT files if you want to use a language extension.

Note Any extensions you wrote for a previous version of PWB must be rebuilt for PWB version 2.0. Building custom PWB extensions for MASM 6.1 requires the Microsoft C/C++ Version 7 programming set. For more information on PWB extensions, see *Programmer's Guide*.

Running MASM

You can run MASM from the MS-DOS command line, or in an MS-DOS application window within the Windows operating system version 3.*x*. The configuration procedure for MASM to run under the two platforms is slightly different. (For information on using MASM after it is running, see *Programmer's Guide*.)

Running MASM from the MS-DOS Command Line

If you plan to run MASM from the MS-DOS command line, make sure of three things:

- Your computer has booted with a CONFIG.SYS file that includes the commands listed in NEW-CONF.SYS.
- The environment variables listed in NEW-VARS.BAT are set.
- The MS-DOS extender file DOSXNT.EXE is in the path or current directory.

During SETUP, NEW-CONF.SYS, NEW-VARS.BAT, and DOSXNT.EXE are copied in the directory you specify for executable files (default \MASM61\BIN).

Running MASM Within the Windows Operating System

If you plan to run MASM within the Windows operating system version 3.*x*, you may want to add some of the MASM utilities to a program group in the Program Manager. MASM.GRP is copied to the \BIN directory you specify during SETUP for your executable files.

Note Make sure the MASM61\BIN directory is in the current path before you

add MASM.GRP to your Program Manager. You may need to exit the Windows operating system to verify the current path. If the directory MASM61\BIN is not part
of the current path, you will have to add the MASM.GRP program items and icons individually.

Adding the MASM Program Group

1. Open the Program Manager.
2. From the File menu, choose New.
3. Select Program Group.
4. Choose the OK button.
5. Type *DRIVE*`:\MASM61\BIN\MASM.GRP` and press ENTER

DRIVE is the MASM-resident drive. (If you specified a different directory for your executable files during the SETUP program, type that as the path for MASM.GRP instead.)

The Program Manager adds a new MASM Program Group. It has five program items: Programmer's WorkBench, MASM 6.1 Reference, CodeView, MS-DOS CodeView, and WXServer.

Once you have added your MASM program group and included the statements from NEW-SYS.INI to your SYSTEM.INI file, you must exit the Windows operating system to save your changes. Restart the Windows operating system to use the items in your new MASM program group.

For more information on adding program items and groups to your Windows operating system, see your Windows operating system *User's Guide*. For information on using PWB, CV or WX Server, see *Environment and Tools*.

Getting More Information

While SETUP is running, press F1 during any screen to access more information on the highlighted option.

For information on particular components, commonly asked questions, or information not available at the time of printing, use SETUP's Main Menu to view the README.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 contacting Microsoft located in the front of *Environment and Tools*.

CHAPTER 3

Configuring Your System

This chapter describes how to configure your system for optimal use of MASM and explains the recommended modifications to your system files (CONFIG.SYS, AUTOEXEC.BAT, and SYSTEM.INI).

This chapter also provides information on conventional memory, extended memory, expanded memory, and memory managers. This will help you:

- make more memory available for MASM and other programs
- optimize the speed at which your programs run
- use the memory in your system more efficiently

You may need to experiment with the described techniques to 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 MASM.

Understanding System Configuration Terminology

This section defines terms that can help you understand the configuration information in this chapter. Figure 3.1 shows the relationship between the different memory areas.

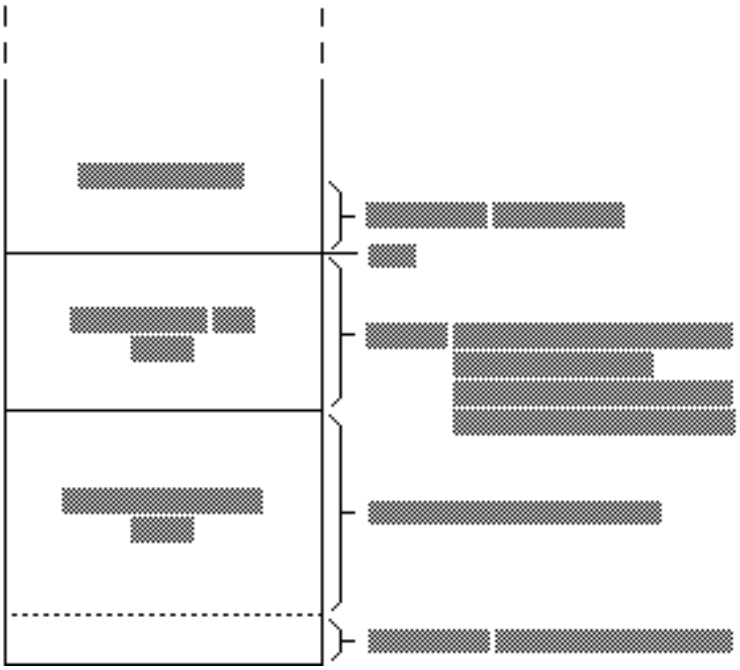


Figure 3.1 Memory Locations

Conventional (Real) Memory
: The first 640K of memory in a computer using an Intel-compatible processor. All MS-DOS systems have conventional memory. All application programs can use conventional memory without additional memory-management programs.

Extended Memory
: Memory above the first 1 MB of memory on systems with 80286 or higher processors. Most 80386 computers come with some extended memory. Extended memory requires an extended memory manager, such as HIMEM.SYS, to prevent programs from using the same area of extended memory at the same time. You need nearly 3.5 megabytes of extended memory to run MASM on your system (4 megabytes of total system memory).

High Memory Area (HMA)
: The first 64K of extended memory. Systems using MS-DOS version 5.0 or later can load MS-DOS into the HMA. This will free about 50K of conventional memory.

Expanded Memory
: 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, an expanded memory manager “maps” or copies the appropriate page to an area called a “page frame” in upper memory. 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
: Also referred to as High MS-DOS Memory. The 384K of memory 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 your system has an 80386 or 80486 processor and extended memory, MS-DOS can be loaded into UMBs, so more conventional memory is free for programs. MS-DOS version 5.0 or later has commands that enable you to store certain device drivers and programs in upper memory. 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, and 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 upper memory, your system must have a memory manager. MASM provides two memory managers, HIMEM.SYS and EMM386.EXE, that can be installed on your system. Although memory managers take up some space in conventional memory, they provide access to extended memory, expanded memory, and upper memory.

HIMEM.SYS
: A memory manager that provides access to extended memory. HIMEM.SYS is required for MASM.

EMM386.EXE
: A device driver provided by MASM to control expanded memory and provide access to upper memory. The EMM386.EXE memory manager can also use extended memory to emulate expanded memory (see page 35).

SMARTDRV.EXE
: A device driver provided by MASM 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.)

Double Buffering
: A SMARTDRV option that provides compatibility for hard disk controllers that cannot work with virtual memory.

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 MASM for systems running MS-DOS (version 4.*x* and later) that reduces disk access. RAMDRIVE.SYS creates a virtual disk drive in RAM to emulate a physical disk drive.

DOS Protected Mode Interface (DPMI)
: A published specification for handling MS-DOS calls in protected mode programs. The Microsoft Windows operating system version 3.*x* provides DPMI services and is therefore called a “DPMI server.”

MS-DOS Extensions to the DPMI
: These extensions provide additional functionality not required by the DPMI specification. The Windows operating system version 3.*x* provides this additional functionality.

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

MS-DOS-Extended Programs
: Programs that have a protected mode MS-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
: WXSRVR.EXE works with WX.EXE, a real-mode MS-DOS program, to allow Windows-based programs to be invoked from within a Windows operating system MS-DOS application window.

Choosing a Development Environment

The MASM assembler components require extended memory and an XMS memory manager, such as HIMEM.SYS. MASM supports the DPMI and VCPI specifications, and will use any available DPMI and VCPI allocated memory. However, neither DPMI or VCPI is required to run MASM.

You may run MASM within MS-DOS, or within an MS-DOS application window under the Windows operating system. If you are using the Windows operating system, version 3.*x*, you can have multiple MS-DOS application windows operating simultaneously. For example, you can use one MS-DOS window for editing and another for compiling.

Revising System Files

You will need to modify your CONFIG.SYS and AUTOEXEC.BAT files to use some of the MASM features. Perform the following procedures to 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 system disk in case a change to your CONFIG.SYS file makes it impossible to restart from your hard disk.

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

The information in this section applies to a system running MS-DOS version 5.0 or later and the Windows operating system version 3.*x*. If you are not using a Microsoft memory manager, follow the manufacturer's instructions. If your system is running MS-DOS version 3.*x* or version 4.*x*, you cannot load MS-DOS into high memory, because the DEVICEHIGH command is not available for CONFIG.SYS files, and the LOADHIGH command is not available for the AUTOEXEC.BAT file in these versions of MS-DOS. Since upper memory cannot be accessed, less memory is available for the MASM components.

Modifying Your AUTOEXEC.BAT File

The AUTOEXEC.BAT file contains settings or definitions for environment variables. Several environment variables need to be defined so the MASM components can work together optimally. The NEW-VARS.BAT file includes most of the SET statements you need to run MASM.

Required Environment Variables

The following environment variables are required by MASM:

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 are invoked, making it unlikely that the correct status file will be loaded the next time PWB or CodeView is run. Only one directory should be specified for the INIT variable.

PATH
: Specifies the search path for finding executable files.

TMP
: Operating-system environment variable that specifies the directory for temporary files. 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, and the filenames of specific .HLP files. Wildcard characters are allowed in the HELPFILES variable to indicate more than one .HLP file.

Note Do not place Microsoft Windows-based Help files in the HELPFILES directory. They are not compatible with MASM help.

Optional Environment Variables

The following environment variables are optional for MASM:

LIB
: Specifies the list of directories, separated by semicolons, where library files (*.LIB) are located.

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

MASM
: Additional options for the MASM 5.10 compatibility driver.

ML
: Specifies additional options for the assembler.

LINK
: Specifies additional options for the linker.

You can use environment variables in makefiles. NMAKE uses a set of macro definitions equivalent to the setting of each environment variable when it runs. The macro definitions can be redefined without changing the value of the environment variable. An environment variable can also be defined in a makefile if it has not already been defined. You can view the current settings for macros by specifying the /p option for NMAKE. For more information on NMAKE and its options, see the *Environment and Tools*.

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 used for temporary file storage.

For example, PWB 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.

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 use more temporary disk space than applications, you may want to assign TMP to your hard disk. Applications generally use less temporary disk space, allowing you to set TEMP equal to a small RAM drive.

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

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

Remember the following when using the TMP environment variables:

- If the TMP environment variable is not set when you run SETUP, SETUP will prompt you for the path to store temporary files (default=\MASM61\TMP). If the path you give it doesn't exist, SETUP will create the directory. If you specify a drive in the path that doesn't exist, SETUP will create and use \MASM61\TMP. This temporary directory is not deleted when SETUP is finished installing MASM. After installation, SETUP will place the line

  ```
  SET TMP=\MASM61\TMP
  ```

  in the NEW-VARS.BAT file if a TMP environment variable is not set when it is run.

- If your TMP environment variable points to a location on a network drive, make sure that the directory is not write protected.
- Make sure the directory pointed to by your TMP environment variable exists. If you set the TMP environment variable to a non-existing directory, SETUP will use the root directory of the current drive for storing temporary files. This can cause a problem, since MS-DOS limits the number of files that can be created in a root directory.

For more information on how MS-DOS and the Windows operating system use temporary files and how to change file settings, see your MS-DOS or Windows operating system 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 assembler 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 /p
```

Note Use the SHELL options carefully. It's a good idea to have a system disk with working CONFIG.SYS and AUTOEXEC.BAT files when experimenting with your system.

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 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 operating system 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 operating system, version 3.*x* documentation.

SETUP does not modify your current system files. It writes all suggested changes
to the NEW-CONF.SYS, NEW-VARS.BAT, and NEW-SYS.INI files in the C:\MASM61\BIN subdirectory. For more information about these files, see page 9.

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

SETUP adds the BUFFERS, FILES, and DEVICE commands to your NEW-CONF.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 increasing the number of buffers past a certain value will cause your system to use more memory without increasing speed. Each buffer requires 532 bytes of memory.

Table 3.1 shows the recommended number of buffers in relation to hard disk size.

Table 3.1 Recommended Number of Buffers

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, set BUFFERS to 10. SMARTDRV.EXE provides much of the increase in system speed that setting BUFFERS to a higher value would accomplish.

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. The default value is 8. If you will be running MASM in a Windows operating system MS-DOS application window, use the setting recommended in your Windows operating system *User's Guide*. If you will be running MASM from the MS-DOS command line, this value will depend on the size of your programs. Start with a minimum of 20, and increase it to a higher value if necessary.

DEVICE, DEVICEHIGH

The DEVICE command loads a device driver. The DEVICEHIGH command loads a device driver into upper memory on MS-DOS version 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 upper memory. This command tells MS-DOS to maintain a link to upper memory. HIMEM.SYS must be installed before you specify the DOS command.

Modifying Your .PIF Files

The .PIF file sets the amount of expanded memory and extended memory used by a program that is not Windows-based and enables background execution by default. None of the MASM programs depend on expanded memory, and the

Windows operating system uses only extended memory. Therefore, the PWB .PIF file provided with MASM sets expanded memory to 0 and extended memory to –1. Setting extended memory 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 be configured as expanded or extended memory, 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, set extended memory to –1. Expanded memory should be set to the amount of expanded memory actually available, either from a memory card or from EMM386.EXE.

If you develop programs under the Windows operating system version 3.*x*, you can allow background execution with certain exceptions. When profiling or doing timing dependent tasks, 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.

Modifying Your SYSTEM.INI File

During installation, SETUP copies the following three lines into the file C:\MASM61\BIN\NEW-SYS.INI. These statements must be added to the [386enh] section of your SYSTEM.INI file:

```
DEVICE=C:\MASM61\BIN\DOSXNT.386
DEVICE=C:\MASM61\BIN\CVW1.386
DEVICE=C:\MASM61\BIN\VMB.386
```

The VMB.386 device line is added only if you choose to have SETUP install PWB. This line is used only by WX.EXE.

Modifying Your TOOLS.INI FILE

PWB and CodeView both use the TOOLS.INI file. Customization settings for PWB are located in TOOLS.INI, and CodeView looks for information in the [CV] and [CVW] sections of the TOOLS.INI file to do remote debugging. 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.PRE to TOOLS.INI. SETUP loads TOOLS.PRE to the C:\MASM61\INIT directory.

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

Using Your DOSXNT.EXE File

DOSXNT.EXE is the MS-DOS extender that allows you to run the MASM assembler. You must make sure that DOSXNT.EXE is located somewhere in the path or in the current working directory (default \MASM61\BIN).

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 cache 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.

To optimize disk access time, perform the following:

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

Using SMARTDRV.EXE

MASM includes SMARTDRV.EXE version 4.0, a sophisticated block-oriented disk cache program that significantly improves compilation and link times. SMARTDRV.EXE is not required by MASM, but 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 the Windows operating system version 3.*x*.

SMARTDRV.EXE sets aside expanded or extended memory as a cache. SMARTDRV.EXE uses this disk cache to store the information read from the 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 from the RAM drive and adding it to the memory available to SMARTDRV.EXE.

Install SMARTDRV.EXE by placing the following line in your AUTOEXEC.BAT file:

```
C:\MASM61\BIN\SMARTDRV.EXE
```

SMARTDRV.EXE automatically loads itself into high memory under MS-DOS version 5.0 if EMM386.EXE is loaded and upper memory blocks are available as
a result of a DOS=UMB or DOS=HIGH, UMB 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 places the following DEVICE statement for SMARTDRV.EXE into your NEW-CONF.SYS file.

```
DEVICE=C:\MASM61\BIN\SMARTDRV.EXE /DOUBLE_BUFFER
```

Using Double Buffering

If you have a SCSI (Small Computer System Interface) hard disk controller, you may need to use the double buffering feature of SMARTDRV.EXE. Double buffering provides compatibility for hard disk controllers that cannot work with virtual memory.

For double buffering to be enabled, SMARTDRV.EXE must be specified in both your AUTOEXEC.BAT file and CONFIG.SYS file. The double buffer driver is installed in CONFIG.SYS, and the cache component of SMARTDRV.EXE is installed during execution of AUTOEXEC.BAT.

To enable SMARTDRV's double-buffering option, add the following line to the end of your CONFIG.SYS file:

```
DEVICE=C:\MASM61\BIN\SMARTDRV.EXE /DOUBLE_BUFFER
```

Some disk controllers do not need double buffering, so using this option when you do not need it results in some penalty in performance. SETUP does not determine if your system needs double buffering. Therefore, once your system is running with SMARTDRV.EXE, type:

```
SMARTDRV ENTER
```

at the command-line prompt. SMARTDRV.EXE displays disk cache information as illustrated in the following example:

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

Cache size: 1,048,576 bytes
Cache size while running the Windows operating system: 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, you do not need the DEVICE statement for SMARTDRV.EXE in your CONFIG.SYS file.

SMARTDRV.EXE always copies data to your hard 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 non-Microsoft utility 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 include this line can result in loss of data.

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 upper memory, making them larger reduces the available memory for MS-DOS applications.

You can start SMARTDRV.EXE program by typing SMARTDRV at the MS-DOS prompt before you start the Windows operating system, 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:

Option	Description
drive	Specify the letter of the disk drive you want to control disk caching. If you don't specify a drive letter, floppy disk drives read operations are cached but write operations are not, hard disk drive read and write operations are cached, 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 (–) disk 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 operations are cached and write operations are not. If you specify a drive letter followed by a plus sign (+), read and write operations are both cached. If you specify a drive letter followed by a minus sign(–), neither read nor write operations are cached.

Option	Description
/E:*ElementSize*	Specify in bytes the amount of the disk 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 disk cache when SMARTDRV.EXE starts (before the Windows operating system is running). The size of the disk cache affects SMARTDRV.EXE's efficiency. In general, the larger the disk 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 Table 3.2).
WinCacheSize	Limit the amount (in kilobytes) the Windows operating system can reduce the disk cache size. The Windows operating system reduces the size of the disk cache to recover memory for its own use. The Windows operating system and SMARTDRV.EXE cooperate to provide optimum use of your system memory. When you exit the Windows operating system, it restores the disk cache to its normal size. The default value depends on how much available memory your system has (see Table 3.2). 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 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 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 information has been written to the hard disk.
/R	Clear the contents of the existing disk 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 if 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.

Table 3.2 shows the default values for *InitCacheSize* and *WinCacheSize*, depending on the amount of available extended memory on your computer.

Table 3.2 Default Values for InitCacheSize and WinCacheSize

Extended Memory	InitCacheSize	WinCache Size
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 disk 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.

Also, earlier versions of SMARTDRV.EXE allowed you to use the /a switch to direct SMARTDRV to use expanded memory. This function is no longer valid with the current version.

The optimal disk cache size for SMARTDRV.EXE depends on the programs you run, and your system configuration. You should experiment to find the best disk cache size for your system, after you have saved a copy of your CONFIG.SYS file. For more information, see your Windows operating system *User's Guide*.

Using RAMDRIVE.SYS

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 assembler temporary files, set the TMP environment variable to the drive and directory of the RAM disk. The minimum recommended size for a RAM disk is 1 MB.

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 optimal 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.

Understanding Memory Requirements

Your CONFIG.SYS file includes statements that define how your system uses memory. This section describes the memory requirements for MASM components.

To run CodeView, LINK, ML, or CVPACK, you need an XMS, DPMI, or VCPI, server. HIMEM.SYS provides XMS services; EMM386.EXE provides VCPI services; the Windows operating system version 3.*x* in enhanced mode provides DPMI 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 the extensions you selected, list boxes may come up empty, and performance may be slower
than usual. Try to make as much conventional and extended memory available as possible when running PWB. PWB will use expanded memory if it is available, but will run faster using extended memory.

Determining Memory Availability

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

Type MEM at the command-line prompt when the Windows operating system is not running:

```
MEM /c | more
```

The MEM command does not report the contents of upper memory if you have the Windows operating system loaded. Use the /c option to tell MS-DOS to display the status of programs loaded into conventional memory and upper memory. (The /c option is only available in MS-DOS version 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. See your MS-DOS documentation for more information about the MEM command.

You can use the CHKDSK command to check the amount of free conventional memory if you are using an MS-DOS version earlier to version 4.0. You can also examine system configuration with the MSD.EXE utility provided with MASM. See MSD.TXT in C:\MASM61\BIN for information about running MSD.EXE.

Freeing Conventional Memory

If you have MS-DOS version 5.0 or later on your system and have followed recommendations in the section on HIMEM.SYS, you should 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 upper memory (see pages 17 and 24).
- 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.

In some situations involving the /Zi option, MASM requires 500Kb of free conventional memory for a successful assembly. This can be resolved by assembling and linking in separate steps (that is, by using NMAKE). For more information on the /Zi option, see *Programmer's Guide*. For more information on using NMAKE, see *Environment and Tools*.

Enabling Extended Memory with HIMEM.SYS

HIMEM.SYS, a memory manager provided in the MASM package, is required by MASM version 6.1. This program provides access to extended memory and ensures that two programs do not use the same part of extended memory at the same time. HIMEM.SYS conforms to the Lotus/Intel/Microsoft (LIM) Extended Memory Standard (XMS), version 2.0.

If you do not have a memory manager installed, SETUP adds the following statement to your NEW-CONF.SYS file and copies HIMEM.SYS to your C:\MASM61\BIN subdirectory:

```
DEVICE=C:\MASM61\BIN\HIMEM.SYS
```

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

Note Some systems require the use of the HIMEM.SYS /m switch. Check your computer's operations guide and MS-DOS manual for details.

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

```
DEVICE=C:\MASM61\BIN\HIMEM.SYS
DOS=HIGH,UMB
```

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

Note If you are not using a Microsoft memory manager, you may need to remove HIMEM.SYS from your system. Follow the manufacturer's instructions.

Freeing Extended Memory

If you are having difficulty running a program that requires additional extended memory, use the MEM command or MSD.EXE to make sure your system has enough extended memory to run the program. 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 operating system documentation.

- Reduce the amount of extended memory you allocate for each device driver by modifying the DEVICE command for each.

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:

- Use EMM386.EXE to provide more expanded memory.
- 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 as an Expanded Memory Emulator

The EMM386.EXE device driver included with MASM 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 36). EMM386.EXE can be run under MS-DOS version 3.*x* or later. Expanded memory boards and managers conform to the Lotus/Intel/Microsoft (LIM) Expanded Memory Standard (EMS) version 3.2 or 4.0.

Note Do not use EMM386.EXE with an expanded memory manager from another manufacturer.

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

The typical NEW-CONF.SYS statement for EMM386.EXE is:

```
DEVICE=C:\MASM61\BIN\EMM386.EXE NOEMS
```

If you specify the NOEMS option, CodeView and CVPACK will not run outside of the Windows operating system. If you want to run MS-DOS-extended programs

both within and outside of the Windows operating system, change the DEVICE statement in your CONFIG.SYS file as follows:

```
DEVICE=C:\MASM61\BIN\EMM386.EXE 2048 RAM
```

This statement installs EMM386.EXE with an allocation of 2048K memory. 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 you are currently using EMM386.SYS, an older version of this memory manager, replace it with EMM386.EXE and adjust the filename in your CONFIG.SYS file.

You can exclude certain blocks or ranges of memory from the memory you allocate to EMM386.EXE. This prevents EMM386.EXE from using ranges of memory reserved for device drivers (such as hard disk cards, net cards, or video display) that use memory above 1 MB.

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, depending on your computer and the memory cards you have installed.)

```
DEVICE=C:\MASM61\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.

You can use the MEM command to determine the hex locations and sizes of the device drivers you have loaded. This will enable you to fine tune the *X*= setting for EMM386.EXE. (For information on the MEM command, see page 32 or your
MS-DOS Reference.)

Using EMM386.EXE to Manage Upper Memory

Upper memory is the region of your 80386 or 80486 computer's memory that is used by the system. Parts of upper memory that are not used are called "upper memory blocks" (UMB). You can use UMBs for running device drivers and other memory-resident programs that make more conventional memory available for running programs. The MS-DOS upper memory manager is

EMM386.EXE. You need to use MS-DOS 4.*x* or higher for EMM386.EXE to access upper memory.

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

Programs such as DOSKEY, SHARE, FASTOPEN, RAMDRIVE.SYS, console and other device drivers are good choices for loading into upper memory. You can also load your TSRs into high memory if your MS-DOS version supports it.

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

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

```
DEVICE=C:\MASM61\BIN\HIMEM.SYS
DEVICE=C:\MASM61\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 upper memory 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 the Windows operating system, but MS-DOS-extended programs in MASM (CodeView, CVPACK, and LINK) fail if you run them from MS-DOS without the Windows operating system.

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 The Microsoft Windows operating system 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 upper memory, your CONFIG.SYS file will have DOS=HIGH,UMB instead of DOS=UMB.

To load programs into upper memory, 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 restart 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 restart 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 upper memory. 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 upper memory after EMM386.EXE has determined this memory is available for running device drivers and programs. To avoid this, you can use the *x*= option when you load EMM386.EXE. This option prevents EMM386.EXE from allocating a specified range of upper memory 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 upper memory, but nothing appears there when you use the MEM /c command, check the following:

- Make sure you are not running the Windows operating system version 3.*x* in 386-Enhanced mode when you execute the MEM command. The MEM command does not report the contents of upper memory when you are running the Windows operating system.
- 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 upper memory, you can experiment to find the most efficient way to use available memory.

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.

Other DPMI Servers

Though MASM version 6.1 will make use of any memory allocated by a DPMI or VCPI server. it does not require the presence of a DPMI server. MASM does require an XMS server, such as HIMEM.SYS.

Optimization Summary

Table 3.3, Summary of Optimization Methods, lists the different optimization methods and how they are used.

Table 3.3 Summary of Optimizing Methods

Method	When to Use	Memory Used
Use HIMEM.SYS.	Required for MASM version 6.1.	Conventional
Run MS-DOS in extended memory.	If your system has extended memory.	High memory (HMA)
Use EMM386.EXE as an expanded memory emula-tor.	If your system has extended memory and your programs need expanded memory.	Extended and conventional
Make sure CONFIG.SYS and AUTOEXEC.BAT are not loading unnecessary programs or device drivers.	To free memory.	Depends on the programs you remove
Run device drivers and programs in upper memory	To free memory.	Upper memory
Use SMARTDRV.EXE. (Don't use with a secondary buffer cache.)	If your system has extended or expanded memory that isn't needed by programs.	All programs
Use RAMDRIVE.SYS.	If your system has extended or expanded memory and you use programs that RAMDRIVE.SYS optimize.	Programs that use tempo-rary files or programs that you run often