ML Error Messages

ML Fatal Errors

A1000 **cannot open file:** *filename*

The assembler was unable to open a source, include, or output file.

One of the following may be a cause:

- The file does not exist.
- The file is in use by another process.
- The filename is not valid.
- A read-only file with the output filename already exists.
- Not enough file handles exist. In MS-DOS, increase the number of file handles by changing the FILES setting in CONFIG.SYS to allow a larger number of open files. FILES=50 is the recommended setting.
- The current drive is full.
- The current directory is the root and is full.
- The device cannot be written to.
- The drive is not ready.

A1001 **I/O error closing file**

The operating system returned an error when the assembler attempted to close a file.

This error can be caused by having a corrupt file system or by removing a disk before the file could be closed.

A1002 **I/O error writing file**

The assembler was unable to write to an output file.

One of the following may be a cause:

- The current drive is full.
- The current directory is the root and is full.
- The device cannot be written to.
- The drive is not ready.

A1003 **I/O error reading file**

The assembler encountered an error when trying to read a file.

One of the following may be a cause:

- The disk has a bad sector.
- The file-access attribute is set to prevent reading.
- The drive is not ready.

A1005 **assembler limit : macro parameter name table full**

Too many parameters, locals, or macro labels were defined for a macro. There was no more room in the macro name table.

Define shorter or fewer names, or remove unnecessary macros.

A1006 **invalid command-line option:** *option*

ML did not recognize the given parameter as an option.

This error is generally caused when there is a syntax error on the command line.

A1007 **nesting level too deep**

The assembler reached its nesting limit. The limit is 20 levels except where noted otherwise.

One of the following was nested too deeply:

- A high-level directive such as **.IF**, **.REPEAT**, or **.WHILE**
- A structure definition
- A conditional-assembly directive
- A procedure definition
- A **PUSHCONTEXT** directive (the limit is 10).
- A segment definition
- An include file
- A macro

A1008 **unmatched macro nesting**

Either a macro was not terminated before the end of the file, or the terminating directive **ENDM** was found outside of a macro block.

One cause of this error is omission of the dot before **.REPEAT** or **.WHILE**.

A1009 **line too long**

A line in a source file exceeded the limit of 512 characters.

If multiple physical lines are concatenated with the line-continuation character (\ ), the resulting logical line is still limited to 512 characters.

A1010 unmatched block nesting :

A block beginning did not have a matching end, or a block end did not have a matching beginning. One of the following may be involved:

- A high-level directive such as **.IF**, **.REPEAT**, or **.WHILE**
- A conditional-assembly directive such as **IF, REPEAT, or WHILE**
- A structure or union definition
- A procedure definition
- A segment definition
- A **POPCONTEXT** directive
- A conditional-assembly directive, such as an **ELSE**, **ELSEIF**, or **ENDIF** without a matching **IF**

A1011 directive must be in control block

The assembler found a high-level directive where one was not expected. One of the following directives was found:

- **.ELSE** without **.IF**
- **.ENDIF** without **.IF**
- **.ENDW** without **.WHILE**
- **.UNTIL**[[**CXZ**]] without **.REPEAT**
- **.CONTINUE** without **.WHILE** or **.REPEAT**
- **.BREAK** without **.WHILE** or **.REPEAT**
- **.ELSE** following **.ELSE**

A1012 error count exceeds 100; stopping assembly

The number of nonfatal errors exceeded the assembler limit of 100.

Nonfatal errors are in the range A2xxx. When warnings are treated as errors they are included in the count. Warnings are considered errors if you use the /Wx command-line option, or if you set the Warnings Treated as Errors option in the Macro Assembler Global Options dialog box of PWB.

A1013 invalid numerical command-line argument : *number*

The argument specified with an option was not a number or was an invalid number.

A1014 too many arguments

There was insufficient memory to hold all of the command-line arguments.

This error usually occurs while expanding input filename wildcards (* and ?). To eliminate this error, assemble multiple source files separately.

A1015 statement too complex

The assembler ran out of stack space while trying to parse the specified statement.

One or more of the following changes may eliminate this error:

- Break the statement into several shorter statements.
- Reorganize the statement to reduce the amount of parenthetical nesting.
- If the statement is part of a macro, break the macro into several shorter macros.

A1017 missing source filename

ML could not find a file to assemble or pass to the linker.

This error is generated when you give ML command-line options without specifying a filename to act upon. To assemble files that do not have a .ASM extension, use the /Ta command-line option.

This error can also be generated by invoking ML with no parameters if the ML environment variable contains command-line options.

A1901 Internal Assembler Error
Contact Microsoft Product Support Services

The MASM driver called ML.EXE, which generated a system error.

Note the circumstances of the error and notify Microsoft Corporation by following the instructions in the "Microsoft Support Services" section of the introduction to this book.

ML Nonfatal Errors

A2000 memory operand not allowed in context

A memory operand was given to an instruction that cannot take a memory operand.

A2001 immediate operand not allowed

A constant or memory offset was given to an instruction that cannot take an immediate operand.

A2002 cannot have more than one ELSE clause per IF block

The assembler found an **ELSE** directive after an existing **ELSE** directive in a conditional-assembly block (**IF** block).

Only one **ELSE** can be used in an **IF** block. An **IF** block begins with an **IF**, **IFE**, **IFB**, **IFNB**, **IFDEF**, **IFNDEF**, **IFDIF**, or **IFIDN** directive. There can be several **ELSEIF** statements in an **IF** block.

One cause of this error is omission of an **ENDIF** statement from a nested **IF** block.

A2003 extra characters after statement

A directive was followed by unexpected characters.

A2004 **symbol type conflict :** *identifier*

The **EXTERNDEF** or **LABEL** directive was used on a variable, symbol, data structure, or label that was defined in the same module but with a different type.

A2005 **symbol redefinition :** *identifier*

The given nonredefinable symbol was defined in two places.

A2006 **undefined symbol :** *identifier*

An attempt was made to use a symbol that was not defined.

One of the following may have occurred:

- A symbol was not defined.
- A field was not a member of the specified structure.
- A symbol was defined in an include file that was not included.
- An external symbol was used without an **EXTERN** or **EXTERNDEF** directive.
- A symbol name was misspelled.
- A local code label was referenced outside of its scope.

A2007 **non-benign record redefinition**

A RECORD definition conflicted with a previous definition.

One of the following occurred:

- There were different numbers of fields.
- There were different numbers of bits in a field.
- There was a different label.
- There were different initializers.

A2008 **syntax error :**

A token at the current location caused a syntax error.

One of the following may have occurred:

- A dot prefix was added to or omitted from a directive.
- A reserved word (such as **C** or **SIZE**) was used as an identifier.
- An instruction was used that was not available with the current processor or coprocessor selection.
- A comparison run-time operator (such as ==) was used in a conditional assembly statement instead of a relational operator (such as **EQ**).
- An instruction or directive was given too few operands.
- An obsolete directive was used.

A2009 **syntax error in expression**

An expression on the current line contained a syntax error. This error message may also be a side-effect of a preceding program error.

A2010 **invalid type expression**

The operand to **THIS** or **PTR** was not a valid type expression.

A2011 **distance invalid for word size of current segment**

A procedure definition or a code label defined with **LABEL** specified an address size that was incompatible with the current segment size.

One of the following occurred:

- A **NEAR16** or **FAR16** procedure was defined in a 32-bit segment.
- A **NEAR32** or **FAR32** procedure was defined in a 16-bit segment.
- A code label defined with **LABEL** specified **FAR16** or **NEAR16** in a 32-bit segment.
- A code label defined with **LABEL** specified **FAR32** or **NEAR32** in a 16-bit segment.

A2012 **PROC, MACRO, or macro repeat directive must precede LOCAL**

A **LOCAL** directive must be immediately preceded by a **MACRO**, **PROC**, macro repeat directive (such as **REPEAT**, **WHILE**, or **FOR**), or another **LOCAL** directive.

A2013 **.MODEL must precede this directive**

A simplified segment directive or a **.STARTUP** or **.EXIT** directive was not preceded by a **.MODEL** directive.

A **.MODEL** directive must specify the model defaults before a simplified segment directive, or a **.STARTUP** or **.EXIT** directive may be used.

A2014 **cannot define as public or external :** *identifier*

Only labels, procedures, and numeric equates can be made public or external using **PUBLIC**, **EXTERN**, or **EXTERNDEF**. Local code labels cannot be made public.

A2015 **segment attributes cannot change :** *attribute*

A segment was reopened with different attributes than it was opened with originally.

When a **SEGMENT** directive opens a previously defined segment, the newly opened segment inherits the attributes the segment was defined with.

A2016 **expression expected**

The assembler expected an expression at the current location but found one of the following:

- A unary operator without an operand
- A binary operator without two operands
- An empty pair of parentheses, ( ), or brackets, [ ]

A2017 **operator expected**

An expression operator was expected at the current location.

One possible cause of this error is a missing comma between expressions in an expression list.

A2018 **invalid use of external symbol :** *identifier*

An attempt was made to compare the given external symbol using a relational operator.

The comparison cannot be made because the value or address of an external symbol is not known at assembly time.

A2019 **operand must be RECORD type or field**

The operand following the **WIDTH** or **MASK** operator was not valid.

The **WIDTH** operator takes an operand that is the name of a field or a record. The **MASK** operator takes an operand that is the name of a field or a record type.

A2020 **identifier not a record :** *identifier*

A record type was expected at the current location.

A2021 **record constants cannot span line breaks**

A record constant must be defined on one physical line. A line ended in the middle of the definition of a record constant.

A2022 **instruction operands must be the same size**

The operands to an instruction did not have the same size.

A2023 **instruction operand must have size**

At least one of the operands to an instruction must have a known size.

A2024 **invalid operand size for instruction**

The size of an operand was not valid.

A2025 **operands must be in same segment**

Relocatable operands used with a relational or minus operator were not located in the same segment.

A2026 **constant expected**

The assembler expected a constant expression at the current location. A constant expression is a numeric expression that can be resolved at assembly time.

A2027 **operand must be a memory expression**

The right operand of a **PTR** expression was not a memory expression.

When the left operand of the **PTR** operator is a structure or union type, the right operand must be a memory expression.

A2028 expression must be a code address

An expression evaluating to a code address was expected.

One of the following occurred:

- **SHORT** was not followed by a code address.
- **NEAR PTR** or **FAR PTR** was applied to something that was not a code address.

A2029 multiple base registers not allowed

An attempt was made to combine two base registers in a memory expression.

For example, the following expressions cause this error:

```
[bx+bp]
[bx][bp]
```

In another example, given the following definition:

```
id1 proc arg1:byte
```

either of the following lines causes this error:

```
mov al, [bx].arg1
lea ax, arg1[bx]
```

A2030 multiple index registers not allowed

An attempt was made to combine two index registers in a memory expression.

For example, the following expressions cause this error:

```
[si+di]
[di][si]
```

A2031 must be index or base register

An attempt was made to use a register that was not a base or index register in a memory expression.

For example, the following expressions cause this error:

```
[ax]
[bl]
```

A2032 invalid use of register

An attempt was made to use a register that was not valid for the intended use.

One of the following occurred:

- **OFFSET** was applied to a register. (**OFFSET** can be applied to a register under the **M510** option.)
- A special 386 register was used in an invalid context.
- A register was cast with **PTR** to a type of invalid size.
- A register was specified as the right operand of a segment override operator (**:**).
- A register was specified as the right operand of a binary minus operator (–).
- An attempt was made to multiply registers using the ***** operator.
- Brackets (**[ ]**) were missing around a register that was added to something.

A2033 invalid INVOKE argument : argument *number*

The **INVOKE** directive was passed a special 386 register, or a register pair containing a byte register or special 386 register. These registers are illegal with **INVOKE**.

A2034 must be in segment block

One of the following was found outside of a segment block:

- An instruction
- A label definition
- A **THIS** operator
- A **$** operator
- A procedure definition
- An **ALIGN** directive
- An **ORG** directive

A2035 DUP too complex

A declaration using the **DUP** operator resulted in a data structure with an internal representation that was too large.

A2036 too many initial values for structure: *structure*

The given structure was defined with more initializers than the number of fields in the type declaration of the structure.

A2037 statement not allowed inside structure definition

A structure definition contained an invalid statement.

A structure cannot contain instructions, labels, procedures, control-flow directives, **.STARTUP**, or **.EXIT**.

A2038 **missing operand for macro operator**

The assembler found the end of a macro's parameter list immediately after the **!** or **%** operator.

A2039 **line too long**

A source-file line exceeded the limit of 512 characters.

If multiple physical lines are concatenated with the line-continuation character (\ ), the resulting logical line is still limited to 512 characters.

A2040 **segment register not allowed in context**

A segment register was specified for an instruction that cannot take a segment register.

A2041 **string or text literal too long**

A string or text literal, or a macro function return value, exceeded the limit of 255 characters.

A2042 **statement too complex**

A statement was too complex for the assembler to parse.

Reduce either the number of tokens or the number of forward-referenced identifiers.

A2043 **identifier too long**

An identifier exceeded the limit of 247 characters.

A2044 **invalid character in file**

The source file contained a character outside a comment, string, or literal that was not recognized as an operator or other legal character.

A2045 **missing angle bracket or brace in literal**

An unmatched angle bracket (either < or >) or brace (either **{** or **}**) was found in a literal constant or an initializer.

One of the following occurred:

- A pair of angle brackets or braces was not complete.
- An angle bracket was intended to be literal, but it was not preceded by an exclamation point (**!**) to indicate a literal character.

A2046 missing single or double quotation mark in string

An unmatched quotation mark (either ' or ") was found in a string.

One of the following may have occurred:

- A pair of quotation marks around a string was not complete.
- A pair of quotation marks around a string was formed of one single and one double quotation mark.
- A single or double quotation mark was intended to be literal, but the surrounding quotation marks were the same kind as the literal one.

A2047 empty (null) string

A string consisted of a delimiting pair of quotation marks and no characters within.

For a string to be valid, it must contain 1–255 characters.

A2048 nondigit in number

A number contained a character that was not in the set of characters used by the current radix (base).

This error can occur if a B or D radix specifier is used when the default radix is one that includes that letter as a valid digit.

A2049 syntax error in floating-point constant

A floating-point constant contained an invalid character.

A2050 real or BCD number not allowed

A floating-point (real) number or binary coded decimal (BCD) constant was used other than as a data initializer.

One of the following occurred:

- A real number or a BCD was used in an expression.
- A real number was used to initialize a directive other than **DWORD**, **QWORD**, or **TBYTE**.
- A BCD was used to initialize a directive other than **TBYTE**.

A2051 text item required

A literal constant or text macro was expected.

One of the following was expected:

- A literal constant, which is text enclosed in < >
- A text macro name
- A macro function call
- A % followed by a constant expression

A2052 **forced error**

The conditional-error directive **.ERR** or **.ERR1** was used to generate this error.

A2053 **forced error : value equal to 0**

The conditional-error directive **.ERRE** was used to generate this error.

A2054 **forced error : value not equal to 0**

The conditional-error directive **.ERRNZ** was used to generate this error.

A2055 **forced error : symbol not defined**

The conditional-error directive **.ERRNDEF** was used to generate this error.

A2056 **forced error : symbol defined**

The conditional-error directive **.ERRDEF** was used to generate this error.

A2057 **forced error : string blank**

The conditional-error directive **.ERRB** was used to generate this error.

A2058 **forced error : string not blank**

The conditional-error directive **.ERRNB** was used to generate this error.

A2059 **forced error : strings equal**

The conditional-error directive **.ERRIDN** or **.ERRIDNI** was used to generate this error.

A2060 **forced error : strings not equal**

The conditional-error directive **.ERRDIF** or **.ERRDIFI** was used to generate this error.

A2061 **[[ELSE]]IF2/.ERR2 not allowed : single-pass assembler**

A directive for a two-pass assembler was found.

The Microsoft Macro Assembler (MASM) is a one-pass assembler. MASM does not accept the **IF2**, **ELSEIF2**, and **.ERR2** directives.

This error also occurs if an **ELSE** directive follows an **IF1** directive.

A2062 **expression too complex for .UNTILCXZ**

An expression used in the condition that follows **.UNTILCXZ** was too complex.

The **.UNTILCXZ** directive can take only one expression, which can contain only **==** or **!=**. It cannot take other comparison operators or more complex expressions using operators like **||**.

A2063 **can ALIGN only to power of 2 :** *expression*

The expression specified with the **ALIGN** directive was invalid.

The **ALIGN** expression must be a power of 2 between 2 and 256, and must be less than or equal to the alignment of the current segment, structure, or union.

A2064 **structure alignment must be 1, 2, or 4**

The alignment specified in a structure definition was invalid.

A2065 **expected :** *token*

The assembler expected the given token.

A2066 **incompatible CPU mode and segment size**

An attempt was made to open a segment with a **USE16**, **USE32**, or **FLAT** attribute that was not compatible with the specified CPU, or to change to a 16-bit CPU while in a 32-bit segment.

The **USE32** and **FLAT** attributes must be preceded by one of the following processor directives: **.386**, **.386C**, **.386P**, **.486**, or **.486P**.

A2067 **LOCK must be followed by a memory operation**

The **LOCK** prefix preceded an invalid instruction. No instruction can take the **LOCK** prefix unless one of its operands is a memory expression.

A2068 **instruction prefix not allowed**

One of the prefixes **REP**, **REPE**, **REPNE**, or **LOCK** preceded an instruction for which it was not valid.

A2069 **no operands allowed for this instruction**

One or more operands were specified with an instruction that takes no operands.

A2070 **invalid instruction operands**

One or more operands were not valid for the instruction they were specified with.

A2071 **initializer too large for specified size**

An initializer value was too large for the data area it was initializing.

A2072 **cannot access symbol in given segment or group:** *identifier*

The given identifier cannot be addressed from the segment or group specified.

A2073 **operands have different frames**

Two operands in an expression were in different frames.

Subtraction of pointers requires the pointers to be in the same frame. Subtraction of two expressions that have different effective frames is not allowed. An effective frame is calculated from the segment, group, or segment register.

A2074 **cannot access label through segment registers**

An attempt was made to access a label through a segment register that was not assumed to its segment or group.

A2075 jump destination too far [: by 'n' bytes]

The destination specified with a jump instruction was too far from the instruction.

One of the following may be a solution:

- Enable the **LJMP** option.
- Remove the **SHORT** operator. If **SHORT** has forced a jump that is too far, *n* is the number of bytes out of range.
- Rearrange code so that the jump is no longer out of range.

A2076 jump destination must specify a label

A direct jump's destination must be relative to a code label.

A2077 instruction does not allow NEAR indirect addressing

A conditional jump or loop cannot take a memory operand. It must be given a relative address or label.

A2078 instruction does not allow FAR indirect addressing

A conditional jump or loop cannot take a memory operand. It must be given a relative address or label.

A2079 instruction does not allow FAR direct addressing

A conditional jump or loop cannot be to a different segment or group.

A2080 jump distance not possible in current CPU mode

A distance was specified with a jump instruction that was incompatible with the current processor mode.

For example, 48-bit jumps require **.386** or above.

A2081 missing operand after unary operator

An operator required an operand, but no operand followed.

A2082 cannot mix 16- and 32-bit registers

An address expression contained both 16- and 32-bit registers.

For example, the following expression causes this error:

```
[bx+edi]
```

A2083 invalid scale value

A register scale was specified that was not 1, 2, 4, or 8.

A2084 constant value too large

A constant was specified that was too big for the context in which it was used.

A2085 **instruction or register not accepted in current CPU mode**

An attempt was made to use an instruction, register, or keyword that was not valid for the current processor mode.

For example, 32-bit registers require **.386** or above. Control registers such as CR0 require privileged mode **.386P** or above. This error will also be generated for the **NEAR32**, **FAR32**, and **FLAT** keywords, which require .**386** or above.

A2086 **reserved word expected**

One or more items in the list specified with a **NOKEYWORD** option were not recognized as reserved words.

A2087 **instruction form requires 80386/486**

An instruction was used that was not compatible with the current processor mode.

One of the following processor directives must precede the instruction: **.386**, **.386C**, **.386P**, **.486**, or **.486P**.

A2088 **END directive required at end of file**

The assembler reached the end of the main source file and did not find an **.END** directive.

A2089 **too many bits in RECORD :** *identifier*

One of the following occurred:

- Too many bits were defined for the given record field.
- Too many total bits were defined for the given record.

The size limit for a record or a field in a record is 16 bits when doing 16-bit arithmetic or 32 bits when doing 32-bit arithmetic.

A2090 **positive value expected**

A positive value was not found in one of the following situations:

- The starting position specified for **SUBSTR** or **@SubStr**
- The number of data objects specified for **COMM**
- The element size specified for **COMM**

A2091 **index value past end of string**

An index value exceeded the length of the string it referred to when used with **INSTR**, **SUBSTR**, **@InStr**, or **@SubStr**.

A2092 **count must be positive or zero**

The operand specified to the **SUBSTR** directive, **@SubStr** macro function, **SHL** operator, **SHR** operator, or **DUP** operator was negative.

A2093 count value too large

The length argument specified for **SUBSTR** or **@SubStr** exceeded the length of the specified string.

A2094 operand must be relocatable

An operand was not relative to a label.

One of the following occurred:

- An operand specified with the **END** directive was not relative to a label.
- An operand to the **SEG** operator was not relative to a label.
- The right operand to the minus operator was relative to a label, but the left operand was not.
- The operands to a relational operator were either not both integer constants or not both memory operands. Relational operators can take operands that are both addresses or both non-addresses but not one of each.

A2095 constant or relocatable label expected

The operand specified must be a constant expression or a memory offset.

A2096 segment, group, or segment register expected

A segment or group was expected but was not found.

One of the following occurred:

- The left operand specified with the segment override operator (**:**) was not a segment register (CS, DS, SS, ES, FS, or GS), group name, segment name, or segment expression.
- The **ASSUME** directive was given a segment register without a valid segment address, segment register, group, or the special **FLAT** group.

A2097 segment expected : *identifier*

The **GROUP** directive was given an identifier that was not a defined segment.

A2098 invalid operand for OFFSET

The expression following the **OFFSET** operator must be a memory expression or an immediate expression.

A2099 invalid use of external absolute

An attempt was made to subtract a constant defined in another module from an expression.

You can avoid this error by placing constants in include files rather than making them external.

A2100 segment or group not allowed

An attempt was made to use a segment or group in a way that was not valid. Segments or groups cannot be added.

A2101 **cannot add two relocatable labels**

An attempt was made to add two expressions that were both relative to a label.

A2102 **cannot add memory expression and code label**

An attempt was made to add a code label to a memory expression.

A2103 **segment exceeds 64K limit**

A 16-bit segment exceeded the size limit of 64K.

A2104 **invalid type for data declaration :** *type*

The given type was not valid for a data declaration.

A2105 **HIGH and LOW require immediate operands**

The operand specified with either the **HIGH** or the **LOW** operator was not an immediate expression.

A2107 **cannot have implicit far jump or call to near label**

An attempt was made to make an implicit far jump or call to a near label in another segment.

A2108 **use of register assumed to ERROR**

An attempt was made to use a register that had been assumed to ERROR with the **ASSUME** directive.

A2109 **only white space or comment can follow backslash**

A character other than a semicolon (;) or a white-space character (spaces or TAB characters) was found after a line-continuation character (\ ).

A2110 **COMMENT delimiter expected**

A delimiter character was not specified for a **COMMENT** directive.

The delimiter character is specified by the first character that is not white space (spaces or TAB characters) after the **COMMENT** directive. The comment consists of all text following the delimiter until the end of the line containing the next appearance of the delimiter.

A2111 **conflicting parameter definition**

A procedure defined with the **PROC** directive did not match its prototype as defined with the **PROTO** directive.

A2112 **PROC and prototype calling conventions conflict**

A procedure was defined in a prototype (using the **PROTO**, **EXTERNDEF**, or **EXTERN** directive), but the calling convention did not match the corresponding **PROC** directive.

A2113 **invalid radix tag**

The specified radix was not a number in the range 2–16.

A2114 **INVOKE argument type mismatch : argument** *number*

The type of the arguments passed using the **INVOKE** directive did not match the type of the parameters in the prototype of the procedure being invoked.

A2115 **invalid coprocessor register**

The coprocessor index specified was negative or greater than 7.

A2116 **instructions and initialized data not allowed in AT segments**

An instruction or initialized data was found in a segment defined with the **AT** attribute.

Data in **AT** segments must be declared with the **?** initializer.

A2117 **/AT option requires TINY memory model**

The /AT option was specified on the assembler command line, but the program being assembled did not specify the **TINY** memory model with the **.MODEL** directive.

This error is only generated for modules that specify a start address or use the **.STARTUP** directive.

A2118 **cannot have segment address references with TINY model**

An attempt was made to reference a segment in a **TINY** model program.

All **TINY** model code and data must be accessed with **NEAR** addresses.

A2119 **language type must be specified**

A procedure definition or prototype was not given a language type.

A language type must be declared in each procedure definition or prototype if a default language type is not specified. A default language type is set using either the **.MODEL** directive, **OPTION LANG**, or the ML command-line options /Gc or /Gd.

A2120 **PROLOGUE must be macro function**

The identifier specified with the **OPTION PROLOGUE** directive was not recognized as a defined macro function.

The user-defined prologue must be a macro function that returns the number of bytes needed for local variables and any extra space needed for the macro function.

A2121 **EPILOGUE must be macro procedure**

The identifier specified with the **OPTION EPILOGUE** directive was not recognized as a defined macro procedure.

The user-defined epilogue macro cannot return a value.

A2122 **alternate identifier not allowed with EXTERNDEF**

An attempt was made to specify an alternate identifier with an **EXTERNDEF** directive.

You can specify an optional alternate identifier with the **EXTERN** directive but not with **EXTERNDEF**.

A2123 **text macro nesting level too deep**

A text macro was nested too deeply. The nesting limit for text macros is 40.

A2125 **missing macro argument**

A required argument to **@InStr**, **@SubStr**, or a user-defined macro was not specified.

A2126 **EXITM used inconsistently**

The **EXITM** directive was used both with and without a return value in the same macro.

A macro procedure returns a value; a macro function does not.

A2127 **macro function argument list too long**

There were too many characters in a macro function's argument list. This error applies also to a prologue macro function called implicitly by the **PROC** directive.

A2129 **VARARG parameter must be last parameter**

A parameter other than the last one was given the **VARARG** attribute.

The **:VARARG** specification can be applied only to the last parameter in a parameter list for macro and procedure definitions and prototypes. You cannot use multiple **:VARARG** specifications in a macro.

A2130 **VARARG parameter not allowed with LOCAL**

An attempt was made to specify **:VARARG** as the type in a procedure's **LOCAL** declaration.

A2131 **VARARG parameter requires C calling convention**

A **VARARG** parameter was specified in a procedure definition or prototype, but the **C**, **SYSCALL**, or **STDCALL** calling convention was not specified.

A2132 **ORG needs a constant or local offset**

The expression specified with the **ORG** directive was not valid.

ORG requires an immediate expression with no reference to an external label or to a label outside the current segment.

A2133 **register value overwritten by INVOKE**

A register was passed as an argument to a procedure, but the code generated by **INVOKE** to pass other arguments destroyed the contents of the register.

The AX, AL, AH, EAX, DX, DL, DH, and EDX registers may be used by the assembler to perform data conversion.

Use a different register.

A2134 **structure too large to pass with INVOKE : argument** *number*

An attempt was made with **INVOKE** to pass a structure that exceeded 255 bytes.

Pass structures by reference if they are larger than 255 bytes.

A2136 **too many arguments to INVOKE**

The number of arguments passed using the **INVOKE** directive exceeded the number of parameters in the prototype for the procedure being invoked.

A2137 **too few arguments to INVOKE**

The number of arguments passed using the **INVOKE** directive was fewer than the number of required parameters specified in the prototype for the procedure being invoked.

A2138 **invalid data initializer**

The initializer list for a data definition was invalid.

This error can be caused by using the R radix override with too few digits.

A2140 **RET operand too large**

The operand specified to **RET**, **RETN**, or **RETF** exceeded two bytes.

A2141 **too many operands to instruction**

Too many operands were specified with a string control instruction.

A2142 **cannot have more than one .ELSE clause per .IF block**

The assembler found more than one **.ELSE** clause within the current **.IF** block.

Use **.ELSEIF** for all but the last block.

A2143 **expected data label**

The **LENGTHOF**, **SIZEOF**, **LENGTH**, or **SIZE** operator was applied to a non-data label, or the **SIZEOF** or **SIZE** operator was applied to a type.

A2144 **cannot nest procedures**

An attempt was made to nest a procedure containing a parameter, local variable, **USES** clause, or a statement that generated a new segment or group.

A2145 **EXPORT must be FAR :** *procedure*

The given procedure was given **EXPORT** visibility and **NEAR** distance.

All **EXPORT** procedures must be **FAR**. The default visibility may have been set with the **OPTION PROC:EXPORT** statement or the **SMALL** or **COMPACT** memory models.

A2146 **procedure declared with two visibility attributes :** *procedure*

The given procedure was given conflicting visibilities.

A procedure was declared with two different visibilities (**PUBLIC**, **PRIVATE**, or **EXPORT**). The **PROC** and **PROTO** statements for a procedure must have the same visibility.

A2147 **macro label not defined :** *macrolabel*

The given macro label was not found.

A macro label is defined with **:***macrolabel*.

A2148 **invalid symbol type in expression :** *identifier*

The given identifier was used in an expression in which it was not valid.

For example, a macro procedure name is not allowed in an expression.

A2149 **byte register cannot be first operand**

A byte register was specified to an instruction that cannot take it as the first operand.

A2150 **word register cannot be first operand**

A word register was specified to an instruction that cannot take it as the first operand.

A2151 **special register cannot be first operand**

A special register was specified to an instruction that cannot take it as the first operand.

A2152 **coprocessor register cannot be first operand**

A coprocessor (stack) register was specified to an instruction that cannot take it as the first operand.

A2153 **cannot change size of expression computations**

An attempt was made to set the expression word size when the size had been already set using the **EXPR16**, **EXPR32**, **SEGMENT:USE32**, or **SEGMENT:FLAT** option or the **.386** or higher processor selection directive.

A2154 **syntax error in control-flow directive**

The condition for a control-flow directive (such as **.IF** or **.WHILE**) contained a syntax error.

A2155 **cannot use 16-bit register with a 32-bit address**

An attempt was made to mix 16-bit and 32-bit offsets in an expression.

Use a 32-bit register with a symbol defined in a 32-bit segment.

For example, if `id1` is defined in a 32-bit segment, the following causes this error:

```
id1[bx]
```

A2156 **constant value out of range**

An invalid value was specified for the **PAGE** directive.

The first parameter of the **PAGE** directive can be either 0 or a value in the range 10–255. The second parameter of the **PAGE** directive can be either 0 or a value in the range 60–255.

A2157 **missing right parenthesis**

A right parenthesis,), was missing from a macro function call.

Be sure that parentheses are in pairs if nested.

A2158 type is wrong size for register

An attempt was made to assume a general-purpose register to a type with a different size than the register.

For example, the following pair of statements causes this error:

```
ASSUME bx:far ptr byte ; far pointer is 4 or 6 bytes
ASSUME al:word         ; al is a byte reg, cannot hold word
```

A2159 structure cannot be instanced

An attempt was made to create an instance of a structure when there were no fields or data defined in the structure definition or when **ORG** was used in the structure definition.

A2160 non-benign structure redefinition : label incorrect

A label given in a structure redefinition either did not exist in the original definition or was out of order in the redefinition.

A2161 non-benign structure redefinition : too few labels

Not enough members were defined in a structure redefinition.

A2162 OLDSTRUCT/NOOLDSTRUCT state cannot be changed

Once the **OLDSTRUCTS** or **NOOLDSTRUCTS** option has been specified and a structure has been defined, the structure scoping cannot be altered or respecified in the same module.

A2163 non-benign structure redefinition : incorrect initializers

A **STRUCT** or **UNION** was redefined with a different initializer value.

When structures and unions are defined more than once, the definitions must be identical. This error can be caused by using a variable as an initializer and having the value of the variable change between definitions.

A2164 non-benign structure redefinition : too few initializers

A **STRUCT** or **UNION** was redefined with too few initializers.

When structures and unions are defined more than once, the definitions must be identical.

A2165 non-benign structure redefinition : label has incorrect offset

The offset of a label in a redefined **STRUCT** or **UNION** differs from the original definition.

When structures and unions are defined more than once, the definitions must be identical. This error can be caused by a missing member or by a member that has a different size than in its original definition.

A2166 structure field expected

The righthand side of a dot operator (**.**) is not a structure field.

This error may occur with some code acceptable to previous versions of the assembler. To enable the old behavior, use **OPTION OLDSTRUCTS**, which is automatically enabled by **OPTION M510** or the /Zm command-line option.

A2167 unexpected literal found in expression

A literal was found where an expression was expected.

One of the following may have occurred:

- A literal was used as an initializer
- A record tag was omitted from a record constant

A2169 divide by zero in expression

An expression contains a divisor whose value is equal to zero.

Check that the syntax of the expression is correct and that the divisor (whether constant or variable) is correctly initialized.

A2170 directive must appear inside a macro

A **GOTO** or **EXITM** directive was found outside the body of a macro.

A2171 cannot expand macro function

A syntax error prevented the assembler from expanding the macro function.

A2172 too few bits in RECORD

There was an attempt to define a record field of 0 bits.

A2173 macro function cannot redefine itself

There was an attempt to define a macro function inside the body of a macro function with the same name. This error can also occur when a member of a chain of macros attempts to redefine a previous member of the chain.

A2175 invalid qualified type

An identifier was encountered in a qualified type that was not a type, structure, record, union, or prototype.

A2176 floating point initializer on an integer variable

An attempt was made to use a floating-point initializer with **DWORD**, **QWORD**, or **TBYTE**. Only integer initializers are allowed.

A2177 nested structure improperly initialized

The nested structure initialization could not be resolved.

This error can be caused by using different beginning and ending delimiters in a nested structure initialization.

A2178 invalid use of FLAT

There was an ambiguous reference to **FLAT** as a group.

This error is generated when there is a reference to **FLAT** instead of a **FLAT** subgroup. For example,

```
mov    ax, FLAT              ; Generates A2178
mov    ax, SEG FLAT:_data    ; Correct
```

A2179 structure improperly initialized

There was an error in a structure initializer.

One of the following occurred:

- The initializer is not a valid expression.
- The initializer is an invalid **DUP** statement.

A2180 improper list initialization

In a structure, there was an attempt to initialize a list of items with a value or list of values of the wrong size.

A2181 initializer must be a string or single item

There was an attempt to initialize a structure element with something other than a single item or string.

This error can be caused by omitting braces (**{ }**) around an initializer.

A2182 initializer must be a single item

There was an attempt to initialize a structure element with something other than a single item.

This error can be caused by omitting braces (**{ }**) around an initializer.

A2183 initializer must be a single byte

There was an attempt to initialize a structure element of byte size with something other than a single byte.

A2184 improper use of list initializer

The assembler did not expect an opening brace (**{**) at this point.

A2185 improper literal initialization

A literal structure initializer was not properly delimited.

This error can be caused by missing angle brackets (< >) or braces (**{ }**) around an initializer or by extra characters after the end of an initializer.

A2186 extra characters in literal initialization

A literal structure initializer was not properly delimited.

One of the following may have occurred:

- There were missing or mismatched angle brackets (< >) or braces (**{ }**) around an initializer.
- There were extra characters after the end of an initializer.
- There was a syntax error in the structure initialization.

A2187 must use floating point initializer

A variable declared with the **REAL4**, **REAL8**, and **REAL10** directives must be initialized with a floating-point number or a question mark (**?**).

This error can be caused by giving an initializer in integer form (such as 18) instead of in floating-point form (18.0).

A2188 cannot use .EXIT for OS_OS2 with .8086

The **INVOKE** generated by the **.EXIT** statement under **OS_OS2** requires the **.186** (or higher) directive, since it must be able to use the **PUSH** instruction to push immediates directly.

A2189 invalid combination with segment alignment

The alignment specified by the **ALIGN** or **EVEN** directive was greater than the current segment alignment as specified by the **SEGMENT** directive.

A2190 INVOKE requires prototype for procedure

The **INVOKE** directive must be preceded by a **PROTO** statement for the procedure being called.

When using **INVOKE** with an address rather than an explicit procedure name, you must precede the address with a pointer to the prototype.

A2191 cannot include structure in self

You cannot reference a structure recursively (inside its own definition).

A2192 symbol language attribute conflict

Two declarations for the same symbol have conflicting language attributes (such as **C** and **PASCAL**). The attributes should be identical or compatible.

A2193 non-benign COMM redefinition

A variable was redefined with the **COMM** directive to a different language type, distance, size, or instance count.

Multiple **COMM** definitions of a variable must be identical.

A2194 COMM variable exceeds 64K

A variable declared with the **COMM** directive in a 16-bit segment was greater than 64K.

A2195 parameter or local cannot have void type

The assembler attempted to create an argument or create a local without a type.

This error can be caused by declaring or passing a symbol followed by a colon without specifying a type or by using a user-defined type defined as void.

A2196 cannot use TINY model with OS_OS2

A **.MODEL** statement specified the **TINY** memory model and the **OS_OS2** operating system. The tiny memory model is not allowed under OS/2.

A2197 expression size must be 32-bits

There was an attempt to use the 16-bit expression evaluator in a 32-bit segment. In a 32-bit segment (**USE32** or **FLAT**), you cannot use the default 16-bit expression evaluator (**OPTION EXPR16**).

A2198 .EXIT does not work with 32-bit segments

The **.EXIT** directive cannot be used in a 32-bit segment; it is valid only when generating 16-bit code.

A2199 .STARTUP does not work with 32-bit segments

The **.STARTUP** directive cannot be used in a 32-bit segment; it is valid only when generating 16-bit code.

A2200 ORG directive not allowed in unions

The **ORG** directive is not valid inside a **UNION** definition.

You can use the **ORG** directive inside **STRUCT** definitions, but it is meaningless inside a **UNION**.

A2201 scope state cannot be changed

Both **OPTION SCOPED** and **OPTION NOSCOPED** statements occurred in a module. You cannot switch scoping behavior in a module.

This error may be caused by an **OPTION SCOPED** or **OPTION NOSCOPED** statement in an include file.

A2202 illegal use of segment register

You cannot use segment overrides for the FS or GS segment registers when generating floating-point emulation instructions with the /FPi command-line option or **OPTION EMULATOR**.

A2203 cannot declare scoped code label as PUBLIC

A code label defined with the “label:” syntax was declared **PUBLIC**. Use the “label::” syntax, the **LABEL** directive, or **OPTION NOSCOPED** to eliminate this error.

A2204 .MSFLOAT directive is obsolete : ignored

The Microsoft Binary Format is no longer supported. You should convert your code to the IEEE numeric standard, which is used in the 80x87-series coprocessors.

A2205 ESC instruction is obsolete : ignored

The **ESC** (Escape) instruction is no longer supported. All numeric coprocessor instructions are now supported directly by the assembler.

A2206 missing operator in expression

An expression cannot be evaluated because it is missing an operator. This error message may also be a side-effect of a preceding program error.

The following line will generate this error:

```
value1 = ( 1 + 2 ) 3
```

A2207 missing right parenthesis in expression

An expression cannot be evaluated because it is missing a right (closing) parenthesis. This error message may also be a side-effect of a preceding program error.

The following line will generate this error:

```
value1 = ( ( 1 + 2 ) * 3
```

A2208 missing left parenthesis in expression

An expression cannot be evaluated because it is missing a left (opening) parenthesis. This error message may also be a side-effect of a preceding program error.

The following line will generate this error:

```
value1 = ( ( 1 + 2 ) * 3 ) )
```

A2209 reference to forward macro redefinition

A macro cannot be accessed because it has not been yet defined.

Move the macro definition ahead of all references to the macro.

A2901 cannot run ML.EXE

The MASM driver could not spawn ML.EXE.

One of the following may have occurred:

- ML.EXE was not in the path.
- The READ attribute was not set on ML.EXE.
- There was not enough memory.

ML Warnings

A4000 cannot modify READONLY segment

An attempt was made to modify an operand in a segment marked with the READONLY attribute.

A4002 **non-unique STRUCT/UNION field used without qualification**

A **STRUCT** or **UNION** field can be referenced without qualification only if it has a unique identifier.

This conflict can be resolved either by renaming one of the structure fields to make it unique or by fully specifying both field references.

The **NONUNIQUE** keyword requires that all references to the elements of a **STRUCT** or **UNION** be fully specified.

A4003 **start address on END directive ignored with .STARTUP**

Both **.STARTUP** and a program load address (optional with the **END** directive) were specified. The address specification with the **END** directive was ignored.

A4004 **cannot ASSUME CS**

An attempt was made to assume a value for the CS register. CS is always set to the current segment or group.

A4005 **unknown default prologue argument**

An unknown argument was passed to the default prologue.

The default prologue understands only the **FORCEFRAME** and **LOADDS** arguments.

A4006 **too many arguments in macro call**

There were more arguments given in the macro call than there were parameters in the macro definition.

A4007 **option untranslated, directive required :** *option*

There is no ML command-line equivalent for the given MASM option. The desired behavior can be obtained by using a directive in the source file.

Option	Directive
/A	.ALPHA
/P	OPTION READONLY
/S	.SEQ

A4008 **invalid command-line option value, default is used :** *option*

The value specified with the given option was not valid. The option was ignored, and the default was assumed.

A4009 **insufficient memory for /EP : /EP ignored**

There is not enough memory to generate a first-pass listing.

A4010 **expected '>' on text literal**

A macro was called with a text literal argument that was missing a closing angle bracket.

A4011 **multiple .MODEL directives found : .MODEL ignored**

More than one **.MODEL** directive was found in the current module. Only the first **.MODEL** statement is used.

A4012 **line number information for segment without class 'CODE'**

There were instructions in a segment that did not have a class name that ends with "CODE." The assembler did not generate CodeView information for these instructions.

CodeView cannot process modules with code in segments with class names that do not end with "CODE."

A4013 **instructions and initialized data not supported in AT segments**

An instruction or initialized data was found in a segment defined with the AT attribute. The code or data will not be loaded at run time.

Data in AT segments must be declared with the **?** initializer.

A4910 **cannot open file:** *filename*

The given filename could not be in the current path.

Make sure that *filename* was copied from the distribution disks and is in the current path.

A5000 **@@: label defined but not referenced**

A jump target was defined with the **@@:** label, but the target was not used by a jump instruction.

One common cause of this error is insertion of an extra **@@:** label between the jump and the **@@:** label that the jump originally referred to.

A5001 **expression expected, assume value 0**

There was an **IF**, **ELSEIF**, **IFE**, **IFNE**, **ELSEIFE**, or **ELSEIFNE** directive without an expression to evaluate. The assembler assumes a 0 for the comparison expression.

A5002 **externdef previously assumed to be external**

The **OPATTR** or **.TYPE** operator was applied to a symbol after the symbol was used in an **EXTERNDEF** statement but before it was declared. These operators were used on a line where the assembler assumed that the symbol was external.

A5003 **length of symbol previously assumed to be different**

The **LENGTHOF**, **LENGTH**, **SIZEOF**, or **SIZE** operator was applied to a symbol after the symbol was used in an **EXTERNDEF** statement but before it was declared. These operators were used on a line where the assembler assumed that the symbol had a different length and size.

A5004 **symbol previously assumed to not be in a group**

A symbol was used in an **EXTERNDEF** statement outside of a segment and then was declared inside a segment.

A5005 types are different

The type given by an **INVOKE** statement differed from that given in the procedure prototype. The assembler performed the appropriate type conversion.

A6001 no return from procedure

A **PROC** statement generated a prologue, but there was no **RET** or **IRET** instruction found inside the procedure block.

A6003 conditional jump lengthened

A conditional jump was encoded as a reverse conditional jump around a near unconditional jump.

You may be able to rearrange code to avoid the longer form.

A6004 procedure argument or local not referenced

You passed a procedure argument or created a variable with the LOCAL directive that was not used in the procedure body.

Unnecessary parameters and locals waste code and stack space.

A6005 expression condition may be pass-dependent

Under the /Zm command-line option or the OPTION M510 directive, the value of an expression changed between passes.

This error message may indicate that the code is pass-dependent and must be rewritten.