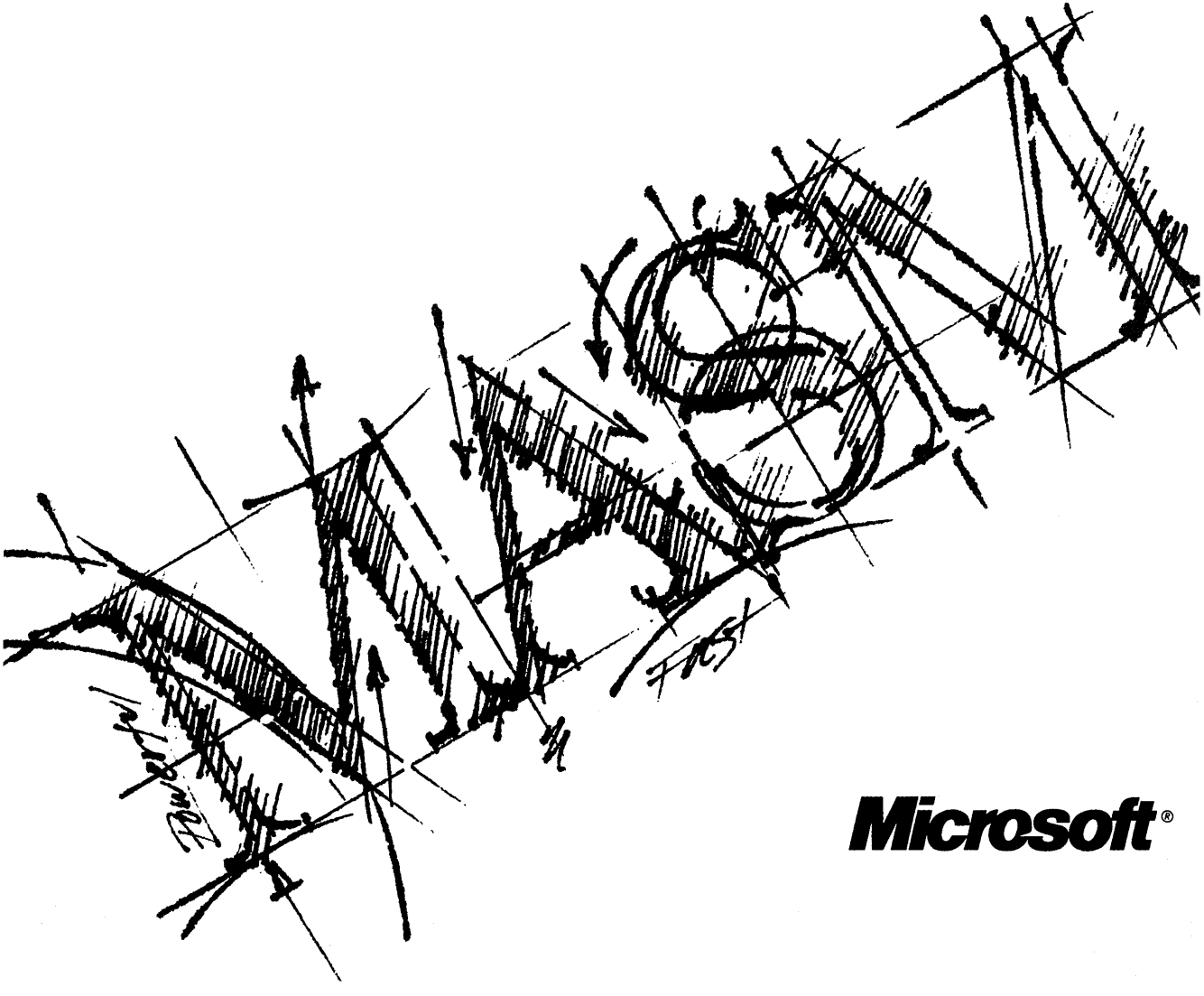
Quick Start for Microsoft® Macro Assembler 5.0 and 5.1 Users

Important Note on Backward Compatibility of MASM 6.0



Microsoft®

1. Getting Up and Running Quickly under MASM 6.0

Microsoft Macro Assembler version 6.0 offers major advances over previous versions of MASM. It incorporates many features previously found only in high-level languages, significantly increasing your programming productivity, while also offering substantial performance improvements. To provide these major enhancements while still offering backward compatibility with previous versions, MASM 6.0 has a special **compatibility mode** of assembly. You can access this mode in three ways:

- **By using the conversion driver, MASM.EXE.** This approach converts your existing command-line options to the new syntax, adds a compatibility option (/Zm), and invokes the new ML.EXE assembler. It also lets you use your existing MAKE and batch files.
- **By using the new ML.EXE assembler with the /Zm option.** This approach eliminates the need for the conversion driver but requires you to convert your command-line options to the new MASM 6.0 syntax.
- **By placing an OPTION M510 statement at the start of each file.** This approach is equivalent to adding the /Zm option to the command line. It lets you assemble old and new modules under ML.EXE using a single command line.

Any of these three approaches will, in most cases, allow you to assemble your existing code under MASM 6.0, gaining full access to its new capabilities. Or, if you have existing MASM 5.0 or 5.1 object files and libraries that don't require changes, you can simply link them with new object files and libraries and run your code: no additional steps are required.

If your code assembles under MASM 5.0 or 5.1 but won't assemble under the compatibility mode, see Section 2, "Additional Guidelines for Using the /Zm Option."

If you want to modify your existing code to assemble under MASM 6.0 without the /Zm option, see Section 3, "Modifying Existing Code to Assemble without the /Zm Option." Additional information is provided in Appendix A (Section A.2, "Compatibility between MASM 5.1 and 6.0") of the Macro Assembler *Programmer's Guide*.

If you need additional help, Microsoft Product Support Services has established a special number for assistance with MASM 6.0.

Call: **(206) 646-5109**

Technical assistance is available at this number Monday through Friday, 8 a.m. through 5 p.m. Pacific Time. If you are located outside the U.S., you must contact your local Microsoft Subsidiary.

2. Additional Guidelines for Using the /Zm Option

In certain situations, your code will not assemble with the /Zm option alone. The two most common cases are described below.

2.1 New Reserved Word Used as a Label

Many new reserved words have been added to MASM 6.0. If your existing code uses a reserved word as a symbol name, you will typically get a syntax error on assembly. For a list of all MASM 6.0 new reserved words, see Appendix A (Section A.2.2.9, "OPTION NOKEYWORD") of the *Programmer's Guide*; for a complete list of all reserved words, see Appendix D.

To locate reserved words, run the assembler and look at any lines that generate syntax errors. If they contain reserved words, either change the symbol names or use the `OPTION NOKEYWORD` statement, as shown in the following example:

```
OPTION NOKEYWORD:<INVOKE STRUCT>
```

This statement would make the `INVOKE` and `STRUCT` keywords unavailable as reserved words. See Appendix A (Section A.2.2.9, "OPTION NOKEYWORD") for more information.

2.2 Pass-Dependent Constructs Used

To optimize performance, MASM 6.0 uses an *n*-pass assembler rather than the two-pass assembler used by previous releases. This means it reads the source code only once: additional optimization passes are made on an intermediate file. Consequently, if your code includes pass-dependent constructs, it will not assemble under MASM 6.0.

Typically, problems with code requiring a two-pass assembler occur when you use:

- An **IF2** or **ELSEIF2** directive.
- An **ELSE** or **ELSEIF** block with an **IF1** directive.
- **IFDEF** and **IFNDEF** with forward references.
- The **.TYPE** operator with a forward reference in an **IF**, **IFE**, or **IFNE** directive.

The first two cases will generate error A2061:

```
[ELSE]IF2/.ERR2 not allowed: single-pass assembler
```

The second two will generate warning A5006:

```
IF condition may be pass-dependent
```

See Appendix A ("Obsolete Two-Pass Directives" in Section A.2.1.3) for some examples of common pass-dependent constructs and ways to correct your code.

2.3. Other Differences between MASM 6.0 and Previous Versions

Several additional differences between the two assemblers may occasionally prevent code from assembling under the compatibility mode. These differences, which are further explained in Appendix A Section (A.2.1, "Rewriting Code for Compatibility") of the *Programmer's Guide* , include:

Bug Fixes from MASM 5.1

- Invalid use of **LOCK**, **REPNE**, and **REPNZ**
- No closing quotation marks in macro arguments
- Making a scoped label public
- Byte form of **BT**, **BTS**, **BTC**, and **BTR** instructions
- Default value for record fields

Design Change Issues

- Conflicting structure definitions
- Forward references to text macros outside of expressions
- **HIGH** and **LOW** applied to relocatable operands
- **OFFSET** applied to group names and indirect memory operands
- **LENGTH** operator applied to record types
- Signed comparison of hexadecimal values using **GT**, **GE**, **LE**, or **LT**
- **RET** used with a constant in procedures with epilogues
- Code labels at top of procedures with prologues
- Use of **%** as an identifier character
- **ASSUME CS** set to wrong value

Code Requiring Two-Pass Assembly

- Obsolete two-pass directives
- **IFDEF** and **IFNDEF** with forward-referenced identifiers
- Address spans as constants
- **.TYPE** with forward references

Obsolete Features No Longer Supported

- The **ESC** instruction
- The **MSFLOAT** binary format

3. Modifying Existing Code to Assemble without the /Zm Option

In most cases, using the /Zm option to assemble your existing code will be the best solution. Your code will require a minimum of changes, and you'll still be able to take advantage of the new assembler features as you modify or add to it. If you prefer to modify your code to allow it to be assembled without /Zm, follow the steps listed below. For more information, see Appendix A, (Section A.2.2, "Using the OPTION Directive") of the *Programmer's Guide*.

1. Convert the command line to the equivalent ML.EXE form and assemble your code.

First, you need to make sure your code assembles with the /Zm option. However, since you'll need to remove the /Zm option later—and since the conversion driver, MASM.EXE, automatically inserts the /Zm option when invoking ML.EXE—you won't be able to use this method of assembly. Instead, convert your command line to the new form, add the /Zm option, and assemble directly under ML.EXE.

If you need help in converting the command line, key in ML/HELP. This generates an on-screen listing of the new options, together with brief definitions.
If your code won't assemble with the /Zm option, refer to Section 2, above.

2. Add the appropriate OPTION directives.

- Begin by adding the following **OPTION** directives to your code:

```
OPTION  OLDSTRUCTS      ; Allows use of old-style structures
OPTION  OLDMACROS       ; Allows use of old-style macros
OPTION  DOTNAME         ; Allows identifiers to begin with a dot [.]
```

- If there is no **.386** or **.386P** directive in your module, also add:

```
OPTION  EXPR 16         ; Uses 16-bit precision in expressions
```

- If there is no **.MODEL** directive in your module, also add:

```
OPTION  OFFSET:SEGMENT  ; Causes OFFSET operator default to be
                        ; segment-relative rather than group-relative
```

- If there is no **.MODEL** directive with a language specifier in your module, also add:

```
OPTION  NOSCOPED        ; Makes code labels not local to the procedure
                        ; in which they appear
OPTION  PROC:PRIVATE    ; Treats code labels defined with PROC as local
                        ; unless otherwise specified
```

3. Remove /Zm from the command line and try assembling your code.

Since most of the effects of the /Zm option are replicated by the **OPTION** directives listed above, your code should now assemble. However, there are certain effects of the /Zm option that can only be enabled with this option. These effects occasionally prevent code using **OPTION** directives from assembling. Refer to Appendix A (Section A.2.2.1, "OPTION M510") of the *Programmer's Guide* for explanations of these effects and suggestions for work-around solutions.

4. Remove the OPTION directives, one at a time, reassembling after each removal.

Once your code assembles with the **OPTION** directives, remove these directives one at a time, reassembling your code after each removal. Refer to Appendix A (Sections A.2.2.2 through A.2.2.9) of the *Programmer's Guide* for suggestions on resolving any problems that appear. When you have removed the last **OPTION** directive, your code will be completely converted to MASM 6.0.

Microsoft Corporation
One Microsoft Way
Redmond, WA 98052-6399

Microsoft®
Making it all make sense™

0391 Part No. 21595